DETECTION AND PREDICTION OF CLIMATE CHANGE WITH TEMPERATURE

UNIVERSITY OF EXETER

Submitted by Adnan Ahmed to the University of Exeteras a dissertation for the degree of Master of Science in September 2020

Table of Contents

1 Introduction

'Weather' is a blanket term under which there are many components to consider. Be it the temperature, humidity, rainfall, air pressure, wind speed, wind direction, and a multitude of factors that play a role in the overall atmospheric conditions of a region, taking this into consideration; it is a safe assumption to say that it is improbable that the weather of a particular region is the same two days in a row. This makes it challenging to predict the conditions one might face when actively participating in an activity. It could be argued that the weather of a region is one of the essential features of the place's development in terms of infrastructure and the foundations of society. For example, if a site is prone to extremes in weather conditions that make it inhabitable by most communities, then obviously, the prosperity of that community living there is also severely affected. Weather is something that has an influence on all living beings daily and cannot be avoided or ignored under any circumstances. Thus, the underlying motive of weather forecasting could be considered one of the ways we can ensure the safety, evolution, and growth of the human species.

Weather forecasting is the science of predicting the weather of a particular region with the help of statistical principles and models that are based on significant and historical data about a place. This and the principles of physics that we know influence the weather combined are the foundational principles of weather forecasting. In most prediction models in weather forecasting, we must leave some room for error as it is not feasible to assume an accurate prediction with no flaws, primarily because of the numerous factors that play a role in the weather. The data that we use tends to have significant factors, but there are multiple factors that could be directly correlated to the weather, such as geographical location, wind speed, moisture, air pressure, seasonality, and a wide range of subsequent features. Now there is a spectrum of variables that indirectly affect the weather, such as calamities like earthquakes, landslides, tsunamis, and other rare occurrences or phenomena that happen so rarely that they may be considered outliers and not considered whilst doing our analysis and predictions, which leads to less accurate results.

Science and technology have made a substantial improvement in this field, and so what we can expect is nothing short of sufficient in terms of predictions. Climate change is a phenomenon that is in direct correlation to weather and may be considered as the periodic modification of Earth's climate brought about as a result of changes in the atmosphere as well as interactions between the atmosphere and various other geologic, chemical, biological, and geographic factors within the Earth system (Jackson, S. T., 2017). Thus, the main aim of this report is to accurately interpret and predict the change in the average temperature of various regions in the world to determine a trend and estimate if the temperature of the planet is increasing or decreasing over time. This is significant and plays a vital role in the well-being of not just the human species but all the living beings on the planet. Ecosystems around the world, both marine as well as land-dwelling creatures as well as plants and microorganisms, live in a delicate balance with each other, and a tiny change in this ecosystem could cause detrimental damage to it. Thus, climate change growing at a rapid pace is dangerous and has already led to the extinction of multiple species that have been recorded.

It is hard to say the damage it has caused as we do know the possible undiscovered species and environments that have been damaged or destroyed. One this is for sure, and that is that climate change growing at such a rapid pace does not have any positive impact on our planet, and many studies have shown how climate change can directly lead to natural calamities such as the melting of the polar caps in the north pole which leads to the increase in the water levels of the planet which subsequently has led to many communities being displaced from their homes as the land has been captured by the ocean. This is just one of the infinite use cases that prove that climate change is real, and we use real-life data to come to our own conclusions as well. We use the average temperature of different cities across the world from the time period 1995 to 2019 and asses to see if there is an increase in the overall temperature or not; we do this using the average temperature of the place. This is provided to us by the dataset source, which calculates the average temperature using normalised functions.

We implement various statistical data modelling techniques, such as time series analysis, long-term short memory, which is an application of artificial intelligence, and a few other models, after which we pass them through a few agnostic models that are specifically designed to test the interpretability of each model and check to see which gives us the best results. The dissertation may be broken down as follows. Chapter 2 consists of a literature review and all the ways, methods, and techniques that have been used in the past to determine the weather forecasting and climate change calculations, followed by Chapter 3, which includes a description of the data we use as well as the methodologies implemented by us to determine our results and finally in chapter 4 we discuss the results and the conclusion we have come to and the significance it beholds.

2 Literature review

As discussed in the introduction section, we can see that weather forecasting is essential not just for convenience but rather for the well-being and safety of a region. Climate change can be accurately measured with respect to changes in the temperature of a region over a period. There are various statistical techniques that have been implemented to determine the change in temperature as well as predict it. In this literature review, we are going to delve deep into the ways that it has been done in the past. We may also compare to see the most common methods used and their effectiveness. Using these techniques, we are able to compare and see the interpretability of each model as well as the statistical significance and accuracy they tend to produce. Some of the methodologies used are as follows.

2.1 General Analysis and Pre-processing

This is usually the first stage of most projects, where we analyse the data to search for missing values, outliers, and any trends that may occur over a period. In the case where we use variables such as temperature, snowfall in cms, and amount of rainfall, which are numerical, we usually clean the data by either eliminating the missing values if they are insignificant, or we may also use the mean or median of the column to fill in the missing data from the previous and next day. There are various libraries in R and Python which are designed to combat this, such as MICE, tidyr, Pandas, and many more. As the temperature is primarily the variable that we used to determine this climate change, we see that in the analysis of climate change in Switzerland by M. Beniston et al. The evolution of daily minimum temperatures at the four stations from the beginning of the century to the end of 1992. Based on the daily temperature values, mean annual statistics have been established; a five-year running mean has been applied in order to filter out some of the high-frequency modes inherent to the inter-annual variability. (Beniston, M., Rebetez, M., Giorgi, F. , 1994). The temperature has a pattern that shows that it has increased by 2K over the year it has been observed.

This, along with several other variables individually, is what the analysis was conducted on in this case. Similarly, we see that M.S Shekhar et al., in the paper Climate-change studies in the western Himalayas used seasonal temperature over a significant period of time where they noted that the maximum temperature, as well as the minimum of the region, has increased by 2.8 degrees Celsius and 1 degree Celsius respectively. This, in turn, leads them to believe there will be anomalies in the other climatic conditions of the region, such as the snowfall in the region, and after careful analysis, they were able to see a significantly less amount of snowfall over the same period. It was observed that there was around 280 cm of snowfall less than the previous years. (Shekhar, M. S et al. (2010)). Another critical feature of general analysis and preprocessing is to determine if two or more variables are correlated with each other in any way and try to spot any trends if possible.

There are different ways that the correlation can be measured, such as Pearson's Correlation coefficient, which is used to quantify the linear relationship between two variables that are random in nature. (W. Xie, M. He et al., '2020') uses Pearson's correlation coefficient as well as Spearman's Rank coefficient to understand the type of relationship that exists between wildfire and drought severity, we let the random variable X denote the number of forest wildfires, and the random variable Y represents the Palmer Modified Drought Index (PMDI) which can be used to measure the drought severity. Using the formulas of each correlation coefficient, they observed that Pearson's correlation coefficient is 0.712. Spearman's correlation coefficient has a numerical value of -0. 714. These numerical results indicate that these two variables are highly correlated in this case study. The major drawback, in this case, is that sometimes we may find that the variables are not correlated with each other, which is why we need to conduct this analysis multiple times with different variables taken into consideration. The other major setback is that correlation does not necessarily mean causation. This means that even if they are very much correlated, we cannot assume that they are dependent on each other in any way. Thus, further analysis must be done, and this can be considered the initial step of analysis in climate change forecasting but definitely not the final step of the process.

2.2 Time Series Analysis (ARIMA)

Models Time series analysis, to its name, is a statistical modelling technique that is used for analysis and prediction over a period of time. From the term climate change, we can see that change is a process that occurs over a period of time. Thus, this technique would seem apt for climate analysis. According to (Kaufmann, R.K. et al., '2016'), the evidence of the effect of human activity on the climate is mainly evident from two sources: The experiments run by climate models and also the statistical analysis of historical data. In general, there are various ways to analyse and predict data over a period of time, but the most common way to go about it is using the principle of Regression; the three most popular models that are used as AR, MA, and ARIMA models. These represent Auto-Regression, Moving Average, and Autoregressive Integrated Moving Average. These are of different orders and have different kappa values that are used to measure which is the best-fitted model for a particular dataset with respect to the time series. The Auto Arima function is used to determine which is a good model to fit, after which we can use the appropriate model to generate predictions. (Dmritri, T. Ahmad, S. et al. '2020') used the precipitation and the maximum as well as minimum temperature from the years 1901-2000 by monthly means to generate a time series dataset and conduct the analysis. They also fit a separate SARIMA model on the precipitation and temperature time series. They then did the following steps to make sure that they had the best model to fit the time series data to find an accurate and reliable prediction.

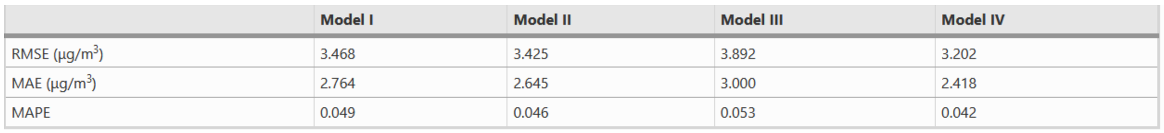
Initially, the first and foremost step was to determine which order of the series was best suited to make the series stationary. Different ARIMA models were fitted with different orders but had a constant coefficient. Now that the differenced series exists, it is considered to be stationary, although it may still have auto-correlated errors. The next step undertaken was to identify the AR(p) and MR(q) components. This can be done by generating an Autocorrelation function (ACF) as well as a Partial Auto Correlation function (PACF) which can show us how well the present value of the series is related to that of the past values. By seeing if there is a sharp cutoff of the differenced series on the PACF graph, we can observe that AR needs to be added to the model, and if the same occurs on the ACF graph, we know that MA needs to be added to the model.

Following this, they have made some estimations using appropriate p,d, and q values which are fitted to the ARIMA models with appropriate residuals. Then the seasonality is removed for the models, and the best SARIMA model needs to be selected. After which, the forecasting is done, and the results are tested under the Akaike Information Criterion, which is an estimate of a constant and the relative distance between the unknown true likelihood and function of the data fitted on the model. Thus we can see that the lower the AIC value, the closer it is to the truth. Lower values indicate that the less information the model loses, the higher the quality of the model. The Bayesian Information Criterion (BIC) is also a criterion used with the same principle of the lower value, the better, and is based on the likelihood function. Using this method, they were able to observe the maximum and minimum temperature of the regions, respectively but also were able to forecast predictions for the precipitation, which showed a constant trend in values.

2.3 LSTM

Air Long term short memory is a type of Recurrent Neural Network which has a long-term dependency which means that it can retain information for a long period of time due to the internal memory it possesses. It is diagrammatically represented in figure 2.2, as shown below. As we can see, the mechanics revolve around gates. Primarily these gates are comprised of gates, which are structures through which information is added or removed in the cell state; and also has weights Wj, where j signifies the state name. It contains sigmoid neural, σ, net layer, and a pointwise multiplication operation. The sigmoid layer is called a forget gate denoted by ft. This element decides what information will be thrown away from the cell state. Its decision is made by looking at xt and ht−1. It consists of two property values; one is the hidden state H(t) which is primarily responsible for the long-term memory, and the forget gate F(t) adjusts the connection of the input with that of the previous hidden state to the cell state, which then allows it to forget when needed (Pak, U. et al. 2018) In the prediction of ozone concentration uses LSTM and CNN models to determine the concentration by implementing the following methods.

In the project undertaken by Pak, U. et al., they combine CNN and LTSM network in different ways to create four models which has a simple combination of the convolution layer and the layer of LSTM in different combinations, such that Model I is a simple combination of the convolutional layer with one layer of LSTM. Model II is a combination of the complete convolutional and pooling layers with one layer of LSTM. Model III is a combination of the convolutional layer with two layers of LSTM. Model IV is a combination of the complete convolutional and pooling layers with two layers of LSTM. They then run all four models with respect to RMSE, MAE, and MAPE, and we find that Model IV has the lowest values amongst the four, and thus we can figure that it is the best fit model for predicting the ozone concentration. As shown in the table below, we can see the values of Model 4 is much better than the rest on all accounts.



	Model I	Model II	Model III	Model IV
RMSE (μg/m^3)	3.468	3.425	3.892	3.202
MAE (μg/m^3)	2.764	2.645	3.000	2.418
MAPE	0.049	0.046	0.053	0.042

As we can see, the mechanics revolve around gates. Primarily these gates are comprised of gates, which are structures through which information is added or removed in the cell state; and also has weights Wj, where j signifies the state name. It contains sigmoid neural, σ, net layer, and a pointwise multiplication operation. The sigmoid layer is called a forget gate denoted by ft. This element decides what information will be thrown away from the cell state. Its decision is made by looking at xt and ht−1.

It consists of two property values; one is the hidden state H(t) which is primarily responsible for the long-term memory, and the forget gate F(t) adjusts the connection of the input with that of the previous hidden state to the cell state, which then allows it to forget when needed. (Pak, U. et all '2018) The prediction of ozone concentration uses LSTM and CNN models to determine the concentration by implementing the following methods. In the project undertaken by Pak, U. et al., they combine CNN and LTSM network in different ways to create four models which has a simple combination of the convolution layer and the layer of LSTM in different combinations, such that Model I is a simple combination of the convolutional layer with one layer of LSTM.

Model II is a combination of the complete convolutional and pooling layers with one layer of

LSTM. Model III is a combination of the convolutional layer with two layers of LSTM. Model IV is a combination of the complete convolutional and pooling layers with two layers of LSTM. They then run all four models with respect to RMSE, MAE, and MAPE, and we find that Model IV has the lowest values amongst the four, and thus we can figure that it is the best fit model for predicting the ozone concentration. As shown in the table below, we can see the values of Model 4 is much better than the rest on all accounts.

2.4 ANN

Agent There are three types of neural networks: recurrent neural networks, multilayer neural networks, and single-layer neural networks. Feed-forward networks with a single layer: The information only moves forward in the network, where the neurons are stacked in layers. The output layers of computational neurons are referred to as the single layers.

Multilayer Feed-forward network: This is different from one layer in that it has one or more hidden layers. Since these layers do not directly communicate with each other, they are referred to as "hidden" in the network's perimeter because their values are not tracked during training.

Recurrent network: Its existence of feedback loops sets it apart from feed-forward neural networks. An example of a recurrent neural network would be neurons that send back their output signal to the input.

Artificial Neural Networks are based on the human brain. In an ANN, the nodes are mostly structured in layers. These layers come in three different types: input, hidden, and output. They are designed to take in a collection of inputs, carry out intricate calculations, and provide an output. The weight of each synapse makes up the ANN's collection of synapses. These weights (wjm) specify how the input will affect a neuron. An adder (additive junction) is a component of the ANN that contributes to adding the weighted signals.
A selection criterion may be imposed by the adder based on the architecture. Among these conditions are minimum, maximum, average, and so forth (Kubat, M., 1999). The basic

structural element of ANN is called perceptron, and the transfer function for neuron m is given by yj = ϕ(vj) = ϕ Xm i=1 wjixj + bj ! with vj = z = Xm i=1 wjixj + bj where {x1, x2, .., xm ∈ R} represents the inputs, w1m, ..., wjm ∈ R are the respective weights of the m neuron. bj ∈ R is the bias that has the effect of increasing or decreasing the net input of the activation function. There exist different kinds of activation functions in machine learning, namely sigmoid, hyperbolic tangent (tanh), and rectified linear unit (ReLU) (Kubat, M., 1999). This is diagrammatically represented below in figure 2.1.

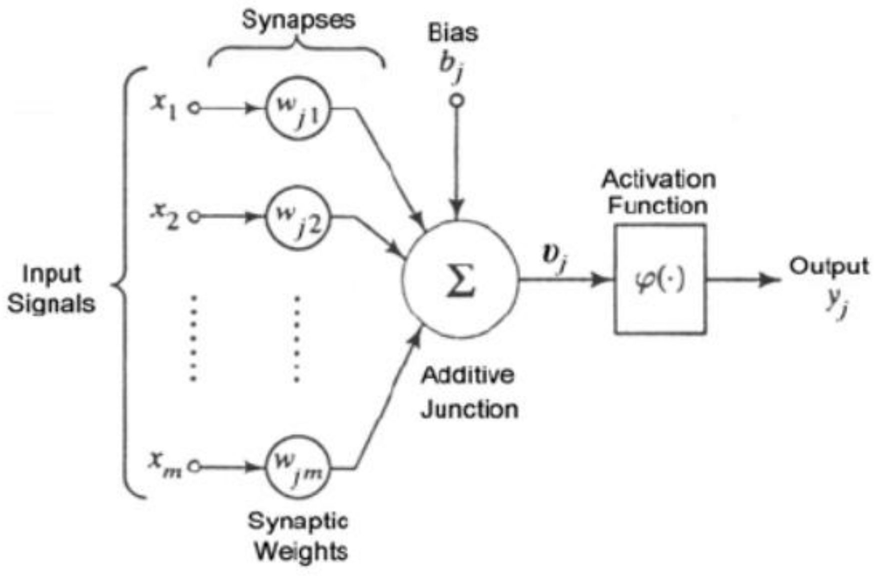


Figure 2.4.1 Artificial Neural Network Mechanics (Kubat, M., 1999)

(Chithra, N.R., Thampi, et al. '2015') used ANN-based models were developed for obtaining projections of monthly mean maximum and minimum temperatures at the station scale. The capability of the model was assessed by applying it to the Chaliyar river basin in Kerala, India. In the case of the prediction of T max, data pertaining to the predictors and the prediction were divided into three seasons, viz. dry period (January–May), wet period (June–November), and the month of December. Although in the case of the prediction of T min, they divided it into two seasons (wet and dry), and the networks were split and trained separately for each season. They used the correlation coefficient between the predictors and the predicted values. They found that the ANN model is feasible to downscale the climate data and generate appropriate results, which in this case was they found both the maximum as well as the minimum value have increased. The main drawback of this model and approach is that there is a certain level of uncertainty that may be present in the result. This could be combatted using ensemble models.

2.5 RNN

A Recurrent Neural Network is an improvement or development over a traditional Artificial Neural Network and mainly consists of a grouping of sequence networks. Usually, gradient errors are a drawback in these models, but this can be rectified with the help of LSTM models as well as the new and upgraded RNN systems. The structural components are shown in figure 2.5.1

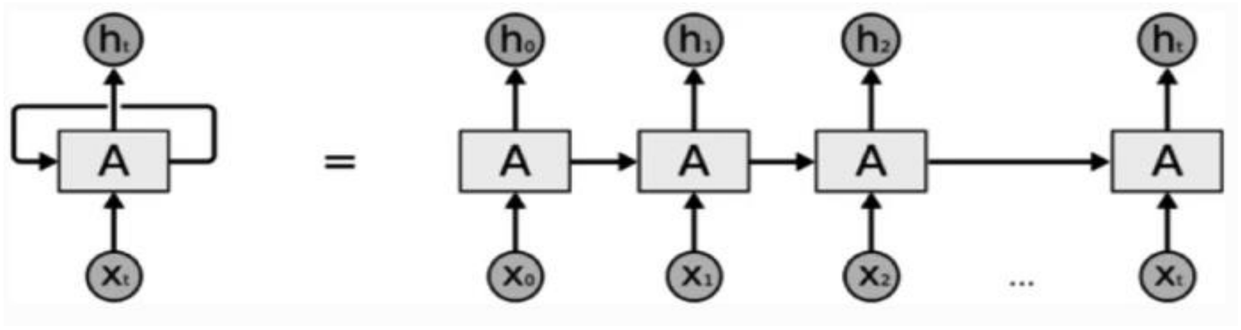


Figure 2.5.1 Represents the structure of unrolled RNN (Joy, D.T., et al.' 2022')

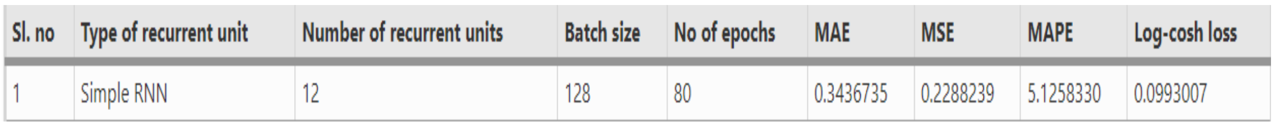
There are four types of RNN's such as One to One, One to Many, Many to One, and Many to Many. They operate on two widely used principles known as feed-forward neural networks as well as backpropagation.

2.5.1 Backpropagation

In the governing equations of RNN, the parameters A, B, and C are learned through backpropagation given by ∂L ∂A = X t ∂L ∂ht ∂ht ∂A, (2.6.4) ∂L ∂B = X t ∂L ∂ht ∂ht ∂B, (2.6.5) ∂L ∂C = X t ∂L ∂yt ∂yt ∂C. (2.6.6)

The optimization is solved iteratively through gradient descent given by w j+1 = w j − η∇wL|w=wj (2.6.7) where j = 0, 1, 2..., and η is the learning rate. The learning in RNN is highly influenced by the value of this parameter. When the parameter is too small, it leads to the vanishing of the gradient, hence preventing the network from converging. On the other hand, the network diverges when the parameter is too large (Rumelhart, D. et al. '1986).

(Sadhukhan, B et al '2022') Used a simple RNN model that consisted of twelve recurrent units in the input layer. These units transmit the output to the first hidden layer after processing the sequences. The first hidden layer is a dense layer with ReLU activation. The next layer is again a dense layer with linear activation. For optimisation, RMSProp has been used. This RMSprop optimiser finds its way into many neural network applications. They found that after training this model for 80 epochs and for over 128 batches and is found that they have a constant loss value. Since the objective of the project is to find a correlation between the number of earthquakes that occur in a region and the corresponding global temperature values in the RNN model, they begin by splitting the dataset into the train and test sub-datasets; this is traditionally done by splitting 80% of the total dataset in the training stage and the rest 20% in the testing stage. Now that this has been accomplished, they proceeded by setting certain metrics to measure the performance of the model. The four metrics used are MAE, MSE, MAPE and log-cosh loss; as we can see in figure 2.4, the values of each.



Sl. no	Type of recurrent unit	Number of recurrent units	Batch size	No of epochs	MAE	MSE	MAPE	Log-cosh loss
1	Simple RNN	12	128	80	0.3436735	0.2288239	5.1258330	0.0993007

Figure 2.5.1.1 shows the values of the RNN model

The convergence of the values has proved that the bias and variance of the training set are low or negligible. The low values of MSE and MAE indicate that the model has fit the dataset, indicating a relationship between the two variables. The highly similar values from the training set indicate a strong correlation between earthquake magnitudes and global temperature fluctuations.

2.6 CNN

Convolutional Neural Network is a deep learning algorithm that is mainly used in a pattern Recognition is made up of four layers: convolutional (conv), nonlinear, pooling (pool), and fully connected layers. Different features are learned as the image moves from one layer to the other in a channel. (Lawrence, S. et al. '1997'). In figure 2.6, we can see the graphical representation of the different layers in a typical CNN architecture.

We can see that there are four main types of layers that work in perfect sync to form a convoluted neural network; these layers all perform a specific function, as detailed in the following,

Convolutional layer: It is the first layer in CNN, and it is where convolution with the kernel or filter. This layer usually learns features such as lines. The filter used must always have the same number of channels as the input.

Nonlinear layer: This layer introduces nonlinearity in the output from the first layer. This is done by adjusting or cutting off the output using a nonlinear function such as ReLU or tanh.

Pooling layer: It conducts dimensionality reduction by reducing the number of parameters through a process called down-sampling. There exist two types of pooling: Max and average pooling. Max returns the maximum value from the region covered by the filter, while average returns the average value.

Fully connected layer: This layer is responsible for connecting outputs from other layers to the main output.

Baño-Medina, J. et al. '2021' have used Convoluted Neural Networks to downscale the temperature and precipitation all across Europe. They created and fitted models that are more traditional statistical techniques like basic linear regression models with the intent to compare the results of these models with that of the CNN model. The major advantage they found was that CNN circumvents the problem of feature selection/extraction—which is highly case-dependent and becomes a very complex task to accomplish in classical downscaling methods—by performing an implicit manipulation of the input space in the internal structure of the network.

The model consists of an input layer that is directly connected to the feeding layer, and the final convolution is directly connected to the output layer, which uses linear or sigmoidal transformations to compute the required results. In this approach, they used a wide range of atmospheric predictors such as temperature, and amount of rainfall.. For temperature, they minimise the MSE, which is equivalent to minimising the negative log-likelihood of a Gaussian distribution for the conditional mean. The final results we see are that the CNN largely preserves the climate change signal given by the global model for all the indices considered, posing no challenges to the interpretation of the patterns obtained. A similar situation is found for temperature, for which CNN preserves to a great extent the global model's climate change signal both for the mean and extremes. The final conclusion that the project has come to is that CNN plays an important role in the statistical downscaling of systematic errors and thus largely increases the accuracy and readability of the models for interpretation. Hence they concluded that CNNs are best for the generation of reliable climate change information on continental-sized domains, which is crucial for the implementation of adequate mitigation policies.

3 Data and Methodology

The aim of Chapter 3 is to explore the dataset, understand all the latent and visible relationships and model on top of the data. This chapter will detail out the process of building out the models, along with the distribution of variables, missing value analysis, imputation of missing values, outlier analysis, univariate analysis and probability distribution. This analysis will help us understand the data. Post this; the focus will be on modelling the data to see what future trends will look like; we will use the baseline as well as more advanced methods of time-series analysis for the same. Interpretation of the results is key, and we will summarise our learnings to bring a holistic end to Chapter 3.

3.1 Introduction

The primary objective of this project is to determine and predict the change in the surface temperature of the Earth. We use a felicitous dataset that has been obtained from the University of Dayton in the United States of America. Based on our learnings from the Literature Review performed in Chapter 2, we will work to build on top of the work done by authors of other papers. We will use the baseline as well as advanced techniques leveraging a wide variety of models to ensure that the dataset exploration, analysis, modelling and interpretation are carried out in a holistic manner.

3.2 Dataset Description

The dataset that we will be using is from the University of Dayton, which has data from the year 1995 up to 2019 and is regularly updated by the university. The National Climatic Data Center's Global Summary of the Day (GSOD) database serves as the files' primary source of information (NCDC). The Global Summary of the Day (GSOD) data's 24-hourly temperature observations is used to calculate the daily average temperatures. The dataset consists of eight columns, namely,

- **Region**: Information indicating the region from where the weather data is gathered
- **Country**: Indicative of the country from where the data has been gathered
- **State**: Indicative of the State from where the data has been gathered
- **City**: Indicative of the City from where the data has been gathered
- **Month**: This column indicates the month in which the data has been collected
- **Day**: This column indicates the day on which the data has been collected
- **Year**: This column indicates the ear in which the data has been collected
- **Average Temperature**: This data has been computed as a function of the average of 24 hourly readings from the Global Summary of the Day (GSOD) data

We have the data of over 324 cities which consist of 157 cities in the US and 167 cities from across the world. Since we are looking at data from 1995, we have 2,885,424 rows of the information under these columns, this is a particularly large dataset, and thus we filter and sort out different regions and cities to conduct our analysis as an effective way to measure the average temperature and determine if it is changing over a period of time or not.

We see that the data that denotes the respective area in which we use as the basis of our analysis is found to be in the object format, and the month, day, and year are integers where the average temperature is in float format as it has decimal values. There are very few missing values that can be easily rectified by using the mean of the previous and next day's temperatures, as it is an accurate assumption to make that the missing value is not too varied from the previous and next day in a season. There are a few outliers that were easily determined with the help of basic boxplots. Although the data is not perfect, and there may be a few discrepancies, we are able to clean it to the point where the dataset is suitable to carry out multiple different statistical calculations using machine learning and artificial intelligence algorithms to make predictions and forecasts.

3.3 Exploratory Data Analysis

In this section, we will explore the data from the National Climatic Data Center. We will analyse the distribution of the variables, perform missing value analysis, necessary transformations, feature engineering, outlier analysis and also look at the probability distribution using KDE. The data is 769 MB and has 2.88 million rows. We perform the preliminary preprocessing, such as typecasting the Month, Day and Year to integer and typecasting the average temperature to float to ensure that there is no discrepancy. In the next section, let us go over the steps of preprocessing that has been carried out.

3.3.1 Preprocessing of the dataset

We start off by removing the ambiguous values in the data frame that will not add value to the analysis. In the year column, we have values such as 200 and 201 that do not make sense as it is not a valid year. We will also drop the State column, as it is specific to the United States of America, as it does not add value to the analysis and increases computation time. We will deal with different levels of aggregation for various visualisations in order to analyse the data that we will discuss in the proceeding sections.

3.3.2 Missing Value Analysis

We look at the percentage of missing values by grouping the data by country. Certain countries such as Burundi and Guyana have missing value percentages such as 80.5% and 57.8%, respectively. These countries will not add value to the dataset; In general, we learn that there is missing data from the countries of Africa, Latin America and Asian countries. In the data dictionary, it is specified that missing values are indicated as -99, so we convert them to NaN (Not a Number), which is a NumPy data type to work better with the data. As a sum of all of the missing values, there are 79,164 data points missing in the given data. We try to keep as much of the data as possible by performing an imputation of the missing values with the mean temperature of the same city and date from all the years. This exercise, in turn, brings the missing values to 25,627 data points. Post this; we analyse the missing values by the city to see what the feasible cities are; we can analyse them in detail.

The rest of the missing values are imputed with the average overall mean temperature of the city. Now, the number of missing values is zero.

3.3.3 Distribution of Variables

It is critical that we analyse the distribution of the variables in a variety of permutations and combinations so that we have a good intuition of the data. In this sub-section, we will analyse what the distribution of the data looks like in this case. First, we will look at the distribution of the yearly global average temperature in Figure 3.3.3.1. Note that the data present here is in degrees centigrade.

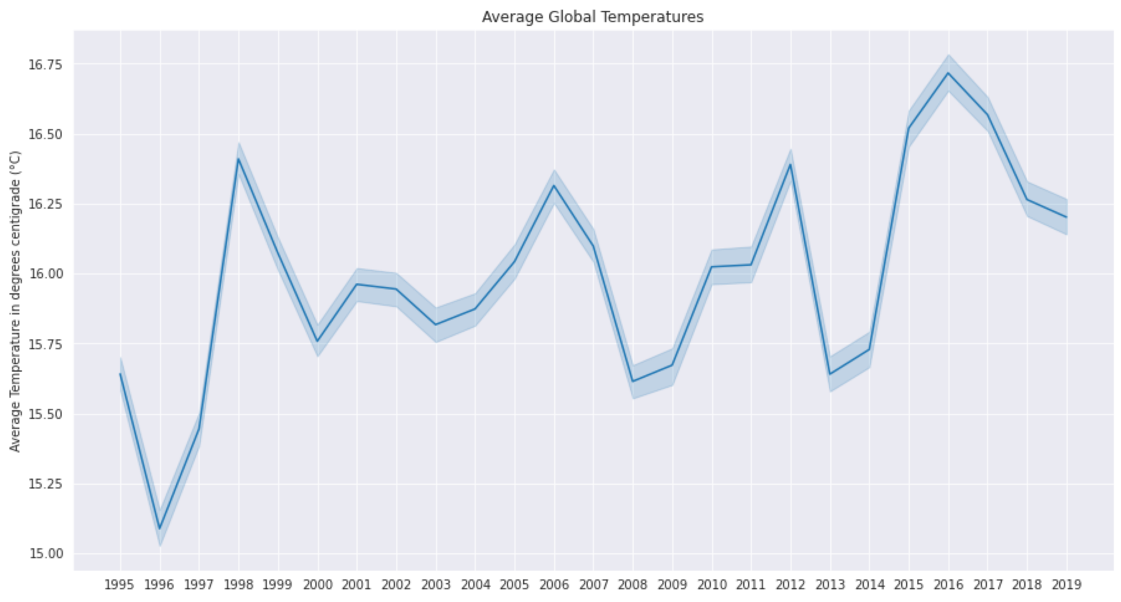


Figure 3.3.3.1: Distribution of average global temperatures by the year

Here, we see a varying trend across the years, but if we were to extrapolate the data, we would see an increasing trend in the global average temperature. To further support our case study, we use the plotly express library to plot the temperatures via an interactive graph from 1995 to 2019. Let us now have a look at the extreme instances of the graphs.

Here, in Figure 3.3.3.2, we have plotted the average temperature in the year 1995 on a graph of the globe. In Figure 3.3.3.3, we will now plot the average temperature in the year 2019. If our interpretation of Figure 3.3.3.1 is correct, we should see an increasing trend in global temperatures.

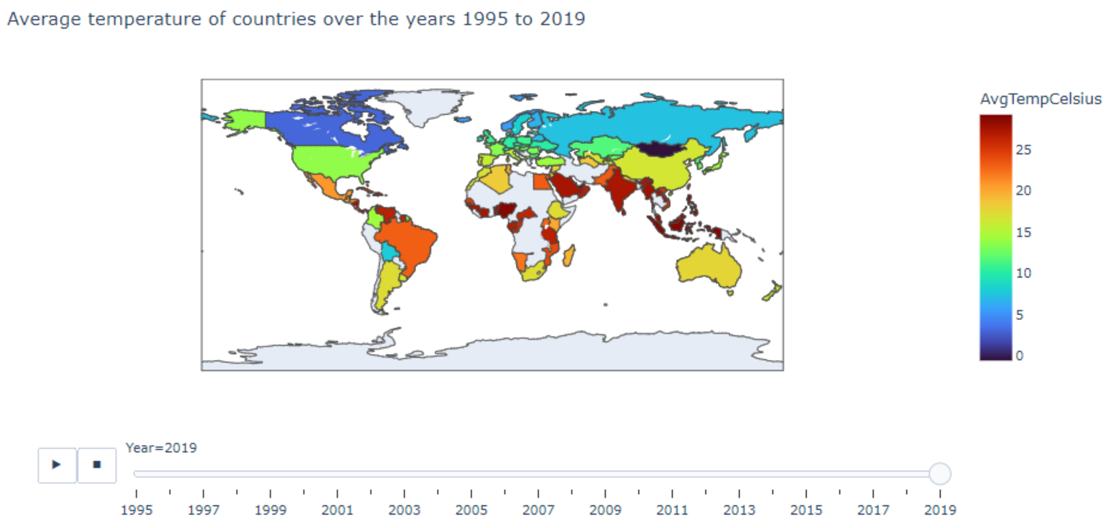


Figure 3.3.3.3: Plot of Average Temperature in the Year 2019 by Country

A lot of the temperatures that were in the range of 10-15 degrees centigrade have now moved to the 15-20 degrees centigrade, and the hotter countries have now gotten even hotter. This indicates that our preliminary analysis was as expected. This can further be corroborated if we plot the average global temperature by month and year.

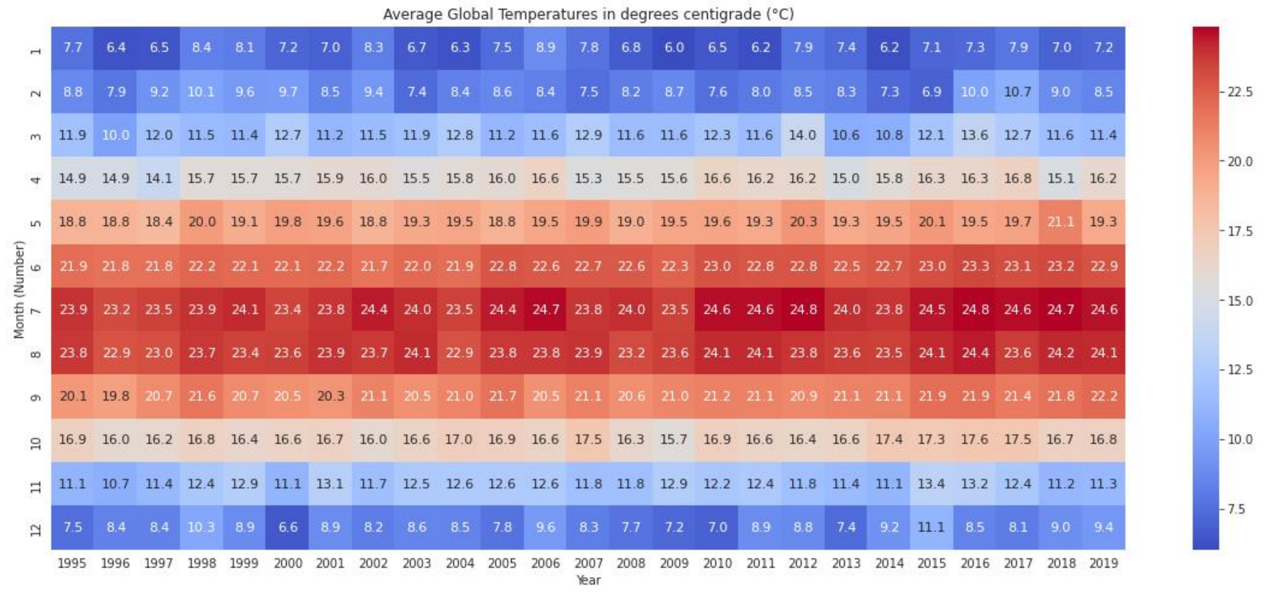


Figure 3.3.3.4: Plot of the Global Average Temperature by Month and Year

As we can see in Figure 3.3.3.4, the global average temperature has a gradual increasing trend in some months. Specifically, in month nine, September, we see that the global average temperature has moved up from 20.1 degrees centigrade to 22.2 degrees centigrade.

Each degree increase in the global average temperature leads to the melting of ice caps, which leads to an increase in sea level. The phenomenon of global warming is further corroborated by the data that has been collected. To further our understanding of the distribution of data by region, we can look at the average temperature across various regions. This will help us observe and learn more about the temperature trends across the globe. We interpret that the hottest regions are Africa and the Middle East, followed by the South/Central America and Caribbean region. From Figure 3.3.3.4, we also note that the coldest regions, as expected, are Europe and North America.

3.3.4 Outlier Analysis

In this section, we will look at the distribution of average temperature. We will leverage box plots and plot the distribution of temperatures from 1995 to 2019 by region. Looking at global temperatures, on average, the temperature is expected to be less than 50 °C and more than -40 °C. Box plots give us an indication if there are outliers in the data. We will retain all of the points as they are in the acceptable range, and these are real-world temperatures. It will not make sense to remove any rows, as we would like to perform a time series analysis.

From the plot in Figure 3.3.4.1, we can see that North America and Asia have a wide range of average temperatures. Now, box plots give us an indication of the median temperature, which is shown by the line in the middle of the box. However, box plots do not give us a complete indication of the density or distribution of the points. We can understand more about the density and distribution by looking at the violin plot.

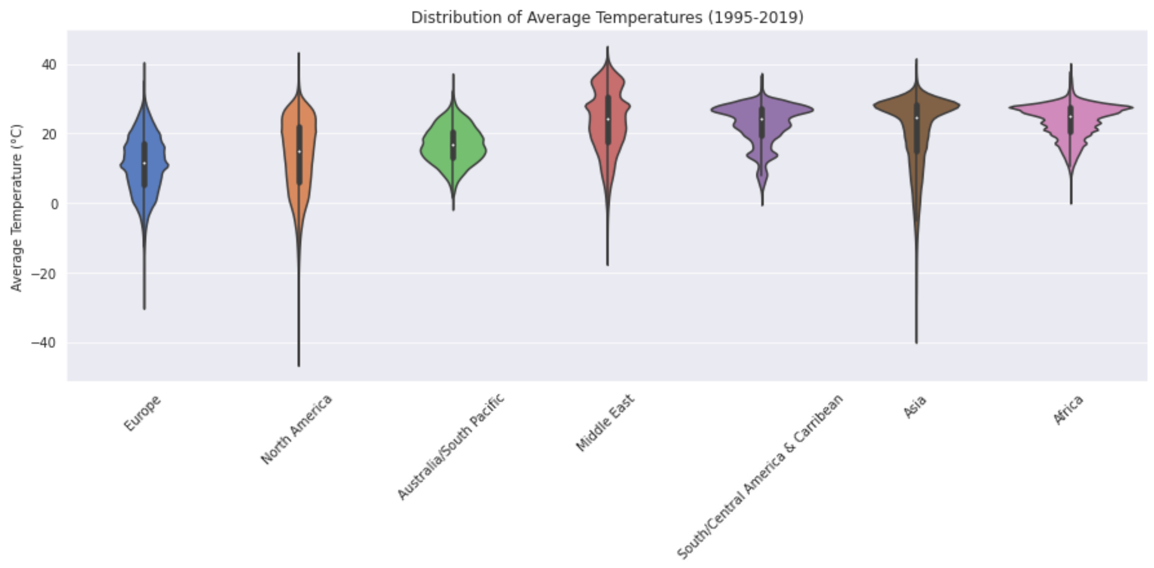


Figure 3.3.4.2: Violin plot of average temperature from 1995 to 2019

From the violin plot in Figure 3.3.4.2, we can see the density distribution of the temperatures. For instance, Asia has a high number of countries that range between 25°C and 30°C, whereas Europe has a higher density between 10°C and 15°C. The Middle East is definitely one of the hotter regions with average temperatures, even around 40°C. In the next section, we will

3.3.5 Univariate Analysis

Understanding the global monthly temperatures is an integral part of the overall analysis. In univariate analysis, we analyse the average temperature for all the regions by month, where we should be able to see the various trends. We use another plotting library to set all of the plots in a fashion that can make them more interpretable. Analysis of the different regions helps us understand the general trends that are present across various months in the data. We see various distributions such as constant, sinusoidal, and even reverse sinusoidal in regions such as Australia.

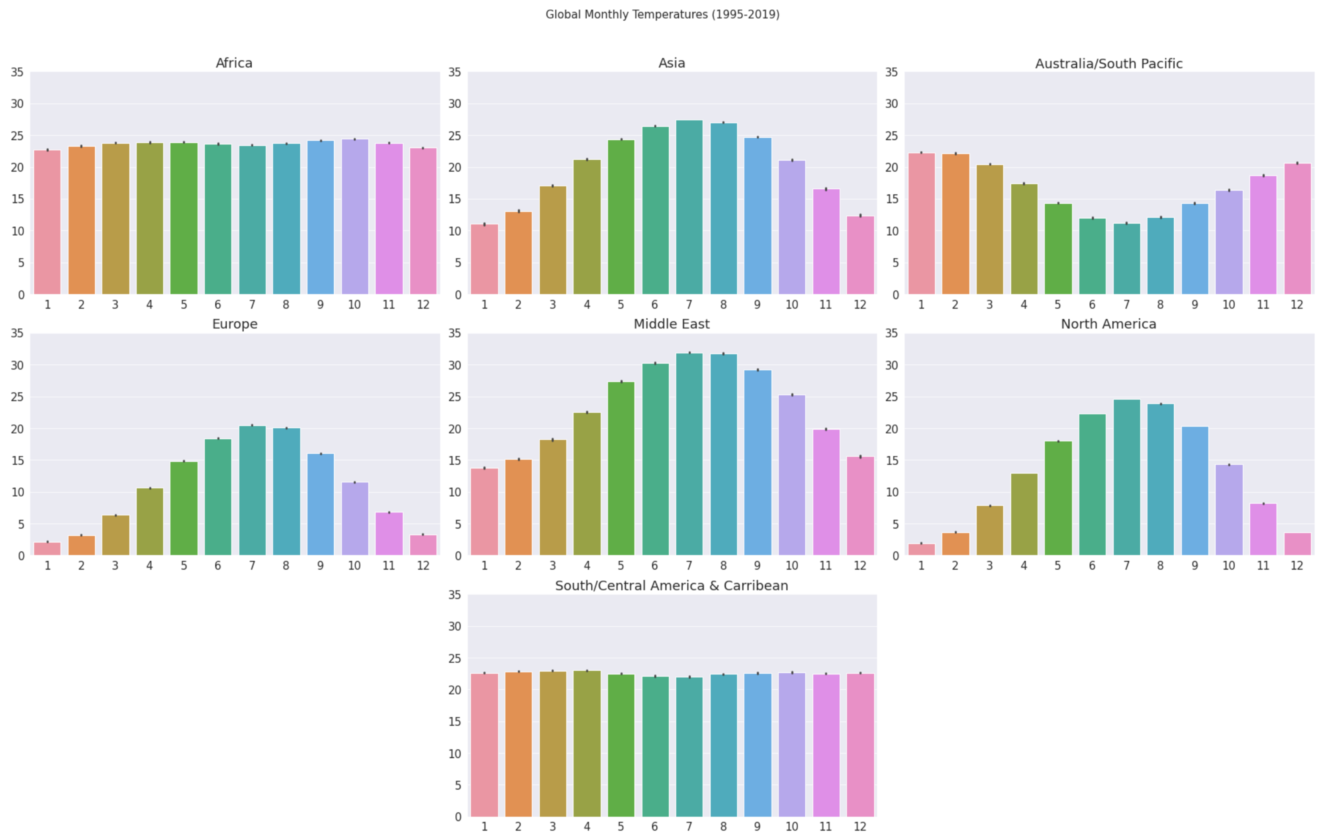


Figure 3.3.5.1: Univariate distribution of Global Monthly Temperatures

It is interesting to note that there is only one country that has a reverse trend as compared to most of the other countries; Australia has its hottest months in December, January and February. There is one other interesting trend, South/Central America and the Caribbean has a near constant trend, where the temperature varies between 22°C and 24°C across all its months.

3.3.6 Probability Distribution using KDE

An effective way to understand the probability distribution of a feature is by looking at the kernel distribution estimator. It takes in the distribution of the variable and a list of probabilities associated with each of its possible values and smooths it out for easy interpretation. When the probability is taken for a continuous random variable, it is known as a probability distribution function.

As depicted in Figure 3.3.6.1, we use a non-parametric density estimator function using Kernel Distribution Estimator to fit the model to the arbitrary distribution of the data. The underlying principle of KDE is a tree-based algorithm that focuses on accuracy, leveraging absolute tolerance and relative tolerance measures. We have used cross-validation as the kernel bandwidth as a smoothing parameter. In the next section, we will focus on our region of choice and understand more about the trends in Europe.

3.3.7 Deep-dive analysis in Europe

In this section, we will deep-dive into the region of Europe and understand more about the region before we put forth more focus on how we can perform modelling on Europe. We will start off by analysing the distribution of temperatures across Europe. We can interpret from Figure 3.3.7.1 that the majority of temperatures are distributed across -10°C to 30°C, which sort of forms a normal distribution. We see a higher probability distribution in the range of 5°C to 20°C. This is an expected distribution, as we had analysed in Section 3.3.4, as Europe is a colder region as compared to the rest of the world in the data that we have analysed. In the corresponding figure 3.3.7.2, we will look at the distribution of temperatures across the countries in Europe.

Russia is the country that has the broadest range of temperatures varying all the way from -30°C to about 35°C. We also note that Cyprus is the hottest country in Europe by median temperature. It is also interesting to note that Germany has the most outliers and hence in certain months, has the most variation as compared with the median. The region of focus would be the United Kingdom in the modelling section, where we notice that the distribution matches the average distribution across Europe. As part of this analysis, it would also make sense to have a look at the average temperature of the countries in Europe.

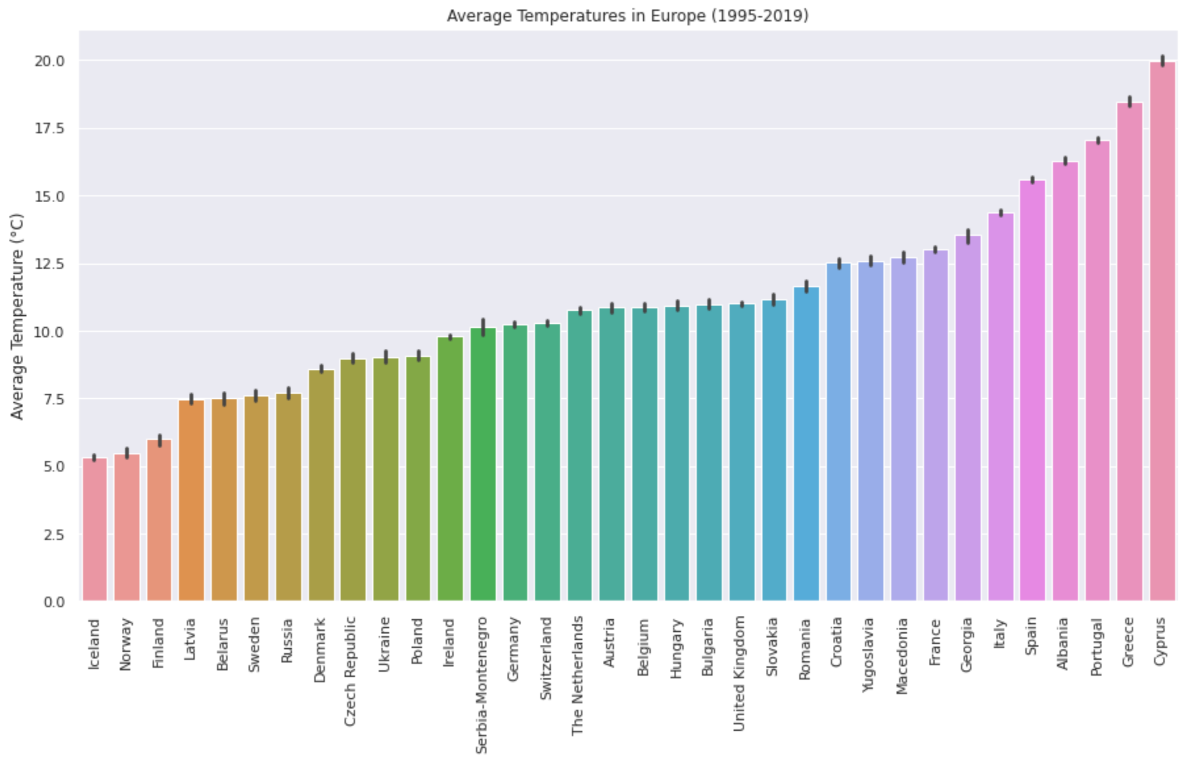


Figure 3.3.7.3: Average temperatures across the countries of Europe

Iceland is pegged to be the coldest country in Europe by mean temperature. This analysis is intriguing because the median temperature, as depicted in Figure 3.3.7.2, indicated that Norway would be a colder country. This is contrary to our assumption, as judging by the distribution, there is a narrower deviation from the median temperature in Iceland. The United Kingdom is in line with our expectations, with the average temperature at around 11°C. This makes the United Kingdom an apt choice to model. One final bit of analysis that can help us understand Europe's trends is by building a similar grid as in section 3.3.3, where we can plot a grid to understand the month-wise trend.

The colder months in Europe are getting less cold as we can see an increasing trend in the months of November and December. There is about a 1.7°C difference in the month of November, and we observe a considerable difference of 2.7°C in the month of December. We are now done with the exploratory data analysis, where we have analysed the trends of the various regions of the world and specifically Europe in detail. In the corresponding sections, we will model the data.

3.5 Time-Series Model

In this section, we will discuss the various models that we have implemented on the Temperature data by the University of Dayton. We will go over multiple advanced models and the process by which we have arrived at the results. It is essential that we present our analysis in a form that is reproducible. All of the models have been performed using the same data preprocessing steps that were discussed in section 3.3.1.

3.5.1 Timeseries Forecasting using Prophet

The Prophet library by Meta (previously Facebook) is one of the latest packages to hit the open-source community, which provides fast and reliable forecasts that are automated. It is based on an additive model, where nonlinear trends are fit considering yearly, weekly, and daily seasonality, plus a big bonus is that it considers holiday effects as well. It is recommended to use Prophet with time series data that has a strong seasonal effect and several cycles of historical data. Since our data is from 1995, it is a great fit for the aforementioned use case. Another significant advantage of the library is that it handles outliers well. It is the time series version of AutoML, where the model is fast and accurate, fully automatic, and the forecasts are even tunable. This is a model that can be used to perform forecasting at scale, even in a production environment.

For our use case, we take the rounded average temperature values, which are continuous variables and fit the prophet model on the preprocessed data. We use the make_future_dataframe and set the period to 730 and set the frequency to day, and perform our predictions.

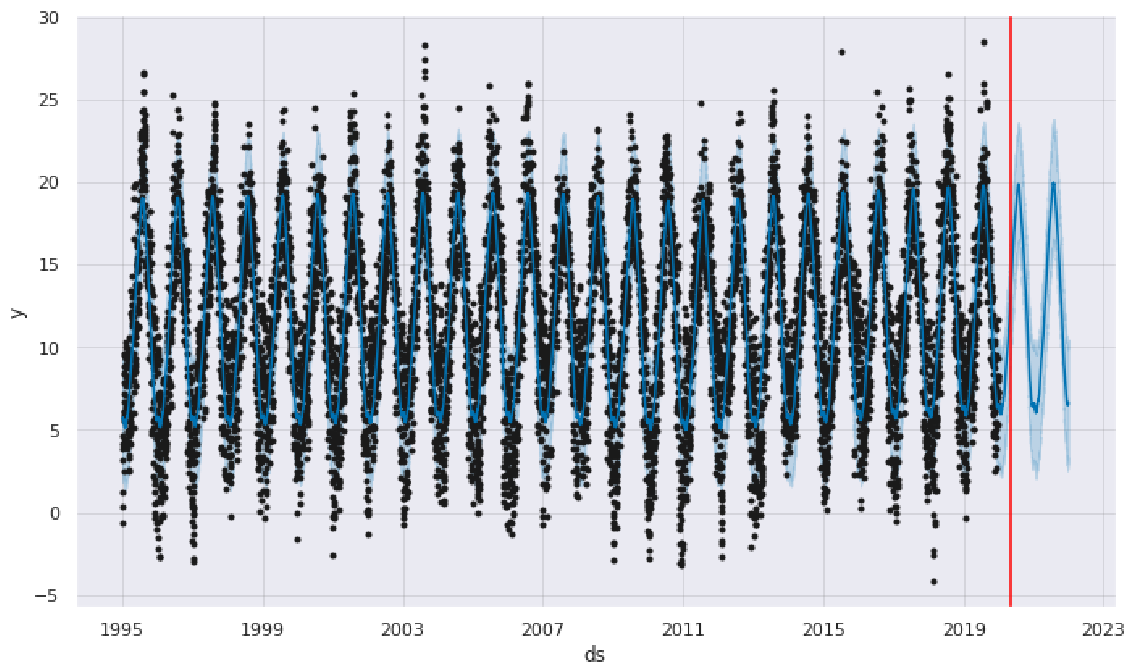


Figure 3.5.1.1: Predictions on temperature data using the Prophet library by Meta

AutoML is the future, and the Prophet library has predictions that are in line with the future predicted values, along with a confidence interval. In the next section, we will look at the Long Short-Term Memory (LSTM) algorithm and model our data.

3.5.2 LSTM

Long Short-Term Memory, or LSTMs, as they are popularly known, is a special kind of Recurrent Neural Network that is able to learn long-term dependencies. It is a model that is used to emulate the way that human beings think by leveraging gates. There is the sigmoid layer that outputs numbers between zero and one, dependent on the amount that each component should let in. There are three gates that use a pointwise multiplication operation and a sigmoid neural net layer.

To model our data effectively, we have done similar preprocessing and filtered out the test and train datasets in 2019. We then standardise our data using a MinMaxScaler() function from the sklearn package. We proceed to define the time steps and use a time series generator, where we have defined the batch size as 32.

In our case, the Rectified Linear Unit (ReLU) activation function gave us the best results. We have a total of 39,345 parameters based on the hyper-parameters that we have selected. We defined the callbacks at five and then proceeded to fit the model with 40 epochs in order to minimise loss. Once we have input the test generator values to the model, we collect the original target values from the test generator. Once we perform an inverse transform on the scaled results, we can proceed to evaluate the model.

As depicted in Figure 3.5.2.1, the predictions are quite close to the actual values in London. We will now proceed to evaluate the model, where we find that the Root Mean Squared Error is 1.96, which is all right for the time series model. Now, we will attempt to forecast beyond 2019 with ten cycles.

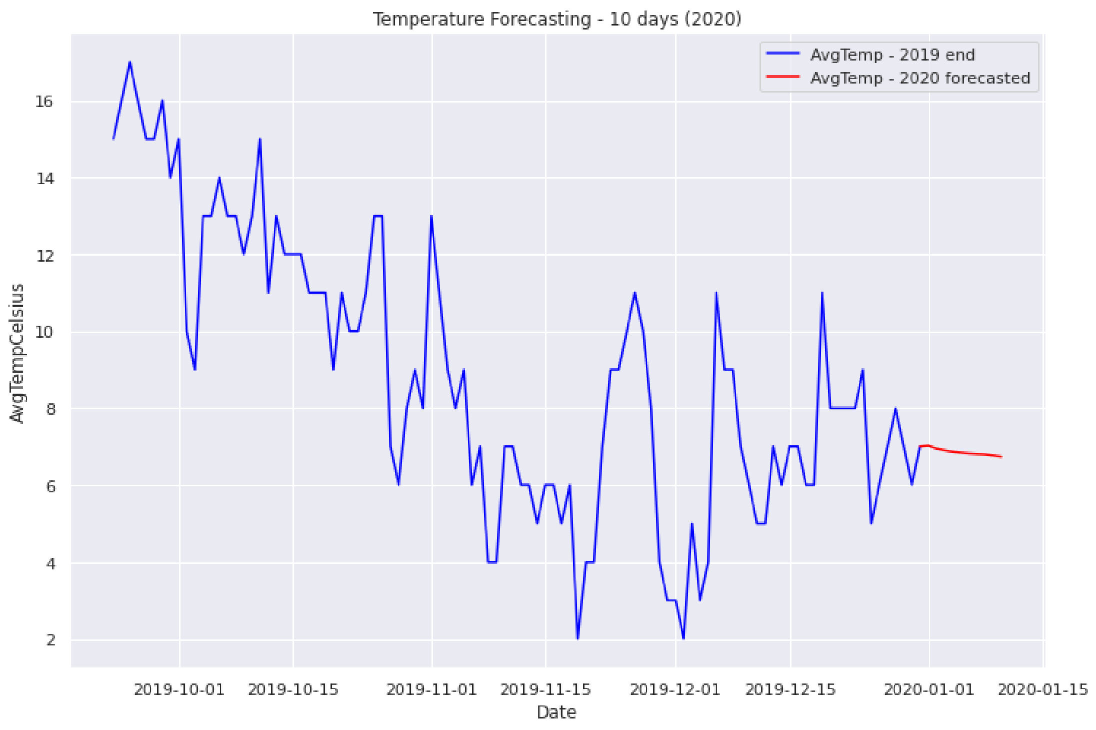


Figure 3.5.2.2: Prediction beyond 2019 using LSTM

The results from LSTM beyond 2019 can be considered as an indication of which direction the temperature is likely to head but is not accurate enough to use. Since there is a high amount of seasonality in the data, we can attempt the Seasonal Autoregressive Integrated Moving Average (Seasonal ARIMA) model next.

3.5.3 Seasonal ARIMA

To reduce computation, we will downsample the data to a monthly frequency. Here, we consider both the seasonal and non-seasonal parts of the model to come up with accurate results. Let us first take a look at the daily average temperature for London from 1995 to 2019; this will give us an intuition of the data.

To understand more about the trends, we also take a 31-day and 365-day rolling mean of the data. We have also performed a rolling standard deviation of the model. We will have a look at the plots before we go ahead and perform seasonal decomposition to break the time series down to its various components.

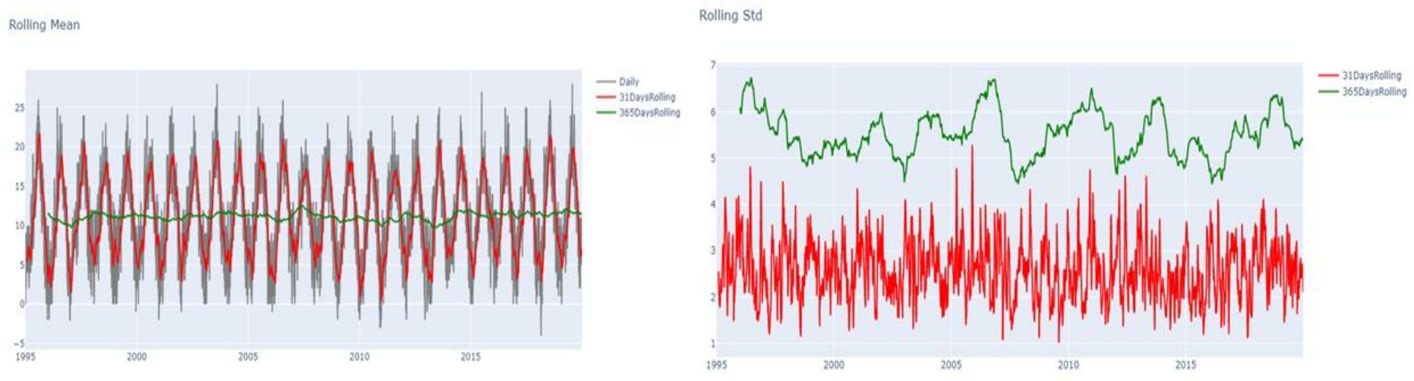


Figure 3.5.3.2: Rolling mean and Rolling standard deviation of London Temperatures

Next, we will perform the seasonal decomposition of the data, where we break the time series data into three components of trend, seasonality and residual (noise). This will help us get an intuition of the data before we proceed to the Dickey-Fuller test to test the stationarity of the data.

Here, we notice that there is a strong seasonal trend, which makes this data a good fit to the model using the Seasonal ARIMA model. One last test we can perform to test for stationarity of the model is the Dickey-Fuller-Test, where we get a very low p-value (1.6140204791222822e-07). Since the p-value is very small, we reject the Null Hypothesis, which states that the time series data is non-stationary. Now, we proceed to run seasonal ARIMA, followed by Auto ARIMA.

Here, we have set it to 12 cycles, as the data that we will be using is monthly. Now, performing Auto ARIMA on the data for the first five months, we come to the final model prediction.

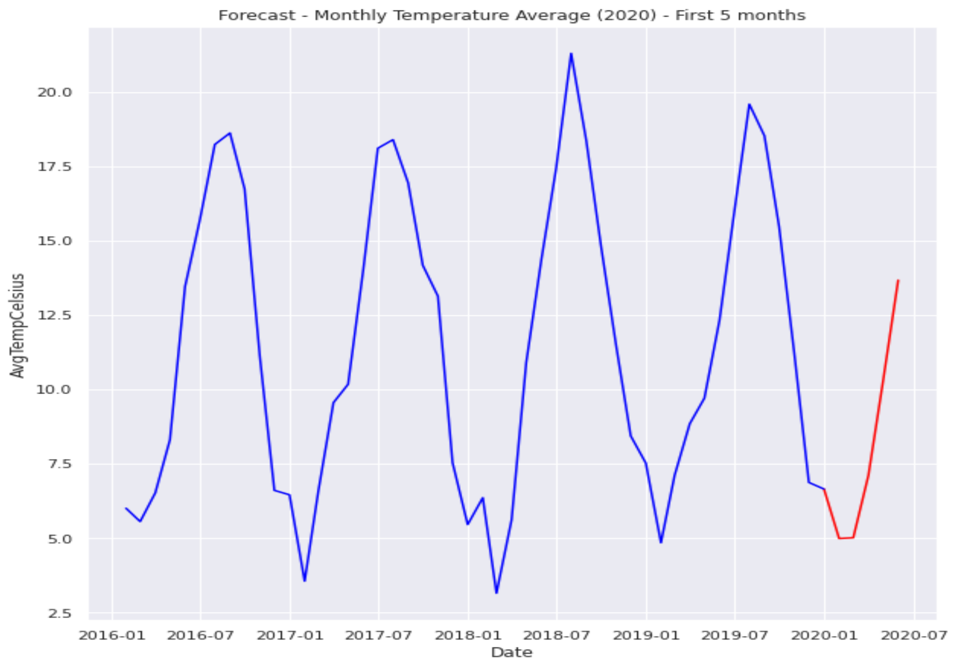


Figure 3.5.3.4: Forecast of monthly temperature average using variations of ARIMA

The forecast of the model is in line with our expectations. Auto ARIMA is able to give us good results that we can leverage, which is an accurate representation of future data.

3.6 Summary

In this section, we went over multiple facts of our analysis. We looked at the preprocessing of the data, performed missing value analysis and imputation of missing values, looked at the distribution of variables, performed outlier analysis, and did a univariate analysis of the temperature data. Further, we also looked at the probability distribution of the variables using the kernel distribution estimator, post which we took a deep-dive look into Europe. In our modelling section, we looked at the implementation of the Prophet Model by Facebook, modelled using LSTMs, and even looked at Seasonal ARIMA and Auto-ARIMA models. In the proceeding section, we will summarise our learnings from the thorough literature review and analysis that we have performed.

4. Summary and Discussion

As we know, this analysis was conducted on a large set of data with over 2,885,424 rows of information under these columns. Thus the data has been filtered to different regions, and the analysis as well as the predictions. We preprocessed the data by grouping the missing values of countries. We observe that the Latin American, Asian and African countries have the most missing data. This is rectified by implementing an imputation of the missing values with the mean temperature of the same region from the past years. Moreover, hence the number of missing values is zero for the analysis. We then have a look at the distribution of the variables to see what type of analysis would be suitable. The average temperature for all the countries present in the dataset has been plotted in Figure 3.3.3.2 from the year 1995 to 2019, and it is quite evident that the overall global temperature spread across the globe shows an increasing trend in the global temperature. We see that the hottest regions in the world exist in Africa and the middle east, which indicates that the hot regions are getting hotter, whereas regions in Europe and North America are getting colder. We then generate a few boxplots to check for outliers, as this is the best way to do so, and we see that they are insignificant as box plots give us an idea of the median temperature, which is primarily used in the models. We then conducted a univariate analysis and found interesting observations like the fact that Australia has their hottest months between December and February which is when most of the other parts of the world are cold. Also, the Caribbean has a temperature between 22 and 24 degrees centigrade throughout the year.

The kernel Distribution Estimator is what we then used on the dataset in order to determine the probability distribution of a feature. We delve deep into the European region to conduct an analysis, and we find that the temperatures are spread in the -10 to 30 degrees centigrade range and are normally distributed. Europe also seems to be colder in general than other parts of the world. Iceland is the coldest country in the region in terms of mean temperature, but if we look at it in terms of median values, then Norway seems to be the clear winner being the coldest, and Cyprus seems to be the hottest. The United Kingdom does not have any extremes and has an average temperature of eleven degrees centigrade. We also observe that there is a clear increase in temperature across the regions in Europe, with an increase of 1.7°C in November and 2.7°C in the month of December. Now that we have completed the Analysis part of things, we generate some models to forecast the temperature using various algorithms. The first algorithm that we implement is one of the time-series analyses with the help of the prophet package that was released by Meta, formerly known as Facebook. In this package, the primary principle used is the additive models, and thus we see that the predicted values, as well as the actual values for the city of London, are more or less similar, which shows a great degree of accuracy; this is then used for the first five months after 2019 to generate our predictions which shows an overall increase in the average temperature of the city. This is clearly depicted in figure 3.5.1.1. This kind of model has predictions that are in line with the future predicted values, along with a confidence interval. The next algorithm that we take into consideration is the Long short-term Memory which is an application of Artificial Intelligence; in this case, we set the batch size as 32 and also set the number of epochs to 40 to reduce and minimise loss and observe that the actual values, in this case, are almost the same as the predicted values once again which then shows us that the model is accurate and well fit for the time-series data and as such we use the same to predict the future values which are depicted in figure 3.5.2.2 although this is not the best fit model as it does not take the seasonality component which is rather crucial to take into consideration as it can change the entire trajectory of a trend or season, so the next and final model that we fit is that of the Seasonal Auto- Regression model which is the SARIMA or Seasonal Arima.

We take the seasonal as well as the non-seasonal components and fit them onto our time-series data which is then passed through an Auto Arima model to best determine which type of time series is best suited for this type of dataset; we then sit a Seasonal ARIMA model to the same and check the stationarity of the data using the Dickey-Fuller test as well as a hypothesis test where we find the p-value to be lesser than 0.05 and thus we can reject the null hypothesis and determine that the time series data is, in fact, a non-stationary component which is suitable for this type of model. Using the rolling mean and the rolling average of London from the year 1995 to 2019, we break the time-series data into three major components which is the trend, seasonality and also the residual or noise component, which is then used for our analysis after which we set twelve cycles and the seasonal Arima model is fitted onto the dataset to find values that are quite closely correlated with the actual values, and thus we use this to predict the average temperature of the city of London which is depicted in figure 3.5.3.4 which finally leads us to the conclusion of this project which has measured, analysed and predicted the average temperature of cities and regions across the world from 1995 to 2019.

5. Conclusion

As we can see from the analysis and the results, we have concurred both by the analysis of the average temperature of various countries and cities across the world have shown a trend of an overall increase in the average temperature. This, along with the prediction models that have been generated for this project in the United Kingdom as well as the city of London, also shows a prediction of an increase in temperature, which is potentially detrimental to the whole world in more ways than one. There are many drawbacks to this kind of analysis, such as the quality of the data and how it is procured; since we are dealing with such a large quantity, there arises the question of the sources and the errors that could be included. Various satellites and devices for this specific purpose are used to combat these errors. This project could pave the way to determine the various effects this increase in temperature could cause on this planet, such as an increase in natural calamities due to the increase in temperature of a region which was measured and predicted quite accurately. This can also be used by corporations that may require natural resources in a region and can make business decisions keeping the component of temperature as an important aspect. To conclude this project, we have discussed the prospective use cases and applications of this analysis as well as the drawbacks that have prevented this from being accurate, but this is definitely a step in the right direction to combat climate change and the detrimental effects it has on all the living beings on the planet earth.

6. Reference

1. Jackson, S. T.. "climate change." Encyclopedia Britannica, April 27, 2021. Available at https://www.britannica.com/science/climate-change.
2. Beniston, M., Rebetez, M., Giorgi, F. et al. An analysis of regional climate change in Switzerland. Theor Appl Climatol 49, 135–159 (1994). https://doi.org/10.1007/BF00865530
3. Shekhar, M. S., Chand, H., Kumar, S., Srinivasan, K. and Ganju, A. (2010) "Climate-change studies in the western Himalaya," Annals of Glaciology. Cambridge University Press, 51(54), pp. 105–112. doi: 10.3189/172756410791386508.
4. Kubat, M. (1999). Neural networks: a comprehensive foundation by simon haykin, macmillan, 1994, isbn 0-02-352781-7. The Knowledge Engineering Review, 13(4):409–412.
5. Chithra, N.R., Thampi, S.G., Surapaneni, S. et al. prediction of the likely impact of climate change on monthly mean maximum and minimum temperature in the Chaliyar river basin, India, using ANN-based models. Theor Appl Climatol 121, 581–590 (2015). https://doi.org/10.1007/s00704-014-1257-1
6. W. Xie, M. He and B. Tang, "Data-Enabled Correlation Analysis between Wildfire and Climate using GIS," 2020 3rd International Conference on Information and Computer Technologies (ICICT), 2020, pp. 31-35, doi: 10.1109/ICICT50521.2020.00013.
7. Dimri, T., Ahmad, S. & Sharif, M. Time series analysis of climate variables using seasonal ARIMA approach. J Earth Syst Sci 129, 149 (2020). https://doi.org/10.1007/s12040-020-01408-x
8. Hrnjica, B., Bonacci, O. Lake Level Prediction using Feed Forward and Recurrent Neural Networks. Water Resour Manage 33, 2471–2484 (2019). https://doi.org/10.1007/s11269-019-02255-2
9. Pak, U.-J. et al. (2018) "A hybrid model based on convolutional neural networks and long short-term memory for ozone concentration prediction," Air Quality, Atmosphere & Health, 11(8), p. 883. doi:10.1007/s11869-018-0585-1.

7. Appendix

Figure Table

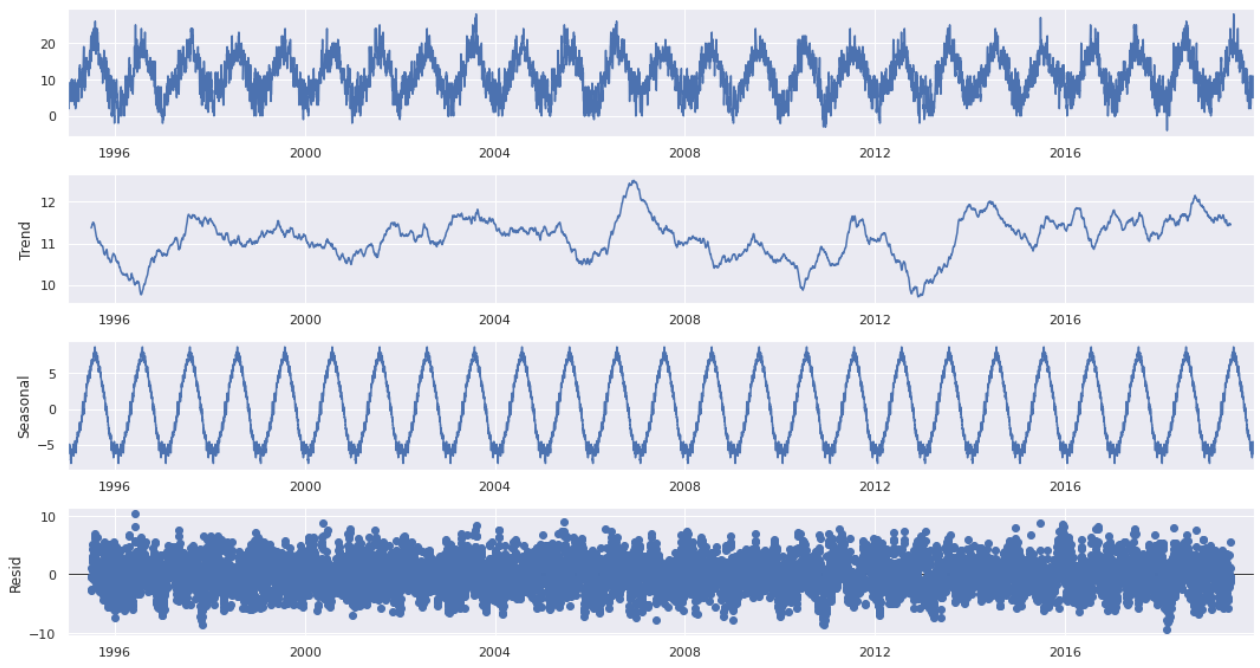


Figure 3.5.3.2: Seasonal decomposition of London temperatures from the year 1995 to 2019

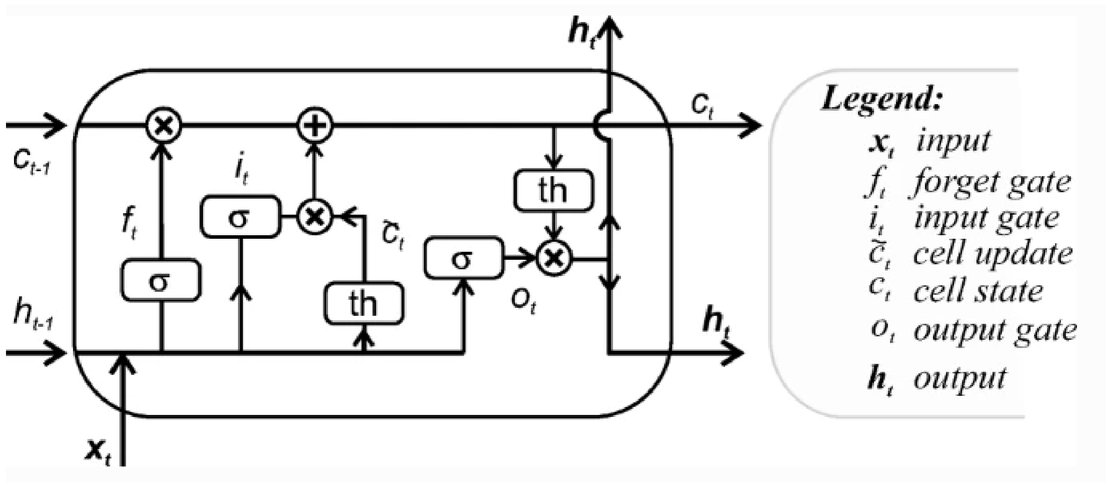


Figure 2.3 represents the internal structure of the LSTM cell

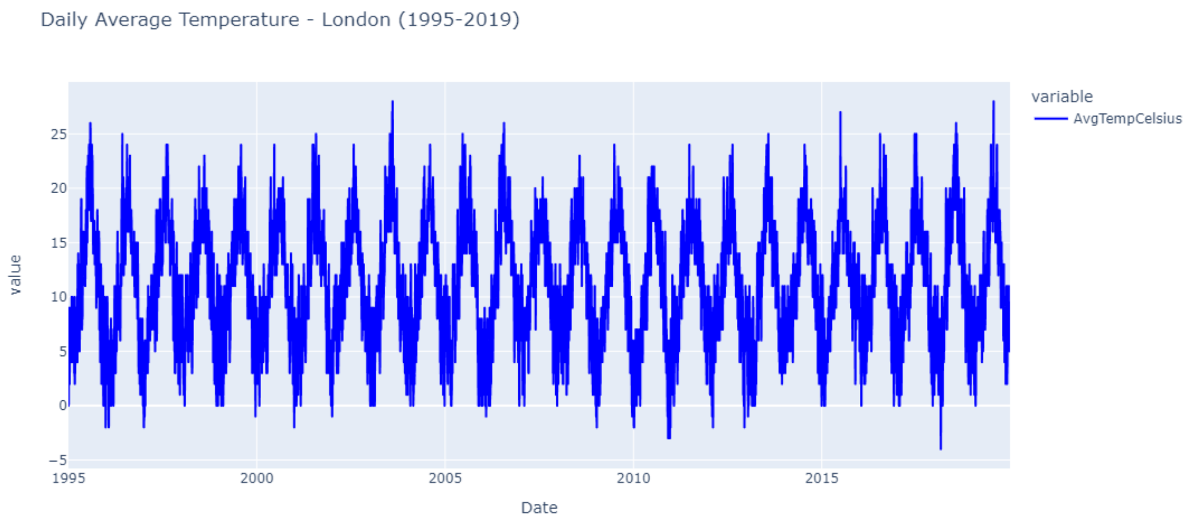


Figure 3.5.3.1: Daily average temperature of London

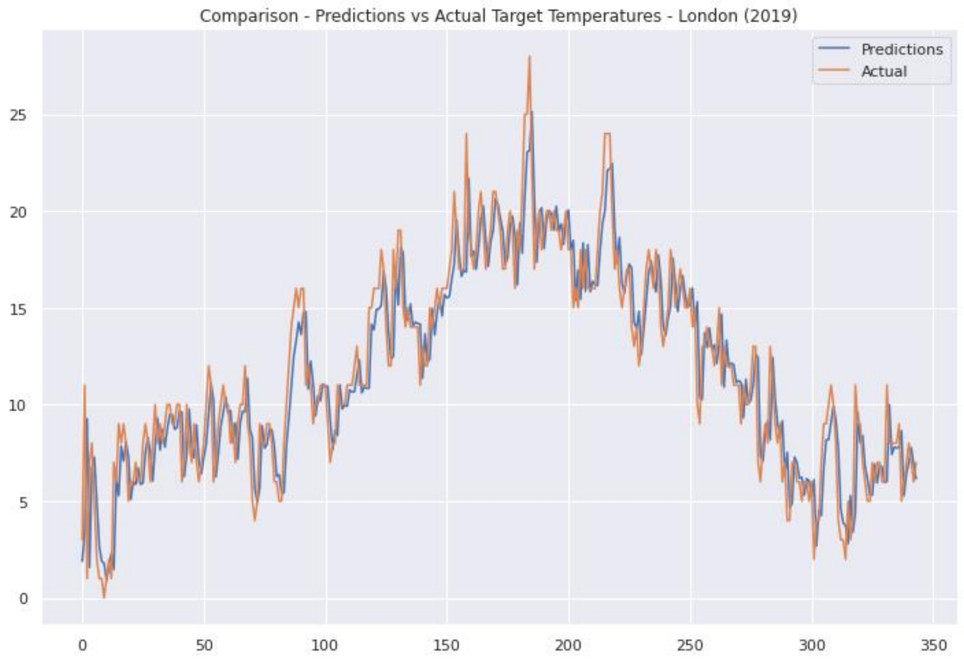


Figure 3.5.2.1: Comparison of the actual and predicted values of London for the year 2019

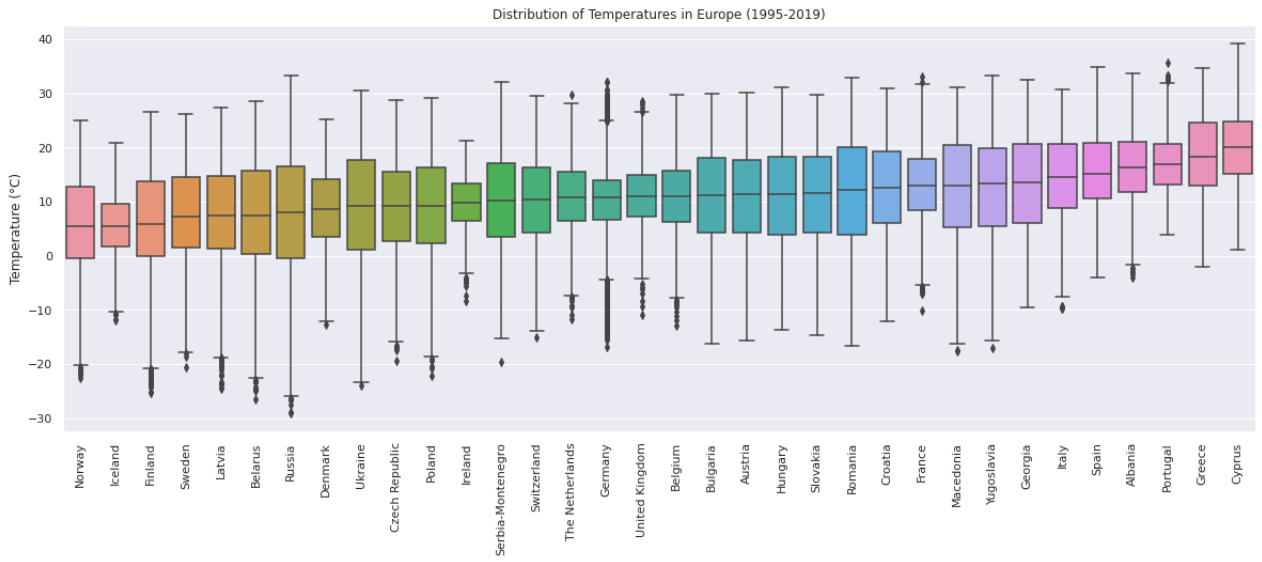


Figure 3.3.7.2: Distribution of Temperature across countries between 1995 to 2019

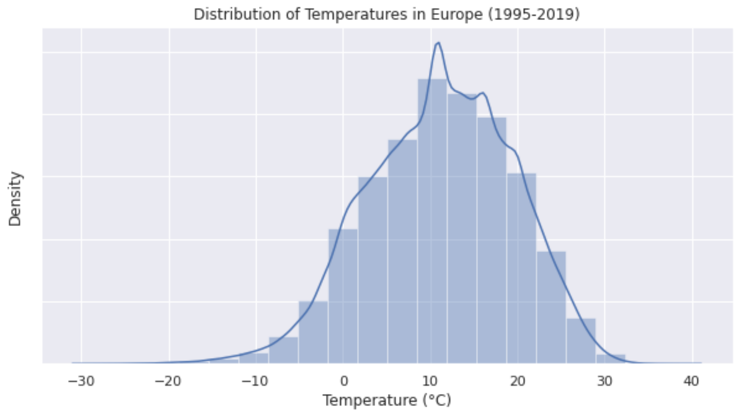


Figure 3.3.7.1: Distribution of temperatures across Europe

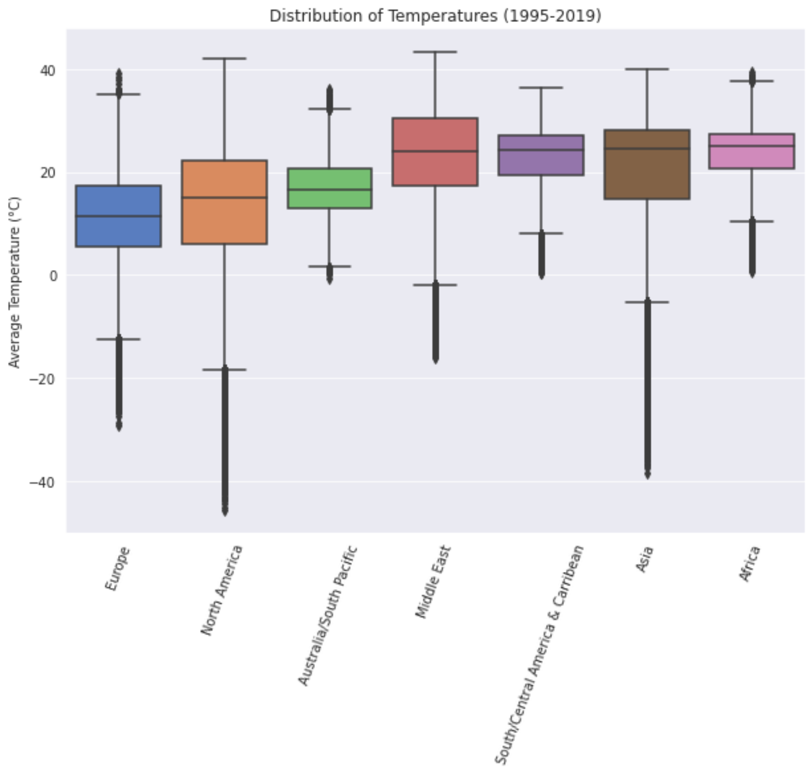
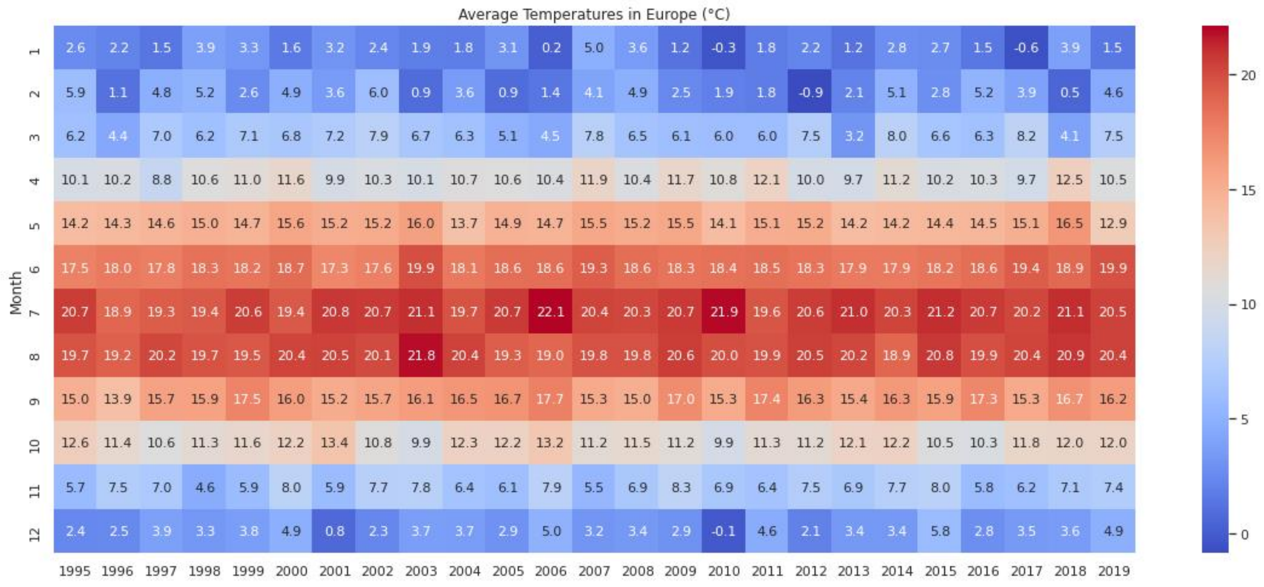


Figure 3.3.4.1: Box plot of the distribution of temperature by region from 1995 to 2019



Month	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019
1	2.6	2.2	1.5	3.9	3.3	1.6	3.2	2.4	1.9	1.8	3.1	0.2	5.0	3.6	1.2	-0.3	1.8	2.2	1.2	2.8	2.7	1.5	-0.6	3.9	1.5
2	5.9	1.1	4.8	5.2	2.6	4.9	3.6	6.0	0.9	3.6	0.9	1.4	4.1	4.9	2.5	1.9	1.8	-0.9	2.1	5.1	2.8	5.2	3.9	0.5	4.6
3	6.2	4.4	7.0	6.2	7.1	6.8	7.2	7.9	6.7	6.3	5.1	4.5	7.8	6.5	6.1	6.0	6.0	7.5	3.2	8.0	6.6	6.3	8.2	4.1	7.5
4	10.1	10.2	8.8	10.6	11.0	11.6	9.9	10.3	10.1	10.7	10.6	10.4	11.9	10.4	11.7	10.8	12.1	10.0	9.7	11.2	10.2	10.3	9.7	12.5	10.5
5	14.2	14.3	14.6	15.0	14.7	15.6	15.2	15.2	16.0	13.7	14.9	14.7	15.5	15.2	15.5	14.1	15.1	15.2	14.2	14.2	14.4	14.5	15.1	16.5	12.9
6	17.5	18.0	17.8	18.3	18.2	18.7	17.3	17.6	19.9	18.1	18.6	18.6	19.3	18.6	18.3	18.4	18.5	18.3	17.9	17.9	18.2	18.6	19.4	18.9	19.9
7	20.7	18.9	19.3	19.4	20.6	19.4	20.8	20.7	21.1	19.7	20.7	22.1	20.4	20.3	20.7	21.9	19.6	20.6	21.0	20.3	21.2	20.7	20.2	21.1	20.5
8	19.7	19.2	20.2	19.7	19.5	20.4	20.5	20.1	21.8	20.4	19.3	19.0	19.8	19.8	20.6	20.0	19.9	20.5	20.2	18.9	20.8	19.9	20.4	20.9	20.4
9	15.0	13.9	15.7	15.9	17.5	16.0	15.2	15.7	16.1	16.5	16.7	17.7	15.3	15.0	17.0	15.3	17.4	16.3	15.4	16.3	15.9	17.3	15.3	16.7	16.2
10	12.6	11.4	10.6	11.3	11.6	12.2	13.4	10.8	9.9	12.3	12.2	13.2	11.2	11.5	11.2	9.9	11.3	11.2	12.1	12.2	10.5	10.3	11.8	12.0	12.0
11	5.7	7.5	7.0	4.6	5.9	8.0	5.9	7.7	7.8	6.4	6.1	7.9	5.5	6.9	8.3	6.9	6.4	7.5	6.9	7.7	8.0	5.8	6.2	7.1	7.4
12	2.4	2.5	3.9	3.3	3.8	4.9	0.8	2.3	3.7	3.7	2.9	5.0	3.2	3.4	2.9	-0.1	4.6	2.1	3.4	3.4	5.8	2.8	3.5	3.6	4.9

Figure 3.3.7.4: Grid of Average Temperatures in Europe across months and year

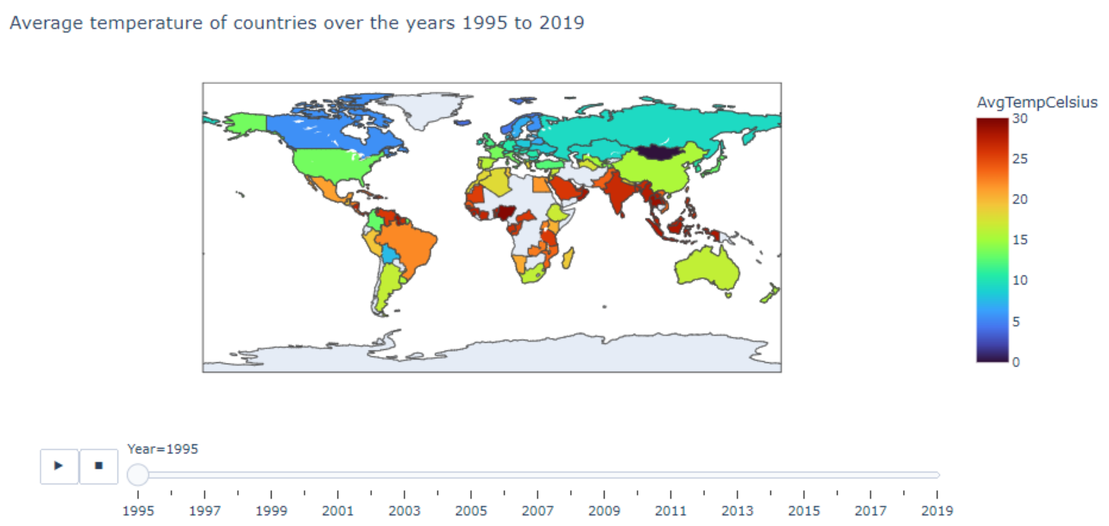


Figure 3.3.3.2: Plot of Average Temperature in the Year 1995 by Country

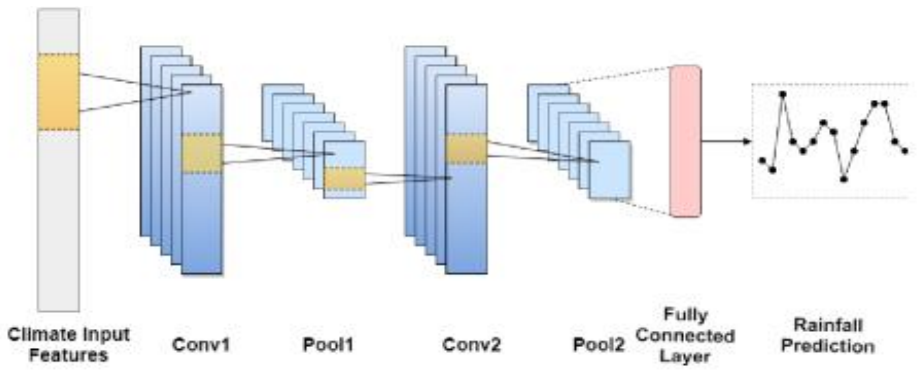


Figure 2.6 The layers in CNN architecture (Haidar, A. et al. '2018')

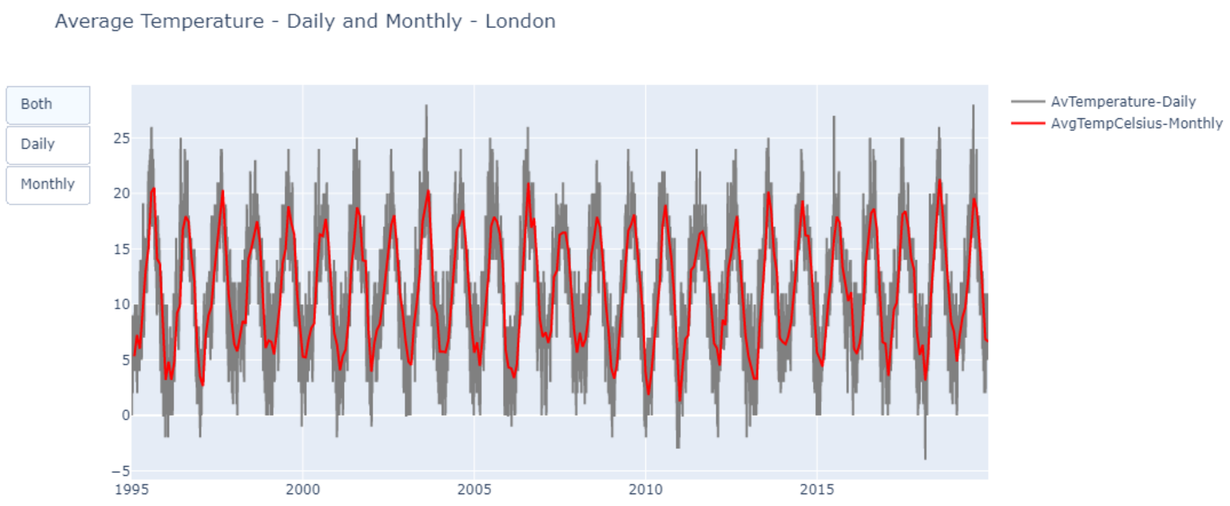


Figure 3.5.3.3: The daily and monthly average temperature of London

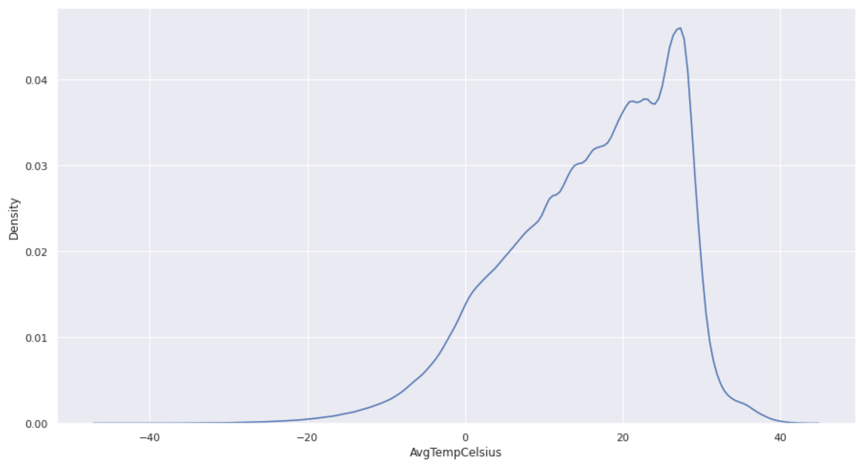


Figure 3.3.6.1: Probability Distribution using KDE

for Average Temperature in degrees Celsius

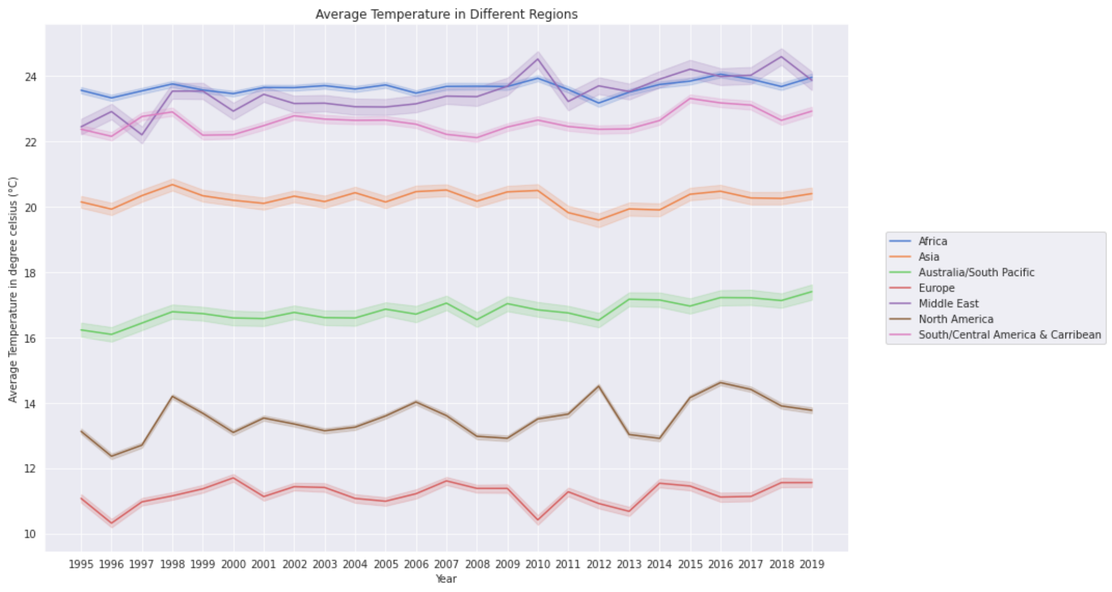


Figure 3.3.3.4: Plot of the Average Temperature

in different regions across years

Code

```
# -*- coding: utf-8 -*-
""" Detection and Prediction of Climate Change with Temperature
v3.ipynb

# Detection and Prediction of Climate Change with Temperature
* Author: Adnan Ahmed
* Year: 2022

This is a dataset to analyse climate change that has been
sourced from the University of Dayton
(https://academic.udayton.edu/kissock/http/Weather/source.htm)
. We will analyse this dataset, perform analysis, model and
interpret the results.
"""

!pip install pystan~=2.14
!pip install fbprophet
!pip install -q pmdarima
# ! pip install --upgrade Cython
# ! pip install --upgrade
git+https://github.com/statsmodels/statsmodels
# import statsmodels.api as sm

!pip install pmdarima scipy==1.2 -Uqq

# Importing the required libraries

from keras.layers import Activation, Dense, LSTM, Dropout,
GlobalMaxPooling1D
```

```
from        tensorflow.keras.preprocessing.sequence        import
TimeseriesGenerator
from keras.callbacks import ModelCheckpoint, EarlyStopping
from statsmodels.tsa.seasonal import seasonal_decompose
from sklearn.model_selection import train_test_split
from tensorflow.keras.callbacks import EarlyStopping
from dateutil.relativedelta import relativedelta
from keras.models import Sequential, load_model
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error

from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM,Dense
from statsmodels.tsa.stattools import adfuller
from tensorflow.keras. optimisers import Adam

from tqdm import tqdm_notebook as tqdm
import matplotlib.gridspec as gridspec
from pmdarima.arima import auto_arima
from matplotlib import pyplot as plt

import plotly.graph_objects as go
import matplotlib.pyplot as plt
from fbprophet import Prophet
from datetime import datetime
import plotly.express as px
from scipy import stats

import seaborn as sns
import pandas as pd
import numpy as np
import datetime
import warnings
import plotly
import random

pd.set_option('display.max_columns', 500)
pd.set_option('display.max_rows', 500)
pd.set_option('display.width', 1000)

warnings.filterwarnings("ignore")
sns.set_style('darkgrid')

# Mounting the data
from google.colab import drive
drive.mount('/content/drive')

# Reading the csv file and viewing a sample to ensure the data
has been correctly read
temperature     =     pd.read_csv('/content/drive/MyDrive/Colab
Notebooks/Temperature/city_temperature.csv',low_memory=False)
temperature.head(5)

# Creating a copy of the dataset
df = temperature.copy(deep=True)

# Typecasting of features to ensure that the data type is
standardised
# We also set the right number of bytes to ensure that memory
is optimaaly used
df[['Month', 'Day']] = df[['Month', 'Day']].astype('int8')
```

```
df[['Year']] = df[['Year']].astype('int16')
df['AvgTemperature'] = df['AvgTemperature'].astype('float16')

# Resetting the index of the pandas dataframe
df.reset_index(drop=True, inplace=True)

"""## Preprocessing and data cleaning"""

df.head()

# Removing ambiguous values from Year column in the data based
on preliminary analysis
df = df[df.Year != 200]
df = df[df.Year != 201]
df = df[df.Year != 2020]

# Dropping the State column as it only applies to the US and we
will not be analysing it in detail
df = df.drop('State', axis = 1)

# Creating a new column, date in datetime format
df['Date']   =   pd.to_datetime(df.Year.astype(str)   +   '/'   +
df.Month.astype(str))

# Let us look at the percentage of missing values
missing   =   pd.DataFrame(df.loc[df.AvgTemperature   ==   -99,
'Country'].value_counts())

# We group by country and look at the percentage of missing
values
missing['TotalNumberOfMissingPoints']                         =
df.groupby('Country').AvgTemperature.count()

# Making the dataframe for percentge of missing values and
sorting it by descending order of percentage of missing values
missing['PercentageOfValuesMissing'] = missing.apply(lambda
row: (row.Country/row.TotalNumberOfMissingPoints)*100, axis =
1).round(2)
missing['PercentageOfValuesMissing'] =
missing['PercentageOfValuesMissing'].astype(str) + '%'
missing.sort_values(by=['PercentageOfValuesMissing'],
inplace=True, ascending = False)
missing.head(20)

# Missing values in AvgTemperature column as indicated as -99,
so we replace it by NumPy's NaN function
df.loc[df.AvgTemperature == -99, 'AvgTemperature'] = np.nan

# Let us have a look at the total number of missing data points
df.AvgTemperature.isna().sum()

# Filling in the missing values with the mean temperature of
the same city and same date
df['AvgTemperature'] =
df['AvgTemperature'].fillna(df.groupby(['City']).AvgTemperatur
e.transform('mean'))

# Looking at the rest of the missing values after we have
performed missing value imputation
df.AvgTemperature.isna().sum()

df.loc[df.AvgTemperature.isna(), 'City'].value_counts()
```

```
# Missing values should be removed now; the result of this query
is 0
df.AvgTemperature.isna().sum()

# Let us now convert this dataset into Celsius
# °F to °C: (°F − 32) × 5/9 = °C
df['AvgTempCelsius'] = (df.AvgTemperature -32)*(5/9)
df  = df.drop(['AvgTemperature'], axis = 1)

# Temperatures beyond two decimal points are not as useful, so
let limit the float values to one and two decimal points
df['AvgTempCelsius_rounded']  =  df.AvgTempCelsius.apply(lambda
x: "{0:0.2f}".format(x))
df['AvgTempCelsius_rounded2'] = df.AvgTempCelsius.apply(lambda
x: "{0:0.1f}".format(x))

# Converting the new rounded values to numeric
df['AvgTempCelsius_rounded']                                   =
pd.to_numeric(df['AvgTempCelsius_rounded'])
df['AvgTempCelsius_rounded2']                                  =
pd.to_numeric(df['AvgTempCelsius_rounded2'])

# Sanity check of a sample of the data
df.sample(5)

data                                                           =
df[['Country','Year','AvgTempCelsius']].groupby(['Country','Ye
ar']).mean().reset_index()
px.choropleth(data_frame=data,locations="Country",locationmode
='country
names',animation_frame="Year",color='AvgTempCelsius',color_con
tinuous_scale = 'Turbo',title="Average temperature of countries
over the years 1995 to 2019")

# Plotting the Yearly Global Average Temperature
plt.figure(figsize=(15,8))
sns.lineplot(x = 'Year', y = 'AvgTempCelsius', data = df ,
palette='tab10')

# Setting the plot parameters
plt.title('Average Global Temperatures')
plt.ylabel('Average Temperature in degrees centigrade (°C)')
plt.xlabel('')
plt.xticks(range(1995,2020))
plt.show();

# Making a new dataframe to plot the average temperature by
month
df_mean_month = df.groupby(['Month',
'Year']).AvgTempCelsius_rounded2.mean()
df_mean_month = df_mean_month.reset_index()
df_mean_month = df_mean_month.sort_values(by = ['Year'])

# Making a pivoted dataframe with Month, Average Temperature
and Year
df_pivoted = pd.pivot_table(data= df_mean_month,
                    index='Month',
                    values='AvgTempCelsius_rounded2',
                    columns='Year')

# Plotting the average temperatures by year and month
```

```
plt.figure(figsize=(20, 8))
sns.heatmap(data = df_pivoted, cmap='coolwarm', annot = True,
fmt=".1f", annot_kws={'size':11})
plt.xlabel('Year')
plt.ylabel('Month (Number)')
plt.title('Average Global Temperatures in degrees centigrade
(°C)')
plt.show();

# Plotting the average temperature of the Regions
s                                                           =
df.groupby(['Region'])['AvgTempCelsius'].mean().reset_index().
sort_values(by='AvgTempCelsius',ascending=False)
s.style.background_gradient(cmap="coolwarm")

# Plotting the average temperature of different regions over
time
f = plt.figure(figsize=(15,8))
sns.lineplot(x = 'Year', y = 'AvgTempCelsius', hue = 'Region',
data = df , palette='muted')

# Filling the required parameters of the plot
plt.title('Average Temperature in Different Regions')
plt.ylabel('Average Temperature in degree celsius (°C)')
plt.xlabel('Year')
plt.xticks(range(1995,2020))
plt.legend(loc='center      left',      bbox_to_anchor=(1.04,
0.5),ncol=1)
plt.tight_layout()
plt.show();
```

```
# Distribution of temperatures across various regions
region_sorted                                                    =
df.groupby('Region')['AvgTempCelsius'].median().sort_values().
index

with sns.color_palette("muted"):
    f, ax = plt.subplots(figsize=(10, 7))
    sns.boxplot(data = df.sort_values("AvgTempCelsius"), x =
'Region', y = 'AvgTempCelsius', order = region_sorted)
    plt.xticks(rotation = 70)
    plt.title('Distribution of Temperatures (1995-2019)')
    plt.xlabel('')
    plt.ylabel('Average Temperature (°C)')

# Making a violin plot of the distribution of average
temperatures by country
with sns.color_palette("muted"):
    f, ax = plt.subplots(figsize=(15, 5))
    sns.violinplot(data = df.sort_values("AvgTempCelsius"), x =
'Region', y = 'AvgTempCelsius_rounded', order = region_sorted)
    plt.xticks(rotation = 45)
    plt.title('Distribution of Average Temperatures (1995-
2019)')
    plt.xlabel('')
    plt.ylabel('Average Temperature (°C)')
    plt.show;

# Plotting the monthly average temperature by month in various
Regions

regions = df.Region.unique().tolist()
```

```
number_plot = [0, 0, 0, 1, 1, 1, 2]
position_a = [0, 2, 4, 0, 2, 4, 2]
position_b = [2, 4, 6, 2, 4, 6, 4]

fig = plt.figure(figsize = (25,15))
plt.suptitle('Global  Monthly  Temperatures  (1995-2019)',  y  =
1.05, fontsize=15)

gs = gridspec.GridSpec(3, 6)

for i in range(7):
    #ax = plt.subplot(3, 3, i+1)
    ax            =            plt.subplot(gs[number_plot[i],
position_a[i]:position_b[i]])
    sns.barplot(x  =  'Month',  y  =  'AvgTempCelsius_rounded2',
data = df[df.Region == regions[i]])
    ax.title.set_text(regions[i])
    ax.set_ylim((0,35))
    ax.set_xlabel('')
    ax.set_ylabel('')

plt.subplots_adjust(wspace = 0.5)
plt.rcParams.update({'font.size': 10})
plt.savefig('demographics.png')
plt.tight_layout()
plt.show();

# Top 5 hottest countries

s =
df.groupby(['Country'])['AvgTempCelsius'].mean().reset_index()
.sort_values(by='AvgTempCelsius',ascending=False)[:5]
s.style.background_gradient(cmap="Reds")

# Top 5 coldest countries

s =
df.groupby(['Country'])['AvgTempCelsius'].mean().reset_index()
.sort_values(by='AvgTempCelsius',ascending=True)[:5]
s.style.background_gradient(cmap="Blues")

# Plotting the Kerner Density Estimation (KDE) for
AvgTempCelsius
sns.set(rc={'figure.figsize':(15,8)})
sns.kdeplot(data=df, x="AvgTempCelsius")

# Selecting a sub-section of the data and filtering for Europe
df_europe = df[df.Region == 'Europe'].copy()

# Sanity check via the sample function on the Europe dataset
df_europe.sample(5)

# Plotting the distribution of temperatures in Europe
f, ax = plt.subplots(figsize=(10, 5))
sns.distplot(df_europe.AvgTempCelsius_rounded, bins = 20)
plt.title('Distribution of Temperatures in Europe (1995-2019)')
plt.xlabel('Temperature (°C)')
#ax.axes.yaxis.set_visible(False)
ax.axes.yaxis.set_ticklabels(['']);

# Distribution of temperatures across different Europen
```

Countries

```
countries_sorted                                                 =
df_europe.groupby('Country')['AvgTempCelsius_rounded2'].median
().sort_values().index

with sns.color_palette("muted"):
    f, ax = plt.subplots(figsize=(20, 7))
    sns.boxplot(data   =   df_europe,   x   =   'Country',   y   =
'AvgTempCelsius_rounded', order = countries_sorted)
    plt.xticks(rotation = 90)
    plt.title('Distribution  of  Temperatures  in  Europe  (1995-
2019)')
    plt.ylabel('Temperature (°C)')
    plt.xlabel('');

#  Average  temperature  of  the  countries  in  Europe,  sorted  by
temperature in ascending order

countries_mean_sorted                                            =
df_europe.groupby('Country').AvgTempCelsius_rounded2.mean().so
rt_values().index

plt.figure(figsize = (15,8))
sns.barplot(x = 'Country', y = 'AvgTempCelsius_rounded2', data
= df_europe,
            order = countries_mean_sorted)
plt.xticks(rotation = 90)
plt.xlabel('')
plt.title('Average Temperatures in Europe (1995-2019)')
plt.ylabel('Average Temperature (°C)');
```

```
# Grouping the European data by month and year

europe_mean_month         =         df_europe.groupby(['Month',
'Year']).AvgTempCelsius_rounded2.mean()
europe_mean_month = europe_mean_month.reset_index()
europe_mean_month    =    europe_mean_month.sort_values(by    =
['Year'])

# Making a pivot on the data

europe_pivoted = pd.pivot_table(data= europe_mean_month,
                    index='Month',
                    values='AvgTempCelsius_rounded2',
                    columns='Year')

# Plotting a grid to understand the month-wise trends across
1995 to 2019

plt.figure(figsize=(20, 8))
sns.heatmap(data  =  europe_pivoted,  cmap='coolwarm',  annot  =
True, fmt=".1f", annot_kws={'size':11})
plt.ylabel('Month')
plt.xlabel('')
plt.title('Average Temperatures in Europe (°C)')
plt.show();

"""## Prophet"""

# First, we will take backup of the code
dataBackup = df.copy(deep=True)
```

```
# df = dataBackup.copy(deep = True)

df.head()

cols=["Year","Month","Day"]
df['Date']      =      df[cols].apply(lambda      x:      '-
'.join(x.values.astype(str)), axis="columns")

df['AvgTemperatureCelsius'] = df['AvgTempCelsius_rounded']

df = df[df['City'] == 'London']

df  =  df.drop(['Region',  'Country',  'Month',  'Day',  'Year',
'City',      'AvgTempCelsius',      'AvgTempCelsius_rounded',
'AvgTempCelsius_rounded2'], axis = 1)

df = df[df['AvgTemperatureCelsius'] < 50]
df = df[df['AvgTemperatureCelsius'] > -20]

df['ds'] = pd.to_datetime(df['Date'])

df = df.drop('Date', axis = 1)

df = df.rename(columns={'AvgTemperatureCelsius': 'y'})

df.tail()

m = Prophet()
m.fit(df)
```

```
future = m.make_future_dataframe(periods=730, freq = 'd')
future.tail()

forecast_rainfall = m.predict(future)
forecast_rainfall[['ds',         'yhat',         'yhat_lower',
'yhat_upper']].tail()

m.plot(forecast_rainfall)
plt.axvline(x= datetime.date(2020,5,13), color='red');

"""## Pre-processing for ARIMA & LSTM"""

df = dataBackup.copy(deep = True)

# Viewing sample of the data - sanity check
df.sample(5)

# Dropping any duplicates and ensuring that there are no null
values
df = df.drop_duplicates()
df.isnull().sum()

# Typecasting AvgTempCelsius
df['AvgTempCelsius'] = df['AvgTempCelsius'].astype(np.int16)

# Average temperature Global from 1995 to 2019
data = df[['Year','AvgTempCelsius']].groupby('Year').mean()
linfit = np.polyfit(data.index,data['AvgTempCelsius'],deg=1)
linfit = linfit[0]*data.index + linfit[1]

fig = px.line(data,title='Average Temperature of the World from
1995 to 2019')
fig.add_trace(go.Scatter(x=data.index,y=linfit,name='Linear
Fit'))

#Preparing data
data = df[['Region','Month','AvgTempCelsius']]
data = data.groupby(['Region','Month']).mean()

months                                                      =
{1:'Jan',2:'Feb',3:'Mar',4:'Apr',5:'May',6:'Jun',7:'Jul',8:'Au
g',9:'Sep',10:'Oct',11:'Nov',12:"Dec"}
data = data.reset_index(level=1)

#Changing the month label from integer to name
data['Month'] = data.loc[:,'Month'].map(months)
data.head(3)

# Viewing the highest and lowest recorded tempratures in this
dataset
data1=df.sort_values(by=['AvgTempCelsius'],ascending=False).he
ad(1)
data2=df.sort_values(by=['AvgTempCelsius'],ascending=True).hea
d(1)
data = pd.concat([data1,data2],)
data.index = ['Highest','Lowest']
data

# Selecting data points from London
data = df[df['Country'] == 'United Kingdom']
bom            =            data.loc[data['City']           ==
'London',['Month','Day','Year','AvgTempCelsius']].reset_index(
drop=True)
bom.head(3)

# Sanity check to ensure that the temperature is being recorded
every day
size = bom.groupby(['Year','Month']).size().reset_index()
size_max = size[0].max()
size_min = size[0].min()
n = size_max - size_min +1
cmap = sns.color_palette("deep",n)
size = size.pivot(index='Year',columns='Month',values=0)
size.head(3)

# We create the datetime column from month, day and year
bom['Date'] = bom[['Year','Month','Day']].apply(lambda row:'-
'.join([str(row['Year']),str(row['Month']),str(row['Day'])]),a
xis=1)
bom['Date'] = pd.to_datetime(bom['Date'])
bom                                                            =
bom.drop(columns=['Month','Day','Year']).set_index('Date')
bom.head(3)

# Plotting hte average temperature of the city of London
px.line(data_frame=bom,color_discrete_sequence=['blue'],title=
"Daily Average Temperature - London (1995-2019)")

# Plotting the rolling statistics
roll_mean = bom.rolling(window=31).mean()
roll_mean2 = bom.rolling(window=365).mean()
fig = go.Figure()
fig.add_trace(go.Scatter(x=bom.index,y=bom['AvgTempCelsius'],m
arker=dict(color='grey'),name='Daily'))
fig.add_trace(go.Scatter(x=roll_mean.index,y=roll_mean['AvgTem
pCelsius'],marker=dict(color='red'),name='31DaysRolling'))
fig.add_trace(go.Scatter(x=roll_mean2.index,y=roll_mean2['AvgT
empCelsius'],marker=dict(color='green'),name='365DaysRolling')
)
fig.update_layout(dict(title='Rolling Mean'))

# Plotting the rolling standard deviation
roll_mean = bom.rolling(window=31).std()
roll_mean2 = bom.rolling(window=365).std()
fig = go.Figure()
fig.add_trace(go.Scatter(x=roll_mean.index,y=roll_mean['AvgTem
pCelsius'],marker=dict(color='red'),name='31DaysRolling'))
fig.add_trace(go.Scatter(x=roll_mean2.index,y=roll_mean2['AvgT
empCelsius'],marker=dict(color='green'),name='365DaysRolling')
)
fig.update_layout(dict(title='Rolling Std'))

# Performing seasonal decompose on the dataframe
decompose = seasonal_decompose(bom,period=365)
decompose.plot();

# Performing the Dickey-Fuller-test to verify the stationarity
# Here, the the NULL hypothesis is that the time series data is
non-stationary
adf = adfuller(x=bom['AvgTempCelsius'])
print('pvalue:',adf[1])
print('adf:',adf[0])
print('usedlag:',adf[2])
print('nobs:',adf[3])
print('critical_values:',adf[4])
print('icbest:',adf[5])

"""Here, the p-value is very small, and close to 0, so we can
reject the NULL hypothesis.

## LSTM
"""

# Train-test-split
test = bom[bom.index>'2019']
train = bom[bom.index<'2019']

# Performing Min-Max scaling
scaler = MinMaxScaler()
train = scaler.fit_transform(train)
test = scaler.transform(test)

time_steps = 20
features = 1

train_gen                                                            =
TimeseriesGenerator(train,train,time_steps,batch_size=32)
test_gen                                                             =
TimeseriesGenerator(test,test,time_steps,batch_size=32)

# Defining the sequential model and using ReLU as the activation
function
model = Sequential()
model.add(LSTM(64,activation='relu',input_shape=(time_steps,fe
atures),return_sequences=True))
```

```
model.add(LSTM(32,activation='relu'))
model.add(Dense(1,activation='relu'))
model.compile(optimizer='adam',loss='mse')

model.summary()

# Defining the callbacks
early_stop = EarlyStopping(patience=5)

# Fitting the model and setting the parameters
model.fit(x=train_gen,epochs=40,callbacks=[early_stop],validat
ion_data=test_gen)

#Save the model
model.save('LSTMAvgTemp.h5')

# Input the test genertor values to the model and see the
predictions
predict = model.predict(test_gen)

# Collect the original target values from the test generator
test_targets                                                 =
test_gen.targets[test_gen.start_index:test_gen.end_index+1]

# Performing inverse transform on the scaled results
predict = scaler.inverse_transform(predict).ravel()
test_targets = scaler.inverse_transform(test_targets).ravel()

# Evaluation of the model

_,ax=plt.subplots(1,1,figsize=(12,8))
```

```
sns.lineplot(x=range(len(predict)),y=predict,ax=ax,label='Pred
ictions')
sns.lineplot(x=range(len(test_targets)),y=test_targets,ax=ax,l
abel='Actual')
plt.legend()
_=plt.title('Comparison   -   Predictions   vs   Actual   Target
Temperatures - London (2019)')

# Printing the output of the model
print('The                    RMSE                    Score
is:',format(np.sqrt(mean_squared_error(predict,test_targets)),
'.2f'))

# Now, we will forecast beyond 2019

data = bom.iloc[-time_steps:].to_numpy() #2D Array
data = scaler.transform(data)

#expand to include batch dimension
data = np.expand_dims(data,0)

#record the last date of observartion from the data
date = bom.index[-1]

date_store = bom.iloc[-time_steps:].index.to_list()

#forecasting
forecasts=10
for i in range(forecasts):
    predicted = model.predict(data[:,-20:,:])
    date = date+datetime.timedelta(days=1)
    data = np.append(data,[predicted],axis=1)
    date_store.append(date)
data = scaler.inverse_transform(data.reshape(1,-1))
forecast_df     =     pd.DataFrame(index=date_store[time_steps-
1:],data={'AvgTempCelsius':data.ravel()[time_steps-1:]})

# Let us look at the table of the forecast (Sanity check)
forecast_df

# Plotting the output
_,ax=plt.subplots(1,1,figsize=(12,8))
sns.lineplot(data=bom.iloc[-
100:,:],y='AvgTempCelsius',x=bom.iloc[-
100:,:].index,color='blue',ax=ax,label='AvgTemp - 2019 end')
sns.lineplot(data=forecast_df,y='AvgTempCelsius',x=forecast_df
.index,color='red',ax=ax,label= 'AvgTemp - 2020 forecasted')
_=plt.title(f'Temperature   Forecasting   -   {forecasts}   days
(2020)')

"""## Seasonal ARIMA Model"""

# We are downsampling to monthly frequency
bom_monthly = bom.resample('M').mean()
bom_monthly.head(5)

# Plotting the output - the seasonal element is 12 months
data
=[go.Scatter(x=bom.index,y=bom['AvgTempCelsius'],name='AvTempe
rature-Daily',marker=dict(color='grey')),go.Scatter(x=
bom_monthly.index,y=bom_monthly['AvgTempCelsius'],name='AvgTem
pCelsius-Monthly',marker=dict(color='red'))]
```

```
fig = go.Figure(data)

buttons                                                      =
[dict(label='Both',method='restyle',args=[{'visible':[True,Tru
e]}]),

dict(label='Daily',method='restyle',args=[{'visible':[True,Fal
se]}]),

dict(label='Monthly',method='restyle',args=[{'visible':[False,
True]}])]

updatemenus=[dict(type="buttons",direction='down',buttons=butt
ons)]
fig.update_layout(updatemenus=updatemenus,title='Average
Temperature - Daily and Monthly - London')

# Re-sampling the data
bom_monthly = bom.resample('M').mean()
bom_monthly.head(5)

# Plotting the average temperature to compare monthly and daily
data
=[go.Scatter(x=bom.index,y=bom['AvgTempCelsius'],name='AvTempe
rature-Daily',marker=dict(color='grey')),go.Scatter(x=
bom_monthly.index,y=bom_monthly['AvgTempCelsius'],name='AvgTem
pCelsius-Monthly',marker=dict(color='red'))]
fig = go.Figure(data)

buttons                                                      =
[dict(label='Both',method='restyle',args=[{'visible':[True,Tru
e]}]),

dict(label='Daily',method='restyle',args=[{'visible':[True,Fal
se]}]),

dict(label='Monthly',method='restyle',args=[{'visible':[False,
True]}])]

updatemenus=[dict(type="buttons",direction='down',buttons=butt
ons)]
fig.update_layout(updatemenus=updatemenus,title='Average
Temperature - Daily and Monthly - London')

"""## Auto-ARIMA"""

import statsmodels.api as sm

#Using default auto_arima arguments
model = auto_arima(bom_monthly,seasonal=True,m=12)

# Storing date and temperature
date = bom_monthly.index[-1]
last_val = bom_monthly.iloc[-1].to_numpy()

# Forecasting the first 5 months
forecasts=5
date_store  =  [(date  +  relativedelta(months=i))  for  i  in
range(0,forecasts+1)]

predict = model.predict(forecasts)
predict =  np.append(last_val,predict)
```

```
forecast_df                                                       =
pd.DataFrame(data=predict,index=date_store,columns=['AvgTempCe
lsius'])
forecast_df.head(5)

# Plotting the results
_,ax=plt.subplots(1,1,figsize=(10,10))
sns.lineplot(data=forecast_df['AvgTempCelsius'],ax=ax,color='r
ed')
sns.lineplot(data=bom_monthly['AvgTempCelsius'][-
4*12:],ax=ax,color='blue')
_=plt.title(f'Forecast - Monthly Temperature Average (2020) -
First {forecasts} months')
```