MAC0460 - Introduction to Machine Learning

Tore Fossland - 13797251

July 21, 2022

Universidade
de São Paulo

Abstract

This report discusses the implementation and use of an SARIMA model and an LSTM neural network in order to analyze temperature fluctuations in the world. Specifically, I examine whether climate change is moving so fast that my models are not able to predict the relatively sudden changes. The SARIMA model generally predicts simpler changes than the LSTM, and is not able to adapt to a rapidly changing climate. The LSTM on the other hand is able to predict jagged changes when it's fed highly fluctuating input. I therefore conclude that predicting climate change accurately is a highly difficult task that may well not be possible without considering more data than the global average temperature. Also LSTM seems as good of a model as one could expect to build using one dimensional historical temperature data only.

Contents

1 Introduction

This report is made based on the Term Project of the course *Introduction to Machine Learning* and was taken Spring 2022 at *Universidade de São Paulo*.

1.1 Contributions

Since I joined the telegram group very late I could not find anyone to team up with. Therefore I decided that I had to complete the project on my own. I am writing my master thesis on time series analysis for a Norwegian company next year, so I do not mind putting in some extra work to learn more. However I hope that you take this into consideration when you are evaluating the project, since I think it would be unfair to evaluated on the exact same criteria as other groups with up to 4 members.

1.2 Problem Statement and Data Analysis

The assignment I have chosen is to explore a problem where a public dataset exists, in other words an exploration of type 1 as stated in the problem statement. My goal in this project was to gain insight into how temperature fluctuates over a long time horizon. I would also like to explore if it's possible for the models to predict the uptick in average global temperature caused by the climate change we have had the last 100 years.

Cleaning and preparation of a "fresh" dataset can take an unreasonable amount of time considering the time and credit available for this project. To design the prediction model, I decided to use the *Climate Change: Earth Surface Temperature Data* is affiliated with *Lawrence Berkeley National Laboratory* on Kaggle [1]. It is a public dataset with a CC0 licence and has no copyright issues. The current version combines 1.6 billion temperature reports from 16 pre-existing archives, from 1743 to 2013. It also allows for slicing into interesting subsets (for example by country). The dataset has been upvoted 1802 times on Kaggle, and the publisher has a strong scientific reputation. My analysis of the data confirms this. Normally, when performing machine learning, you need to remove erroneous values, also called outliers. This data set is nearly complete, and I had to do very little removing of outliers.

The problem to explore is to perform time series forecasting on the chosen dataset. I will also try to vary the amount of datapoints used, timeframe and also add external noise to it to gauge the algorithms performance. The data frame will be separated into train and test sets using scikit-learn. I will not touch the test set until the end and use it only once to evaluate the final model. The training set will be exploited in different ways for training and choice of model hyperparameters.

To show the structure of the data, table 1 provides the first data points of the most important columns of the GlobalLandTemperatures dataset.

dt	LandAvgTemp	LandAvgTempUncertainty	LandMaxTemp	LandMinTemp
1850-01-01	0.749	1.105	8.242	-3.206
1850-02-01	3.071	1.275	9.970	-2.291
1850-03-01	4.954	0.955	10.347	-1.905

The LandAvgTemp column is the average temperature that month, LandAvgTempUncertainty is he 95% confidence interval around the average temperature, LandMaxTemp is the global

[1]https://www.kaggle.com/datasets/berkeleyearth/climate-change-earth-surface-temperature-data

average maximum land temperature in celsius and finally LandMinTemp is the global average minimum land temperature in celsius.

1.3 Reasons for choosing topics and data

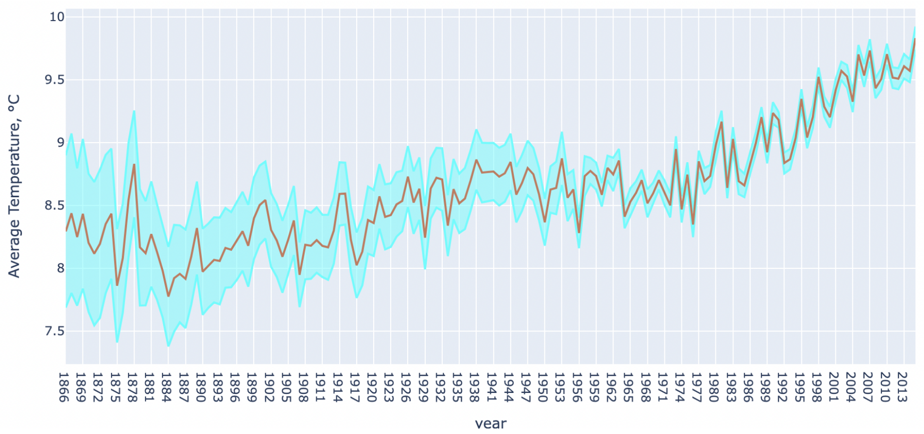
While my degree is in software engineering, I have a lot of interest in the world climate and preservation of our ecosystems. This is a field I would like to apply my technological efforts in later, and therefore a big motivation to choose this dataset.

Predicting climate change has been done for a long time already, but the urgency of the reports have increased in later years. The general nature of the climate change and increase global temperature has been predicted since the early 19th century. The question is if it's possible to accurately predict the future temperature only using the historical temperature timeseries. It is likely to one need to use tons of real-world data such as Co2 concentration in the atmosphere, politics, deforestation etc to have a chance at accurately predicting climate change.

Said in other words, climate is influenced by many factors, and I do not expect to get a very accurate result from only this dataset. But I am still very interested in the topic and really want to deepen my knowledge about how existing approaches work, as well as their performance. In addition, I want to check how accurate predictions I can generate myself.

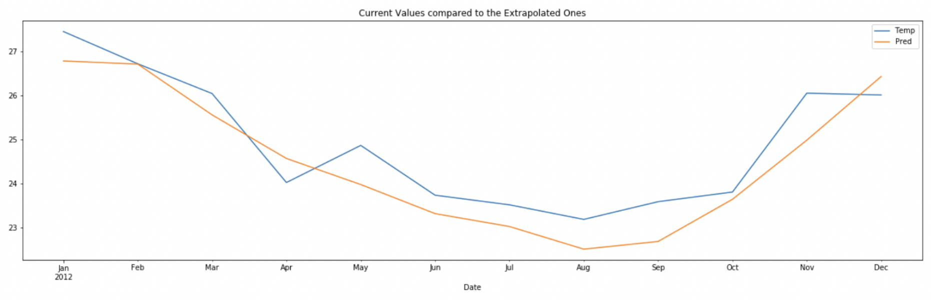
1.4 Existing Approaches, Algorithms, and Perspectives

Most of the previous work I found on Kaggle is plots and simple analysis. Therefore I started by borrowing some of their nice plots and data reformatting. The figure under shows a plot of yearly global temperature with confidence interval from this repository.



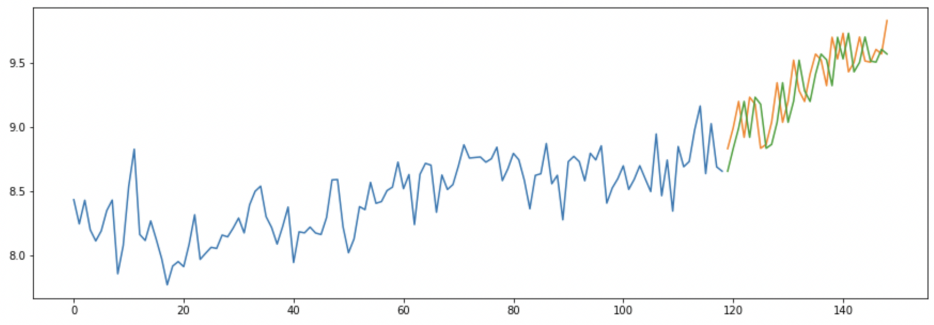
There is also some previous works on linear regression models like ARIMA and SARIMA to predict future temperatures. I will explain more about ARIMA and SARIMA in section 2.1.1. The figure under shows Single-step SARIMA prediction from this repository.

Regarding previous work for LSTM, to my knowledge no one has implemented an LSTM on this dataset. Regarding other datasets on Kaggle the approach is usually to try and predict one day into the future. I found one person who tried to predict 10 days into the future, and reported a worsening in performance. This makes intuitive sense as predicting further into the future implies a greater uncertainty. Furthermore, most attempts on Kaggle made predictions based on fewer than the 60 previous days. Most LSTM implementations perform similarly. They manage to capture the general trend of the values, but appear to be lagging behind a day or two.



To evaluate the performance of the two models there has to be a baseline solution. For time series forecasting this usually comes from the persistence algorithm. The persistence algorithm uses the value at the previous time step (t-1) to predict the expected outcome at the next time step (t+1). This baseline will serve as a benchmark for evaluating the performance of other models. Any model that does not provide better results than a trivial reference model, like the persistence algorithm, should be discarded. I used this implimentation of the algorithm.

The figure under shows the persistence algorithm on the yearly average temperature dataframe.



1.5 Machine Learning Algorithms

Based on both the problem and previous approaches, it seems most appropriate to try a somewhat simple algorithm and a more complex one to see have some variation and to see which one performs best. Specifically, I chose to implement an SARIMA model (Seasonal Auto-Regressive Integrated Moving Average) and a LSTM (Long Short-Term Memory) model. The former is a modified regression model while the latter is type of neural network. I was planning on implementing a SARIMAX model but the workload is already very high for a one person project. Therefore I had to make some adjustments and went for a SARIMA instead. Both models are further explained in section 2.

1.6 Preprocessing

The pre-processing step is an essential step in any sort of machine learning implementation. It consists of all the operations that are needed to be performed on the data before it can be fed into the machine learning model. In order to use the LSTM I scaled all values linearly to values between -1 and 1. That is to say, the largest positive difference between two consecutive numbers gets mapped to 1, the largest negative difference gets mapped to -1, and the other differences get mapped in-between. Here, a positive value means an increase in temperature and a negative value signifies a decrease in temperature.

The orignial plan was to reformat the data into yearly averages, to make the climate change much more visible. Since the SARIMA model does not require a big amount of data points, I used this formatting for that one. However when I tried this for the LSTM it was insufficient, and I therefore had to revert to a monthly average for that one. The results and reasoning will be explained further in the discussion.

Part of the preprocessing is to make modifications to the data to see how the algorithms will handle it. In this project I create three different dataframes for each algorithm. The first one is the original dataset, the second a smaller subset of the original and the third is the original with added noise. The noise is implemented differently for the two models but have the same Gaussian properties in practice.

2 Theory

2.1 SARIMA

The SARIMA (Seasonal Autoregressive integrated moving average) model is a combination of the AR and MA models modeled on a differentiated time series, that also takes seasonality into consideration. Both of these models are used on time series data to get a better understanding of the data, or to predict future points in the series. I will only use the model for the latter purpose.

ARIMA models are applied where the data is not stationary in relation to the mean value. The non-stationarity of the mean function (in other words, the trend) can be eliminated by applying the inital differencing step one or more times. This corresponds to the "integrated" part of the model and the parameter d.

The AR (autoregressive) model, of order p, models the current value as a linear combination of the previous p values. The name, autoregressive, comes from it being a linear regression of itself. The MA (Moving Average) model, models the current value as a linear combination of the previous q error terms. In other words, it checks how much the prediction were wrong in the previous q values and corrects for this.

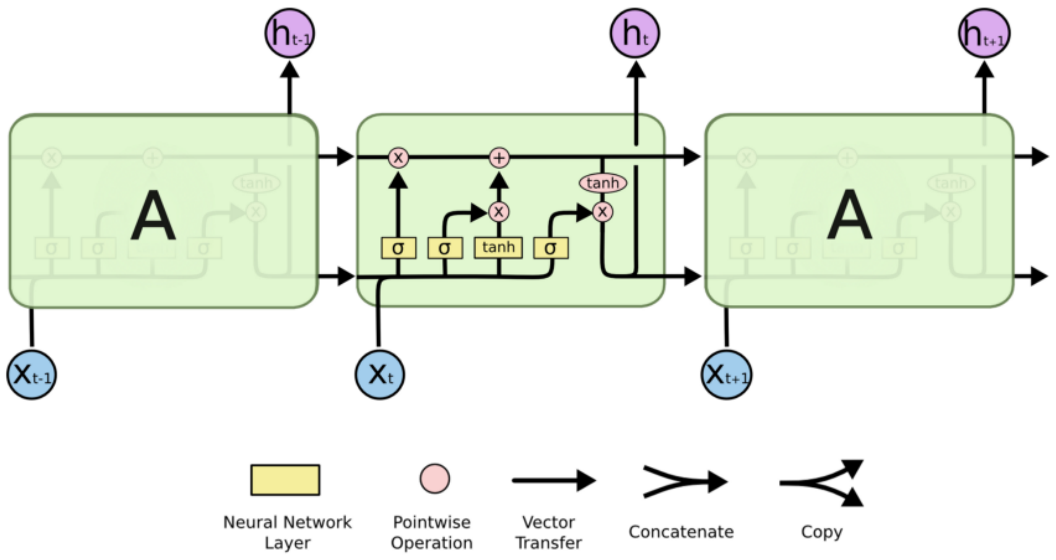
2.1.1 AUTO ARIMA

To automatically determine model parameters I wanted an automated process instead of analyzing things like The Augmented Dickey-Fuller Test and autocorrelation plots. This is why I use the auto-ARIMA function. The function uses a step-by-step procedure with the Hyndman-Khandakar algorithm. The algorithm iterates through all the different permutations of models one can have, up to a maximum value of p and q. The function also finds the minimum value for d to make the data set stationary. It then selects the best model choice according to AIC, and returns the values of p, d and q respectively. The AIC (Akaike Information Critera) is a widely used measure of a statistical model. Given a collection of models for the data, AIC estimates the quality of each model, relative to each of the other models. Thus, AIC provides a means for model selection.

2.2 Long Short-Term Memory (LSTM)

A Long Short-Term Memory (or LSTM for short) network is a type of recurrent neural network that is as the name suggests able to predict based on long sequences of data while also incorporating short term trends. The LSTM rose to prominence due to solving the problem of vanishing gradients which is apparent in many RNNs. Its use cases include problems like language processing, where predicting the next word in a sequence is based both on what came immediately before as well as the context of the conversation. In other words, information from longer past sequences. Visually, an LSTM can be represented by the workflow shown in fig. 9.

In fig. 9, x_t and h_t is the input and output at time step t. The upper arrow going horizontally through the blocks represents the LSTM's internal state, which can be thought of as hidden. A single cell can roughly be broken into 3 parts, going from left to right; the forget layer, the input layer, and the output layer. The forget layer consists of the leftmost sigmoid function and a pointwise multiplication, and decides what information to disregard in each time step.



The input layer consists of the sigmoid and tanh functions in the middle, and is responsible for deciding what states to update (signified by the sigma function) and how much (signified by the tanh function). This is then added to the internal cell state through pointwise addition. Finally, the LSTM needs to decide what to output, and this is done through the rightmost sigmoid and tanh functions. The tanh function simply squeezes all internal states to values between -1 and 1, while the sigmoid function and pointwise multiplication decide on the individual weighting on each internal state.

In this report, I use a multi-step LSTM. This simply means that instead of inputting one data point at each time-step and predicting the next data point, I input n data points and predict m data points into the future. In my case, I input data points from the past n months' average temperature and use it to predict m months in the future.

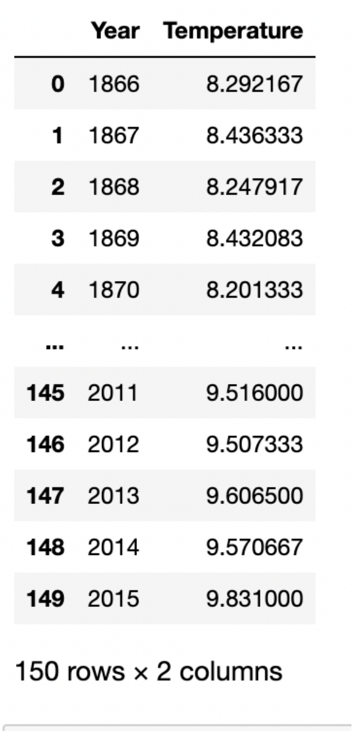
3 Implementation

3.1 SARIMA

Using Jupyter Notebook, I have to download the dataset locally to use it. After loading the dataset into the Jupyter Notebook I choose a column, in this case global average temperature, to analyze.

First a range of temperature is fetched from the dataset with Pandas, as well as a function to parse the date-time field. The data table is then narrowed to only contain the average temperature for each month or year.

Running the corresponding code prints the data table, showing the first and last data entries.



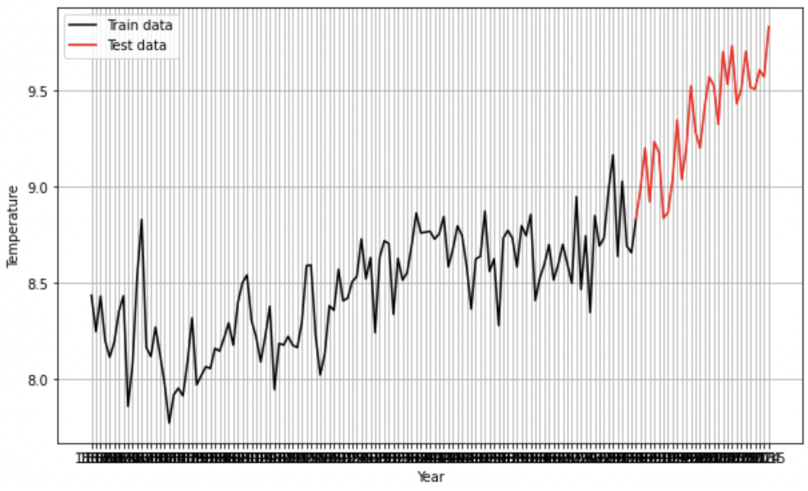
	Year	Temperature
0	1866	8.292167
1	1867	8.436333
2	1868	8.247917
3	1869	8.432083
4	1870	8.201333
...	...	...
145	2011	9.516000
146	2012	9.507333
147	2013	9.606500
148	2014	9.570667
149	2015	9.831000

150 rows × 2 columns

Before I start the analysis however, I have to split the data. The data set is be split into two parts, one training set and one testing set. The training set will be used to train the SARIMA model. The testing set, on the other hand, will be used to determine how accurate my SARIMA prediction is. This split is done according to a split ratio. In my case the split is 80% for training and 20% for testing. The training data is what I am going to base my analysis on.

Running the corresponding code prints the plotted data, showing the training data in black and the test data in red.

From the plot one can see that the time series has a clear trend. This means that it is not stationary and will require differencing of at least a magnitude of 1.



Instead of deciding the model parameters manually I can handle them automatically by using the function auto ARIMA described in the theory section. Analysing graphs manually every time I input a new parameter, data range or add noise is not feasible.

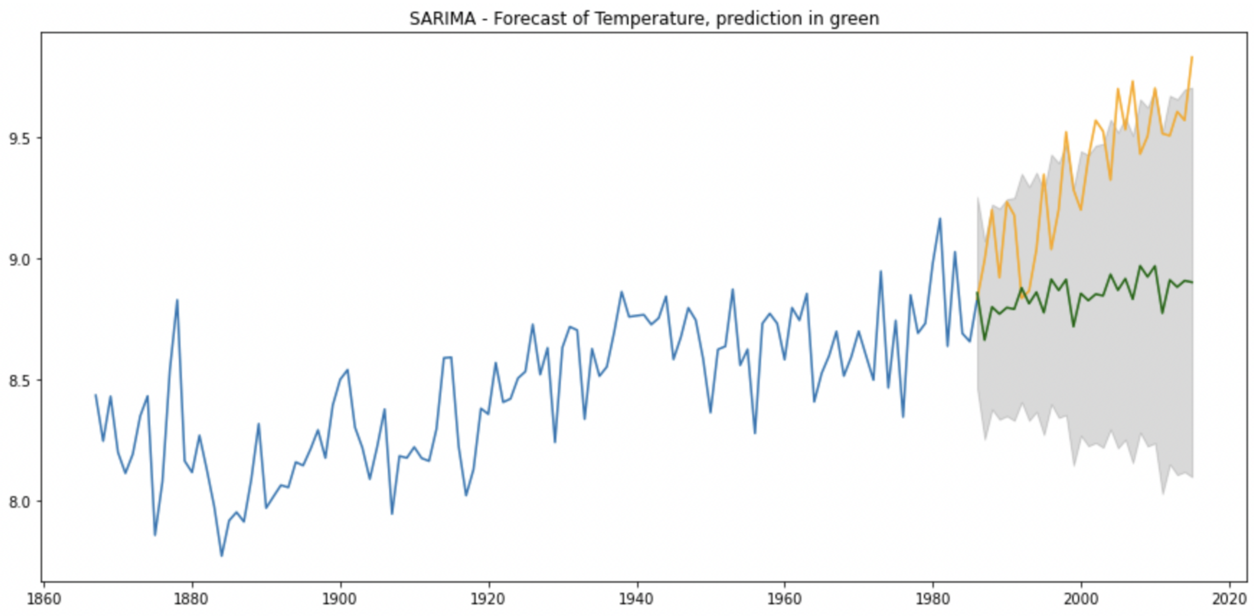
```
Performing stepwise search to minimize aic
 ARIMA(1,1,1)(0,1,1)[12]             : AIC=-20.185, Time=0.24 sec
 ARIMA(0,1,0)(0,1,0)[12]             : AIC=37.368, Time=0.02 sec
 ARIMA(1,1,0)(1,1,0)[12]             : AIC=14.990, Time=0.05 sec
 ARIMA(0,1,1)(0,1,1)[12]             : AIC=-20.770, Time=0.11 sec
 ARIMA(0,1,1)(0,1,0)[12]             : AIC=9.381, Time=0.01 sec
 ARIMA(0,1,1)(1,1,1)[12]             : AIC=-18.800, Time=0.16 sec
 ARIMA(0,1,1)(0,1,2)[12]             : AIC=-18.806, Time=0.78 sec
 ARIMA(0,1,1)(1,1,0)[12]             : AIC=-5.948, Time=0.05 sec
 ARIMA(0,1,1)(1,1,2)[12]             : AIC=inf, Time=0.88 sec
 ARIMA(0,1,0)(0,1,1)[12]             : AIC=14.227, Time=0.06 sec
 ARIMA(0,1,2)(0,1,1)[12]             : AIC=-20.240, Time=0.15 sec
 ARIMA(1,1,0)(0,1,1)[12]             : AIC=-1.309, Time=0.11 sec
 ARIMA(1,1,2)(0,1,1)[12]             : AIC=-18.250, Time=0.27 sec
 ARIMA(0,1,1)(0,1,1)[12] intercept   : AIC=-19.129, Time=0.17 sec

Best model:  ARIMA(0,1,1)(0,1,1)[12]
Total fit time: 3.086 seconds
```

Running the Auto ARIMA function outputs the best ARIMA model with parameters, according to the AIC minimizing discussed in the theory section. Notice that the latter part of the model output is for seasonality, so even though it says it's an ARIMA model it's actually a SARIMA.

The last part of the implementation builds the SARIMA model with the parameters I have found through my analysis. Running the corresponding code graphs my training data, actual data and the prediction forecast with a 95% confidence interval. This is done with the original, subset and noise dataframes respectively.

Lastly I have performance measures to review my predictions numerically.



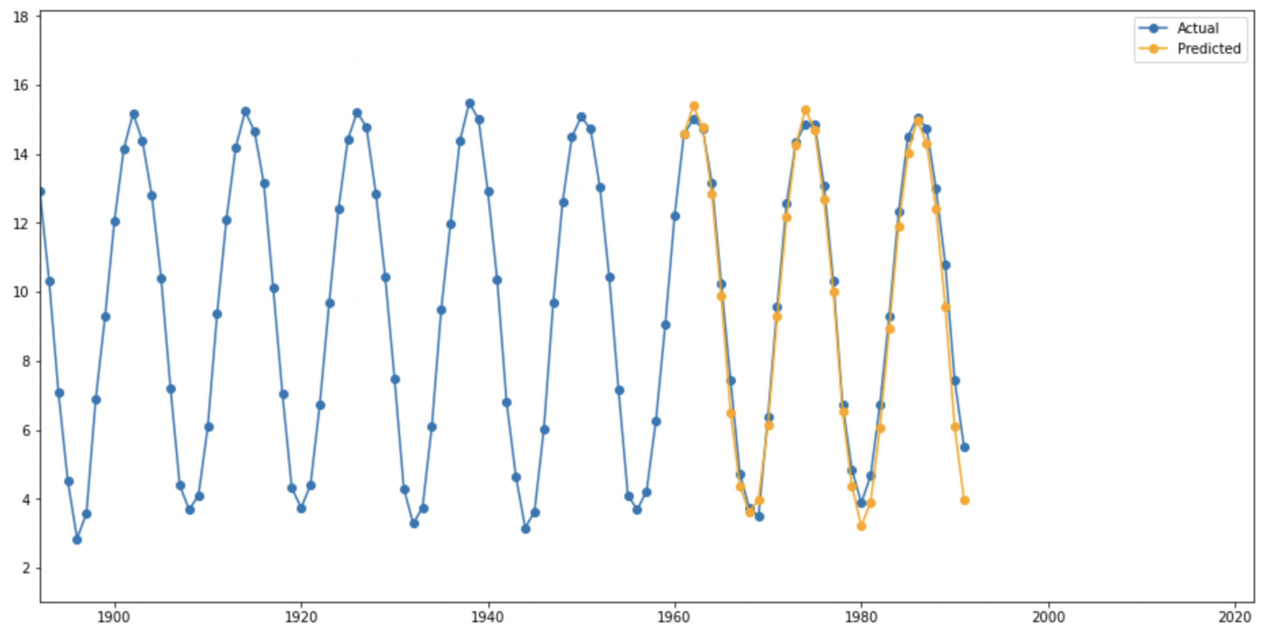
The python implementation is shown in it's entirety in **??**.

3.2 Long Short-Term Memory (LSTM)

I initially implemented the LSTM model in Jupyter Notebook using Keras, Pandas, and a few other python libraries. Jupyter Notebook made it much easier to create a working prototype and to implement/test the model in chunks. Once finished, I exported the code to a normal python script.

My implementation in python is fairly straightforward. First, the data is loaded and split into a training set and a test set. Note that only one feature is used at a time, e.g. only the average monthly temperature. Then, the data is scaled to numbers between -1 and 1, which concludes the pre-processing of the data.

At this point, the LSTM neural network itself is created using keras. This is done by fully connecting a LSTM layer (with a selectable amount of neurons) to an output layer (with size equal to the amount of time-steps I wish to predict). Note that I could have added more layers or more neurons in each layer, but according to other reports and my own quick trials, this only seemed to extend running time without any better predictions. In addition, the amount of data isn't nearly enough to justify creating a large and complex neural net. Once created, the model is then trained over a given number of epochs to fit the training data. After this, the model is ready to start predicting future temperatures. Note that since the data has been transformed, the model will yield predictions that need to be inverse transformed before making intuitive sense. This is done by scaling all numbers up and taking cumulative sums.



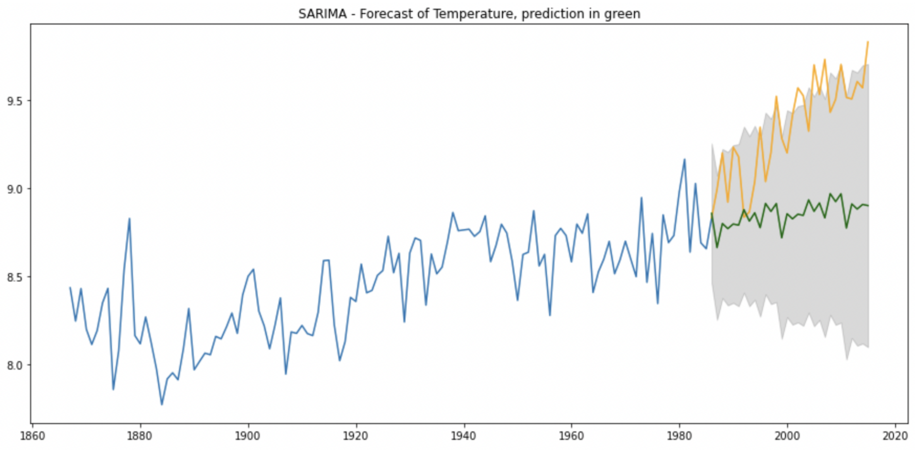
The python implementation is shown in it's entirety in **??**.

4 Results and discussion

4.1 SARIMA

4.1.1 Prediction result

The prediction states how the SARIMA model performs on the global average temperature. The model was trained on the previous 120 years of data and predicted the following 30 years of temperature development. The global temperature has had a consistent rising pattern in the given time horizon. The parameters for the SARIMA model is given by the auto ARIMA function.

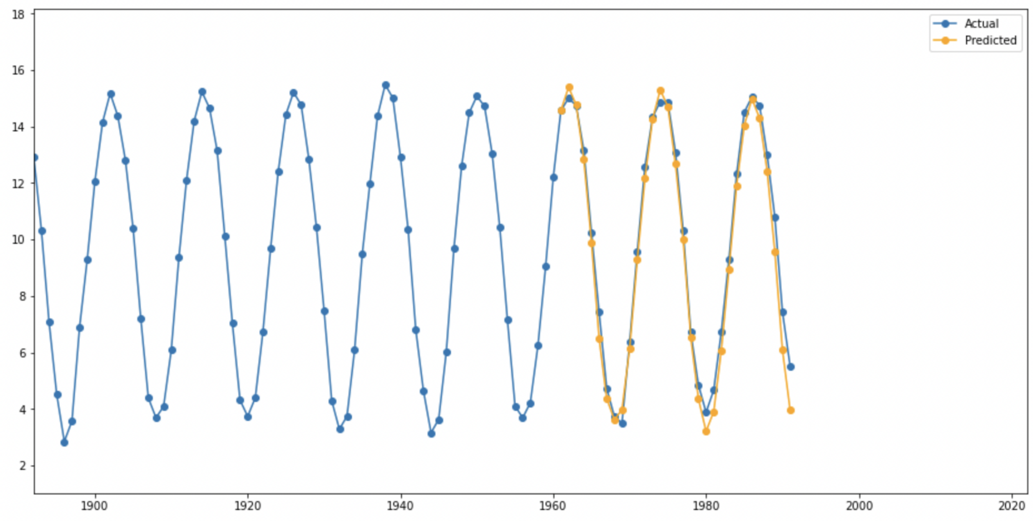


Plot fig. 10 shows the SARIMA(0,1,1)(0,1,1)[12] model on the global average temperature. The model seems to give a prediction that is reasonable based on earlier years temperatures. However because of the last few decades accelerating global warming it is not responsive enough to take this into account. The actual observed values is barely within the 95% confidence bounds.

The predictions shows that the SARIMA model works well if your data does not have sudden strong developments in a novel direction. The global average temperature unfortunately does contain a strong upwards development on the set horizon. The SARIMA model is therefore not able to deliver a prediction inside my confidence interval. Sudden strong developments are usually a result of factors independent of the historical global temperature. The SARIMA model will therefore never be able to predict this based on the historical temperature alone.

4.2 Long Short-Term Memory (LSTM)

Plot fig. 11 shows how my LSTM model performs on a the global average temperature with a prediction horizon of 30 months. The global average temperature has 1992 data points. The model was trained with a rolling window of 120 measurements, with 30 neurons over 20 epochs. Although my model misses a few individual data points, it's predictions align extremely well with the general trend of the temperature. The fact that it misses individual data points is as expected. Temperature can fluctuate in a seemingly random manner, but the general trend is preserved. If real world events were taken into account, one could perhaps have an even better chance at predicting future temperature.



4.3 Choice of final model

To show the permance of the different algorithms and its permutations, table 2 provides the loss function results for each model permutation.

Model	RMSE	MSLE	MAE
Baseline, SARIMA	0.2144	0.0004	0.1893
SARIMA, original	0.5503	0.0029	0.48941
SARIMA, subset	0.6982	0.0048	0.6439
SARIMA, noise	0.8134	0.0067	0.7228
Baseline, LSTM	0.2.166	0.0.0681	1.9213
LSTM, original	0.5244	0.0055	0.3966
LSTM, subset	0.6461	0.0087	0.5106
LSTM, noise	0.6047	0.0068	0.4899

Even though the two models aren't trained on the exact same data, looking at the plots one can see that LSTM is more suitable for data series with sudden sudden strong developments in a novel direction. This argument is further solidified when I compare the values of the loss functions between the algorithms and the baseline solution. SARIMA performs worse than the persistence algorithm on all three of my performance evaluation criterias, making it not useful. LSTM on the other hand performs better on all the criterias, even with a smaller part of the dataset for training, or with added noise.

5 Conclusion

In this report I have used two distinct models in order to try to predict how the global average temperature will develop in the future. In my experiments, I found that the simpler SARIMA generally predicts simpler changes than the LSTM. While the SARIMA model might predict a almost liner rise or fall over a period of time, the LSTM is able to predict jagged changes when it's fed highly fluctuating input. Both models performs worse when trained or smaller datasets or after having noise added to the dataset. However they both still perform reasonably well compared to their original performance.

Both of these models are able to predict the general trend in the data, but not necessarily individual points. Although the parameters could be further tuned and more data could be examined, it seems clear that the root cause of incorrect predictions is factors the algorithms don't have access to. The global climate is notoriously difficult to predict. Going into this project, this was clear to me, but nonetheless I wanted to try for myself. In the end, I conclude that the SARIMA model is not able to predict future climate change to any significant or useful degree. This could be different if I added more exogenous variables with a SARIMAX approach. The LSTM however is much quicker at picking up changes and can adapt to rapid changes in global temperature, making it as good of a model as one could expect using only historical temperature data.