Time series forecasting

Sunspot prediction

Name:	**Manideep Pendyala**
Mentor:	Dr. Saurabh Das
Start date:	March 01, 2022
End date:	June 30, 2022

Abstract

This report sums up the work I performed on time series forecasting under Dr.Saurabh Das of IIT Indore from march to june 2022 as part of my Internship.

1 Introduction

1.1 Time Series

In mathematics, a time series is a series of data points indexed (or listed or graphed) in time order that is sequence taken at successive equally spaced points in time. Thus it is a sequence of discrete-time data. Examples of time series are heights of ocean tides, counts of sunspots, and the daily closing value of the Dow Jones Industrial Average.

Time series forecasting is the use of a model to predict future values based on previously observed values. While regression analysis is often employed in such a way as to test relationships between one or more different time series, this type of analysis is not usually called "time series analysis", which refers in particular to relationships between different points in time within a single series.

A stochastic model for a time series will generally reflect the fact that observations close together in time will be more closely related than observations further apart. In addition, time series models will often make use of the natural one-way ordering of time so that values for a given period will be expressed as deriving in some way from past values, rather than from future values. Time series analysis can be applied to real-valued, continuous data, discrete numeric data, or discrete symbolic data

1.2 Sunspots and Solar cycle

Sunspots are temporary phenomena on the Sun's photosphere that appear as spots darker than the surrounding areas.They are regions of reduced surface temperature caused by concentrations of magnetic field flux that inhibit convection. Sunspots are darker, cooler areas on the surface of the sun in a region called the photosphere. The photosphere has a temperature of 5,800 degrees Kelvin. Sunspots have temperatures of about 3,800 degrees K. They look dark only in comparison with the brighter and hotter regions of the photosphere around them. Sunspots usually appear in pairs of opposite magnetic polarity. Their number varies according to the approximately 11-year solar cycle.

2 Data Exploration

Sunspots are one of the most well documented phenomenon. Over centuries scientists collected number of sunspots, due to which we have monthly sunspot data starting from 1700's to present day. To carry

out the time series forecasting we obtain the data from kaggle which contains monthly mean total sunspot number from 1749-01-31 to 2021-01-31.
The given csv file was converted to panda dataframe and the date column is converted to date time index. The data set has 3265 rows and 3 columns.The monthly mean total sunspot number are in float data type. The date column is given in string format which was converted to date time index further. There are no null values in the data set. Given below are first five rows of the data set.This data set is based on independent observations.

	Unnamed: 0	Date	Monthly Mean Total Sunspot Number
0	0	1749-01-31	96.7
1	1	1749-02-28	104.3
2	2	1749-03-31	116.7
3	3	1749-04-30	92.8
4	4	1749-05-31	141.7

Figure 1: First five rows of sunspot data set

2.1 Time series decomposition

Time Series Decomposition is a technique to extract multiple types of variation from the dataset. There are three important components in the temporal data of a time series: seasonality, trend, and noise.
Seasonality is a recurring movement that is present in your time series variable. Trend can be a long-term upward or downward pattern. Noise is the part of the variability in a time series that can neither be explained by seasonality nor by a trend.
Given below is the plot of distribution of monthly number of sunspots over time. Here, row index were used instead of date for convenience.

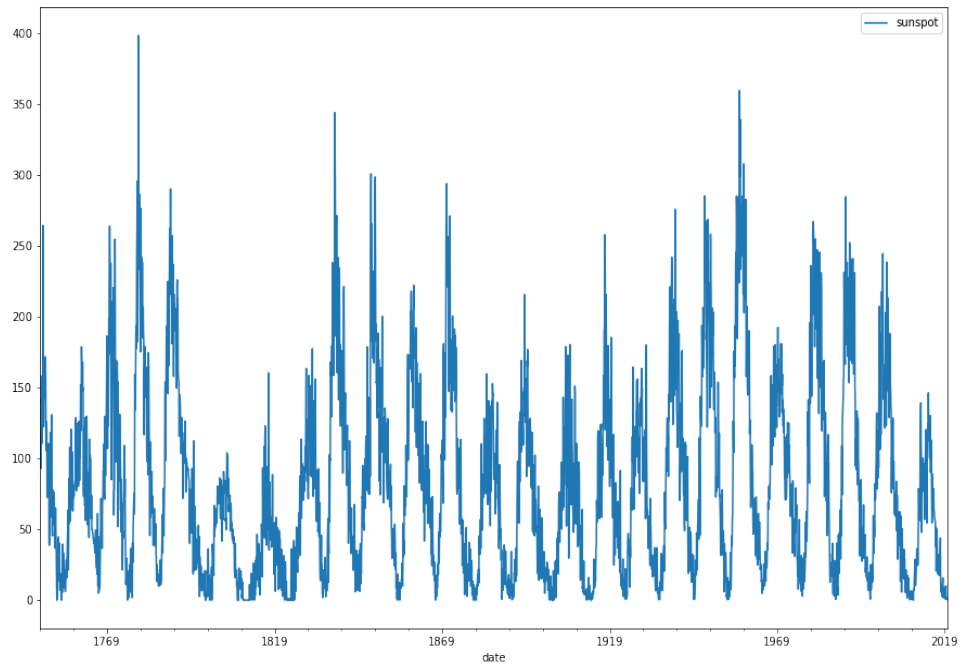


Figure 2: Distribution of number of sunspots

Decomposition is performed using statsmodels' seasonal_decompose function. The plot below doesn't show any trend in particular and a strong seasonality. So, our data doesn't need detrending.

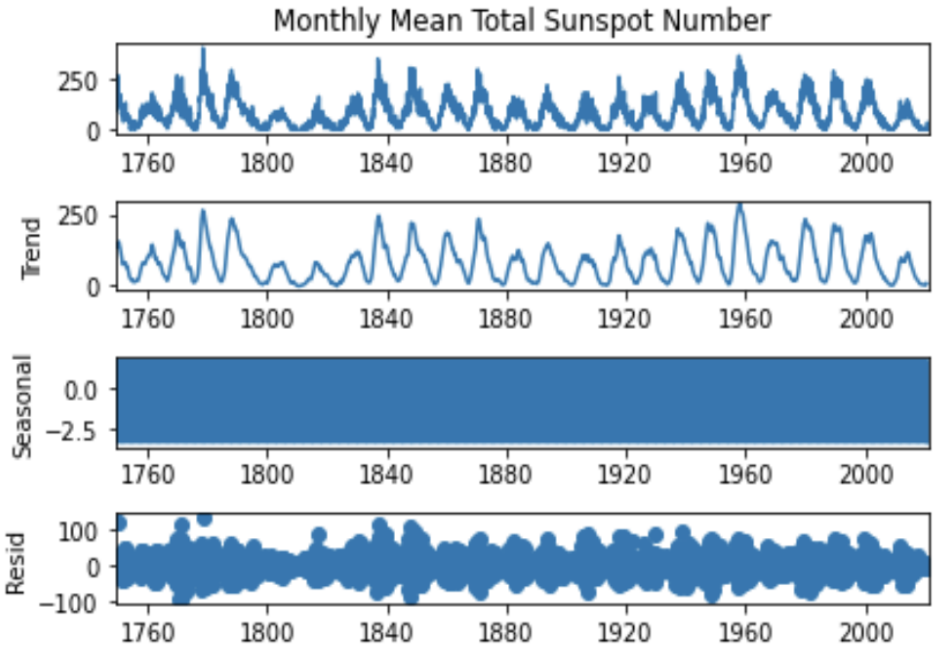
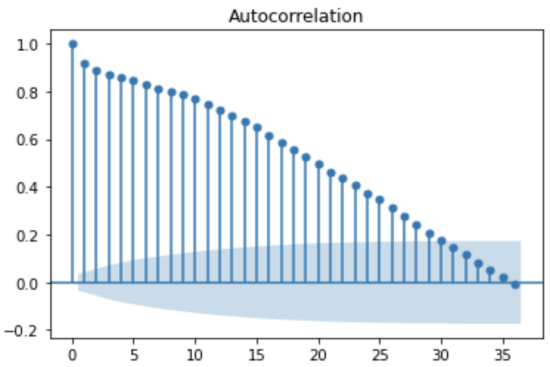


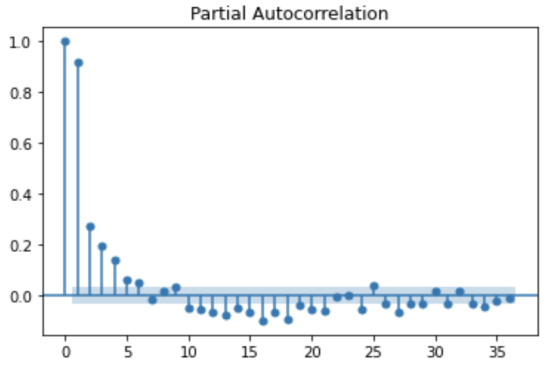
Figure 3: Time series decomposition

2.2 Autocorrelation

Autocorrelation is the correlation between a time series' current value with past values. If this is the case, you can use present values to better predict future values.
Autocorrelation can be positive or negative. Positive autocorrelation means that a high value now is likely to yield a high value in the future and vice versa. Negative autocorrelation is the opposite: a high value today implies a low value tomorrow and a low-value today implies a high-value tomorrow
Two famous graphs can help you detect autocorrelation in your dataset: the ACF plot and the PACF plot.



(a) ACF



(b) PACF

ACF

The autocorrelation function is a tool that helps identify whether autocorrelation exists in your time series. On the x-axis, you can see the time steps also called the number of lags. On the y-axis you can see the amount of correlation of every time step with 'present' time. Its obvious that there is significant autocorrelation in this plot.There is a strong positive autocorrelation with lag 6: a high value now means that you are very likely to observe a high value in the next step.

PACF

The PACF is an alternative to the ACF. Rather than giving the autocorrelations, it gives you the partial autocorrelation. This autocorrelation is called partial, because with each step back in the past, only additional autocorrelation is listed.

You can see below that this PACF plot gives a much better representation of the autocorrelation in the sunspot data. Since the autocorrelation shown here is partial, you do not see any duplicate effects with earlier lags, making the PACF plot neater and clearer.

2.3 Stationarity

A stationary time series is a time series that has no trend. We can detect non-stationarity using the Dickey-Fuller Test and can remove non-stationarity using differencing. **Augumented Dickey-Fuller test** is a statistical hypothesis test that allows you to detect non-stationarity.

For a Time series to be stationary, its ADF test should have: p-value to be low (according to the null hypothesis) The critical values at 1%,5%,10% confidence intervals should be as close as possible to the Test Statistics

Table 1: ADF test results

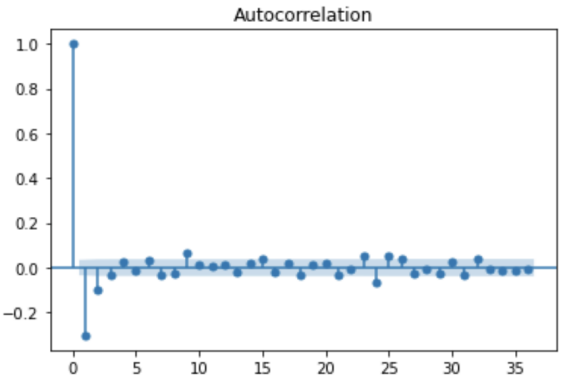
ADF	-10.49705166254614
P-value	1.10855249219565e-18
Lags used	28
Number of observations used	3236
Critical Value 1%	-3.4323724026124003
Critical Value 5%	-2.8624335760905684
Critical Value 10%	-2.5672456699774324

From the above ADF test result, we see that p-value(at max can be 1.0) is less than 0.05, Hence we can discard the null hypothesis and assume time series at the moment is stationary.

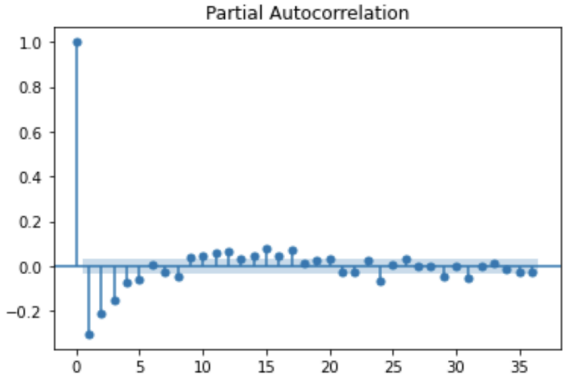
2.4 Differencing

We apply differencing if it data is not stable or to remove the trend from your time series. The goal is to have only seasonal variation: this can be a way to use certain models that work with seasonality but not with trends.

We have to drop null values created while differencing during plotting the ACF and PACF. After



(c) ACF



(d) PACF

redoing ADF test on the differenced data, that this data is now indeed stationary, The p-value is very small, indicating that the alternative hypothesis (stationarity) is true.

From the autocorrelation graph, we can decide if more differencing is needed. If collectively the autocorrelations, or the data point of each lag (in the horizontal axis), are positive for several consecutive lags, more differencing might be needed. Conversely, if more data points are negative, the series is over-differenced.
Clearly, there is no trend and no stationarity so need for differencing the data from AR and MA.

3 Model Training

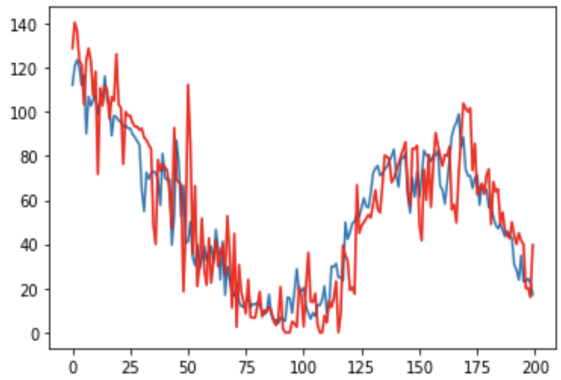
Now that specificities of time series data are analysed, time to look into the types of models that can be used for predicting time series which is generally referred to as forecasting. There are three types of models for time series forecasting :

3.1 Classical time series models

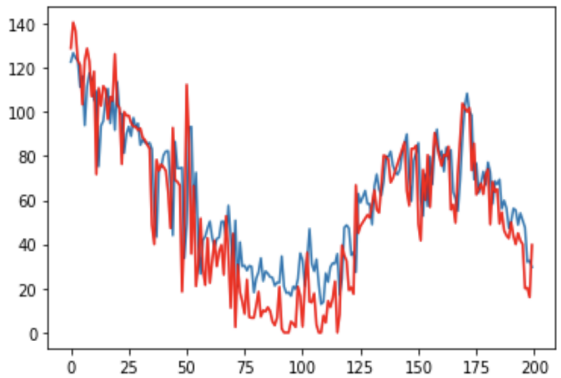
Classical time series models are a family of models that have been traditionally used a lot in many domains of forecasting. They are strongly based on temporal variation inside a time series and they work well with univariate time series. Some advanced options exist to add external variables into the models as well. These models are generally only applicable to time series and are not useful for other types of machine learning. I'll be using AR, MA, ARMA and ARIMA models for this project.

3.1.1 Autoregression (AR)

You can see the AR model as a regression model that explains a variable's future value using its past (lagged) values. The order of an AR model is denoted as p, and it represents the number of lagged values to include in the model.
As, clearly seen from the ACF plot we choose 6 to be the number of lags as they are withing the highlighted region and hence we are using of AR model of order 6. I tried with other values for p, but they gave similar or more error compared to 6. As seen in the plot below which includes a small portion of the whole data, we can say that AR model gives decent results but there is error and it shifted downwards in some parts.



(e) AR(6)



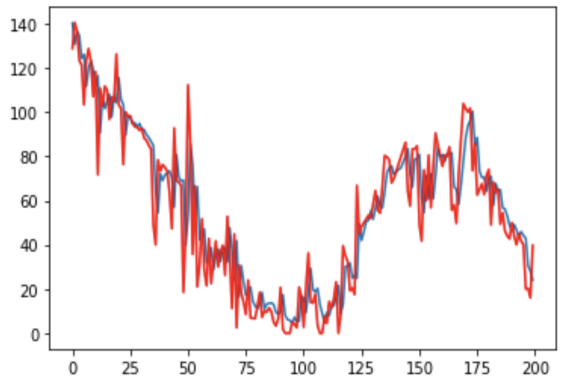
(f) MA(6)

3.1.2 Moving average (MA)

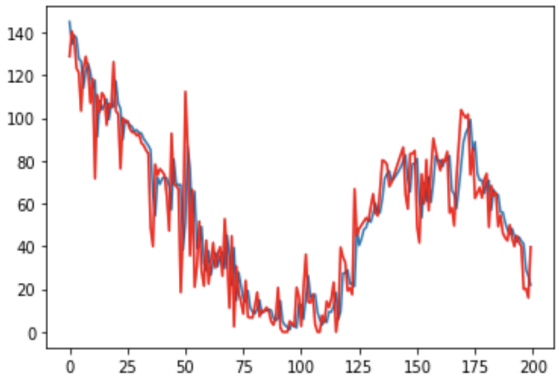
Similar to AR model the Moving Average uses the prediction error in previous time steps to predict the future. When a model has some unknown but regular external perturbations, your model may have a seasonality or other pattern in the error of the model. The MA model is a method to capture this pattern without even having to identify where it comes from. The MA model can use multiple steps back in time as well. This is represented in the order parameter called q. We took q to be 6.

3.1.3 Autoregressive moving average (ARMA)

The Autoregressive Moving Average or ARMA model combines the two previous building blocks into one model. ARMA can therefore use both the value and the prediction errors from the past. ARMA can have different values for the lag of the AR and MA processes. ARMA needs the data to be stationary, but we observed that for this data differenced and normal data gives similar results. So, we take ARMA model with parameters (6,2) after using auto_arma to get the best parameters. ARMA(6,2) gives fairly accurate plot.



(g) ARMA(6,2)



(h) ARIMA(6,1,2)

3.1.4 Autoregressive integrated moving average (ARIMA)

The ARIMA model adds automatic differencing to the ARMA model. It has an additional parameter that you can set to the number of times that the time series needs to be differenced. For example, ARIMA(1, 1, 1), The first 1 is for the AR order, the second one is for the differencing, and the third 1 is for the MA order. ARIMA(1, 0, 1) would be the same as ARMA(1, 1). Differencing is chose at 1 as further differencing isn't giving any better results. ARIMA(6,1,2) gives a little better results than ARMA(6,2) which is clearly evident from the plots.
There are others like Seasonal autoregressive integrated moving-average with exogenous regressors (SARIMAX) and Seasonal autoregressive integrated moving-average (SARIMA) which we are not getting into.

3.2 Supervised models

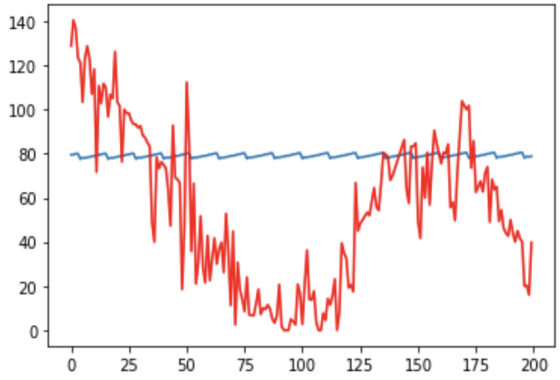
Supervised Machine Learning models consider that variables are either dependent variables or independent variables. Dependent variables, or target variables, are the variables that you want to predict. Independent variables are the variables that help you to predict.

Building a supervised model on time series, you have the disadvantage that you need to do a little bit of feature engineering to extract seasonality into variables in a way or another. There are often no independent variables in time series data. Yet it is fairly simple to adapt them to time series by converting the seasonality (based on your time stamps for example) into independent variables. We converted the date into months and years and used that as independent variables. We are just going ahead with a few regression models.

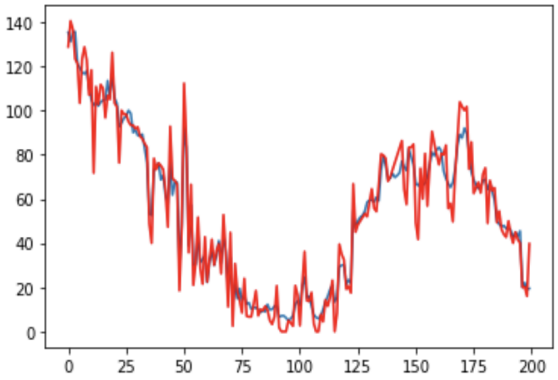
3.2.1 Linear regression

Linear Regression estimates linear relationships: each independent variable has a coefficient that indicates how this variable affects the target variable.

Simple Linear Regression is a Linear Regression in which there is only one independent variable. In Multiple Linear Regression, rather than using only one independent variable, you use multiple



(i) Linear Regression



(j) Random Forest

independent variables. In this case, we built a Linear Model using number of sunspots add as many variables as you need from given data.

When using this code, you will obtain the above plot that shows a terrible fit with the data. Clearly the linear model is very limited and can't work on data that doesn't have a clear linear trend which is the case with our data.

3.2.2 Random forest

Random Forest is a much used model that allows fitting nonlinear relationships, that's why I decided to use it. The scikit-learn library has the RandomForestRegressor that you can simply use to replace the LinearRegression from code block made for linear regression. Clearly, fit on the training data is now even better than before. We understand that this Random Forest has been able to learn the training data better.

3.2.3 XGBoost

Random Forests and XGBoost are considered absolute classics among the supervised machine learning family. XGBoost is an ensemble model of weak learners just like the Random Forest but with an

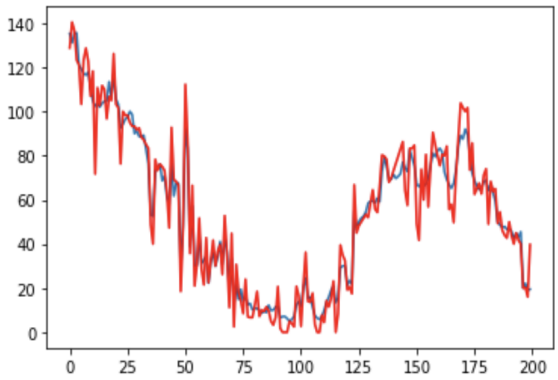


Figure 4: XGBoost

interesting advantage. In standard gradient boosting, the individual trees are fit in sequence and each consecutive decision tree is fit in such a way to minimize the error of previous trees. XGBoost obtains the same result but is still able to do parallel learning.As you can see, this model fits the data quite well too.

3.3 Deep learning and recent models

Classical time series models are focused on relations between the past and the present. Supervised machine learning models are focused on relations between cause and effect. Deep learning based

models are even more complex to apprehend and master and may (or may not) produce better results, depending on the data and the specifics of the use case.
I still don't have great understanding over these topics, but I summed up my understanding of the models below. I haven't used them to make predictions.

3.3.1 LSTM (Long Short-Term Memory)

LSTMs are Recurrent Neural Networks. Neural Networks are very complex machine learning models that pass input data through a network. Each node in the network learns a very simple operation. The neural network consists of many such nodes. The fact that the model can use a large number of simple nodes makes the overall prediction very complex. Neural Networks can therefore fit very complex and nonlinear data sets.

RNNs are a special type of Neural network, in which the network can learn from sequence data. This can be useful for multiple use cases, including understanding time series (which are clearly sequences of values over time), but also text (sentences are sequences of words).

LSTMs are a specific type of RNNs. They have proven useful for time series forecasting on multiple occasions. They require some data and are more complicated to learn than supervised models. Once you master them, they can prove to be very powerful depending on your data and your specific use case. To go into LSTMs we use the Keras library in Python.

3.3.2 Prophet

Prophet is a time series library that was open-sourced by Facebook. It is a black-box model, as it will generate forecasts without much user specification. This can be an advantage, as you can almost automatically generate forecasting models without much knowledge or effort.

On the other hand, there is a risk here as well, if you do not pay close enough attention, you may very well be producing a model that seems good to the automated model building tool, but that in reality does not work well.

3.3.3 DeepAR

DeepAR is another such black-box model developed by Amazon. The functioning deep down is different, but in terms of user experience, it is relatively equal to Prophet.

Again, caution is needed, as you can never just expect any black-box model to be perfectly reliable. We know that the more complex a model, the more wrong it can be! A great and easy-to-use implementation of DeepAR is available in the Gluon package.

The first specificity of time series is that the timestamp that identifies the data has intrinsic meaning. Univariate time series models are forecasting models that use only one variable (the target variable) and its temporal variation to forecast the future. Univariate models are specific to time series.

4 Evaluation criteria

We used Root of Mean Squared Error as a metric for model selection. This metric measures the error at each point in time and takes the square of it. The average of those squared errors is called the Mean Squared Error. An often-used alternative is the Root Mean Squared Error: the square root of the Mean Squared Error.

Table 2:

Model	RMSE
AR(6)	25.042417443963135
MA(6)	46.990222848330845
ARMA(6,2)	25.029602507559396
ARIMA(6,1,2)	25.20270223155756
LR	67.8303717205597
RF	9.194701404233683
XGB	13.698351042406388

5 Results and Observations

We can clearly see based on the evaluation criteria, that Random forest gives the least error followed by XGBoost and ARIMA(6,1,2). This is further validated by visual observation of the plots. So, we select Random forest and use it to predict the future sunspot numbers.
The Linear model is very limited: it can only fit linear relationships. Sometimes this will be enough, but in most cases, it is better to use other models with better performance. AR model plot is shifted a little downard from the actual distribution and MA model plot shifted a little upward in somw regions. ARMA and ARIMA models gave almost similar results with ARIMA giving a little less error. RF and XGB are by far the best performing models, I think it's because they are suitable non-linear data.
I also used tensor flow library, rnn gave pretty decent results but I choose not to include them with others as my understanding of it is limited.

6 Discussions and Conclusion

To improve the performance we could try working with longer or shorter training periods. You could also try adding additional data, like seasonal data (day of the week, month, etc.). we can switch to the SARIMAX model to get better result in classical models. We can apply deep learning methods, use deep neural networks. We manually split the data, to ensure that the models generalise well we can implement train-test split, cross validation or rolling split.

7 References

Data Source : kaggle.

Pushpendu Ghosha, Ariel Neufeldb, Jajati Keshari Sahooc Forecasting directional movements of stock prices for intraday trading using LSTM and random forests. arxiv2004.

Liu, C.D., Wang, J.H., Xiao, D. and Liang, Q. (2016) Fore- casting SP 500 Stock Index Using Statistical Learning Models. Open Journal of Sta- tistics, 6, 1067-1075.

Jerome H. Friedman, Robert Tibshirani, and Trevor Hastie. The Elements of Statistical Learning. Springer, second edition, 2008.

8 Github link

Github link with the report and the jupyter notebook file can be found here. It contains all the code I used for this project. It also contains some extra predictions using tensorflow whose results are not included in the report.