UNIVERSITY OF BELGRADE

FACULTY OF MATHEMATICS

Kwasi Appiah Takyi

PREDICTION OF POPULATION SIZES OF AFRICAN COUNTRIES BY RECURRENT NEURAL NETWORK

Master thesis

Belgrade, 2024

Mentor:

Prof. Mladen Nikolić
Faculty of Mathematics, University of Belgrade.

Members of Commission:

Prof. Zorica Stanimirović
Faculty of Mathematics, University of Belgrade.

Prof. Aleksandra Delić
Faculty of Mathematics, University of Belgrade.

Date of defence :_____

Contents

Declaration

I hereby declare that this submission is my own work towards the award of the Master of Science degree and that, to the best of my knowledge, it contains no material previously published by another person nor material which had been accepted for the award of any other degree of the university, except where due acknowledgement had been made in the text.

Acknowledgement

This thesis could not have materialized without the divine protection and guidance of the Almighty God. First and foremost, I am immensely grateful to the government of the Republic of Serbia and Ghana, which provided me with the unique opportunity to pursue my master's degree at the University of Belgrade, Faculty of Mathematics under the "World in Serbia" scholarship program. This collaboration has not only broadened my academic horizons but has also allowed me to experience the rich culture and hospitality of Serbia.

My profound gratitude to the Serbian ministry of education, science and technology and other ministries for their invaluable support during my master's studies.

I extend my heartfelt appreciation to the Faculty of Mathematics and staff of the Department for Numerical Mathematics and Optimization for providing a conducive and intellectually stimulating environment for my studies. The resources and facilities made available to me have significantly contributed to my research.

I am profoundly thankful to my lecturers Prof. Zorica Stanimirović (Head of Department), Prof. Milan Dražić, Prof. Sandra Živanović, and Prof. Aleksandra Delić at the University of Belgrade for their invaluable guidance, support, and encouragement. Their expertise and insights have been instrumental in shaping this thesis. Special thanks go to my mentor Prof. Mladen Nikolić, whose continuous support and constructive feedback have been pivotal in the successful completion of this work. I am grateful to my lovely wife Mrs. Ruby Baidoo and daughter Jenaiah Appiah Takyi, and other family members for being with me from the beginning of this academic journey. Lastly, I would like to thank my fellow students and colleagues at the University of Belgrade for their camaraderie, shared experiences, and the collaborative spirit that made this journey memorable.

This thesis would not have been possible without the collective support and contributions of all these individuals and institutions. Thank you all for being an integral part of this academic endeavour.

Abstract

Population growth forecasting shows the future mortality and fertility rate, as well as migration of people from one country to another. This is really vital to the health system of every country, especially the ones in Africa. The rate at which the population of Africa is increasing, there is the need to pay active attention to it. Recently, machine learning concepts are most growing and popular for predicting future values. In order to predict population growth, the machine learning models are applied to a pool of indicators that affect population growth. There was an issue of missing values in the respective data, hence three imputation method were used. Namely, mean, linear interpolation, and linear interpolation with additional indicators. The thesis investigates the population growth of 39 selected African countries using time series forecasting machine learning technique recurrent neural network (RNN). The data optimum prediction method is based on the method that gives lowest error rate through the use of the RNN model. The Python programming language was used because it is user friendly and it showed to be successful when solving most mathematical and statistical learning problems. The libraries Pandas, Numpy, Scikit-Learn, Keras were utilized.

Keywords: Machine Learning, Recurrent Neural Network, Python, Time series forecasting, Population, Feed Forward Neural Network, Data Imputation, Linear Interpolation, Mean, Indicators.

Chapter 1

Introduction

1.1 Importance of population on earth

Population development is a significant factor in understanding numerous angles of human society and the environment. Its significance can be assessed from a few viewpoints, including financial, social, and political measurements. Here are a few key focuses outlining the significance of population growth in the world. A developing population can give a greater labour drive, which can contribute to financial development. More labourers can lead to expanded generation and utilization, cultivating financial advancement. This can increase product and service delivery, driving development and entrepreneurship. Populace development frequently leads to urbanization, which requires the improvement of framework such as lodging, transportation, and open administrations. This can drive improvements in living guidelines and quality of life. A developing populace can improve social differences, cultivating imagination, advancement, and social dynamism. Overseeing population development is basic for arranging and giving satisfactory instruction and healthcare administrations. Quick development can strain these frameworks, whereas well-managed development can lead to enhancements in human capital. On the other hand, population growth increases the talent pool and human capital. Humans have a distinctive life history even among our closest living relatives: compared to other primates, we endure a long adolescent period of intensive growth, extreme dependency and learning; we mature and reproduce late; we have long reproductive careers with short inter-birth intervals; and we enjoy a long post-reproductive lifespan [**1**]. It necessitates improvements in infrastructure and public services, prompting governments to invest in beneficial sectors, such as healthcare, transportation, and education. This can lead to higher living standards and improved quality of life of citizens. Moreover, a youthful population can drive societal progress and adaptability, ensuring a steady supply of future leaders and innovators. While population growth brings numerous benefits, it also poses important environmental problems on societies.

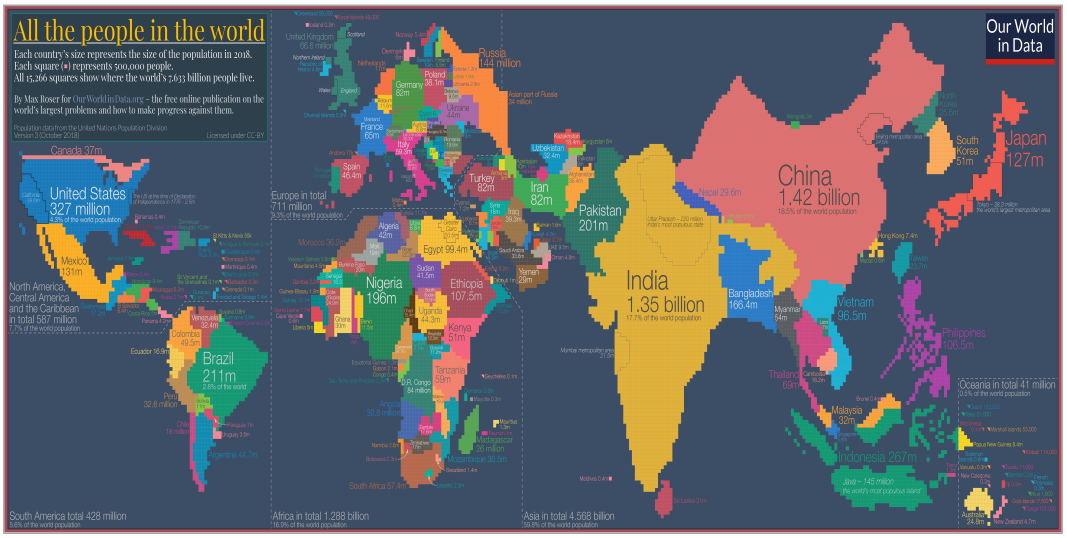

Figure 1.1: A map showing the world's population. Source: Max Roser (2018), `https://ourworldindata.org/world-population-cartogram`.

1.2 Problems of some African countries

African countries encounter various challenges that stem from a complex interplay of historical, social, economic, environment, and political factors. The fastest-growing population in the world is found in Africa. Figure 1.1. shows the world population as at 2018 [**2**]. If we compare the population data of 2018 and 2022, it is evidence that some African countries like Democratic Republic of Congo, Ethiopia, and Nigeria are the countries driving this expansions. The youthful demographic of the Africa population, with a median age of approximately 19.5 years [**3**], in contrast to the global median age of 30.6 years as at 2024 [**4**], is a significant contributing element. The trend of rapid urbanisation is also noteworthy. Cities like Kinshasa, Nairobi, Lagos, and other quickly growing cities are occasionally growing faster than the infrastructure required to support this growth [**5**]. The migration from rural to urban areas in search of better living and work opportunities is what fuels urbanization. Among these issues are governance and political instability. Corruption remains a known issue, destroying development and public trust in government institutions. Many African countries have damaged political institutions, which led to poor governance and ineffective public administration. In recent times conflicts, civil wars, and insurgencies disrupt social and economic instability, with regions like parts of Nigeria, Ethiopia, Sahel and Somalia being particularly affected.

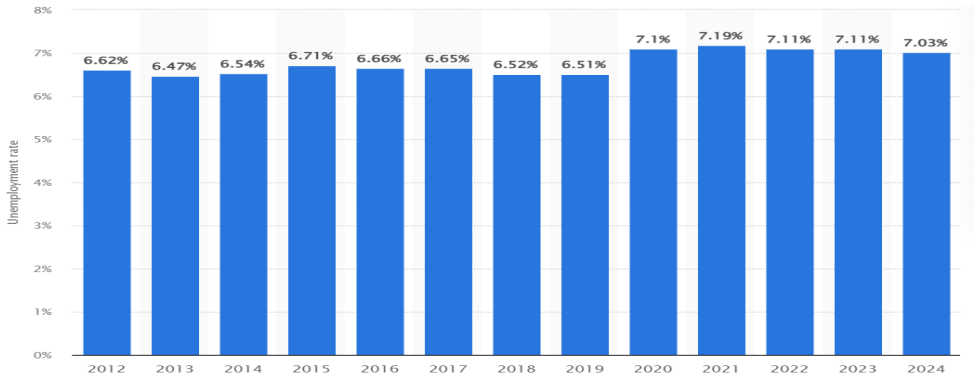

Figure 1.2: Unemployment rate in Africa from 2012 - 2024 by Statista.

A large proportion of the population in many African countries lives in poverty, with limited access to basic needs such as shelter, clean and safe water for drinking and food. Figure 1.2. shows the unemployment rate in Africa [**6**]. There is a high level of unemployment, specifically among the youth, which can lead to economic stagnation and social tension. Many African economies rely heavily on the export of primary commodities, which is exposing them to global price variations. Also considering some social related issues such as healthcare and education, healthcare administration is not well organised, leading to high rates of disease and low life expectancy.

Diving into environmental challenges caused by high population growth, Africa can not be excluded. Africa is commonly portrayed as the driver of a global "population explosion" [**7**]. This rapid growth is putting pressure on an already strained water and sanitation infrastructure, with municipalities struggling to make improvements toward water and sanitation targets outlined in the Sustainable Development Goals [**8**].
Challenges include high fertility rates, varying migration patterns, and diverse economic conditions, which complicate accurate population forecasting. Reliable predictions are crucial for managing resources and planning for future development.

Figure 1.3: Illegal mining in some part of Ghana has polluted many water bodies and caused land degradation due to increasing population and high job demands. Source: *Emmanuel Ofosu-Mensah Ababio. Historical and Modern Artisanal Small-Scale Mining in Akyem Abuakwa, Ghana*

Success requires cooperation between the public and business sectors, foreign partners, and civil society. Below are some few suggestions to help mitigate the problems of increasing population:

1. **Economic development:** This could be done by diversifying the economy to lessen reliance on a small number of items. Making investments in vital infrastructure, such as energy, ports, and highways. Giving support to Small and Medium Enterprises (SMEs) by having access to markets and financing. The countries must encourage the integration of regions and draw in foreign direct investment (FDI).

2. **Healthcare:** Africa's undeveloped healthcare systems require creative thinking and unconventional solutions to overcome the existing service delivery deadlock. Public-private projects, for instance, have to be pursued, in which multinational corporations that take resources from Africa could be persuaded to reinvest a portion of their revenues on healthcare for the populations that supply their labour force. The majority of issues and their fixes are related to management, budgeting, and human resources. Setting these as a top priority is crucial to enhancing health outcomes.

3. **Governance and political stability:** This is another source of some of the issues that Africa is currently facing. Government agencies should be well-established, transparent, and accountable in order to lessen it. Furthermore; establishing strict anti-corruption guidelines and practices, promoting democratic processes, and upholding the law are essential. Numerous instances suggest that the majority of laws support particular groups of people.

4. **Education and skills development:** The majority of African nations continue to use antiquated curricula; therefore, we must improve our curriculum and incorporate digital literacy and critical thinking to modernise our educational system. A developed skill set that can meet the demands of the market today is required.

5. **Agriculture and food security:** Leaders in Africa must advocate for climate-resilient and sustainable farming practices, and give smallholder farmers access to markets, technology, and funding. In addiction, one should put in place initiatives to guarantee food security and lessen hunger.

6. **Environmental sustainability:** Climate change is wreaking havoc on development in all its forms. Therefore, minimising the harmful effects of climate change on our ecosystem can be achieved by developing ways to address its implications. Preserving biodiversity and natural resources is a very important point to consider. For instance, reducing the illegal mining that occurs in certain nations, such as Nigeria, Ghana, and South Africa. Additionally, investing in renewable energy sources will help reduce reliance on fossil fuels.

7. **Technology and innovations:** It is necessary to make investments in digital infrastructure and guarantee that technology is accessible and affordable. Establishing an atmosphere that promotes creativity and entrepreneurship. Additionally, improving governance and service delivery by utilising ICT more effectively.

1.3 Statement of problem

The growth in the size of the world population is often considered a major challenge for the sustainable development of the world economy and the maintenance or improvement of the living conditions of people around the world. Population growth is currently the most pronounced in African countries, and it calls for special attention. Hence, accurate prediction of population sizes is crucial for effective planning and resource allocation in African countries, where rapid demographic changes and data collection challenges are prevalent. Traditional statistical methods often fall short in capturing the complex, non-linear patterns inherent in population growth. Additionally, the presence of missing values in demographic datasets further complicates the prediction process, leading to inaccurate estimates and suboptimal policy decisions. Therefore, there is a critical need for advanced methodologies that can handle incomplete data and provide reliable population forecasts.

1.4 Objectives

As the population size changes over time, it can be considered a time series. Machine learning methods appropriate to time series and sequential data can be used to model it. This study's main goal is to create and assess a Recurrent Neural Network (RNN) model that can accurately forecast the population sizes of African nations. The purpose of this study is to:

- Examine how successful RNNs are in representing the non-linear patterns and temporal dependencies found in population data.
- Address the challenge of missing values in demographic datasets by employing imputation techniques and incorporating mechanisms within the RNN architecture to handle incomplete data.
- Compare the performance of the RNN model on various imputation method on the dataset to highlight the advantages and potential improvements offered by machine learning approaches.
- Provide reliable population forecasts that can assist policymakers and stakeholders in making informed decisions for sustainable development and resource allocation in African countries.
- Analyse the outcome result and to identify the best model and imputation method for each country.

1.5 Good predictive performance

Forecasting population sizes for African nations with recurrent neural networks (RNNs) may be complicated because of various influencing factors like birth rate, GDP growth, death rates, economic factors, migration, and governmental policies. The predictive accuracy may differ based on the dataset and circumstances, but standard quality measures like mean squared error (MSE), R-squared (R^2), and mean absolute error (MAE) will be used to assess models effectively.

Preliminary results indicate that the RNN model achieves good predictive performance on the mean imputation method, accurately forecasting population sizes with lower error rates compared to other imputation methods used in the experiment.

1.6 Structure of the thesis

This thesis has six chapters. Chapter 1 presents the motivation of the study. Chapter 2 will focus on the machine learning and neural networks, taking into consideration the basic concepts of ML (model, loss, loss minimization, gradient descent), basics of evaluation (training and test set, root mean square error, absolute error), feed forward neural networks and recurrent neural networks. Chapter 3 will discuss data imputation. Missing data and the concept of data imputation, and basic methods of data imputation will briefly be discussed. Chapter 4 will present the prediction of population size. And the description of predictive task, dataset description, model description, quality metrics and implementation (code snippets). Chapter 5 will highlight experimental results. Considering report on basic metrics and per-country analysis. Finally, Chapter 6 will draw the conclusions.

Chapter 2

MACHINE LEARNING AND NEURAL NETWORKS

Machine learning (ML) can be define as a process of building computer systems that automatically improve with experience, and implement a learning process [**9**]. In contrast to traditional programming, which requires a programmer to explicitly set the rules and logic, machine learning algorithms enable the system to derive patterns and insights from the data it receives. Machine learning can be categorised into three main types, namely supervised, unsupervised, and reinforcement learning. In this chapter, we will be focusing on some aspect of ML.

2.1 Basic concept of machine learning

Machine learning is continuously growing in the IT world and gaining strength in different business sectors. Although machine learning is in the developing phase, it is popular among various technologies. It is a field of study that makes computers capable of automatically learning and improving from experience. Hence, machine learning focuses on the strength of computer programs with the help of collecting data from various observations. Major roles in machine learning are performed by some concepts like model, loss, loss minimization, and gradient descent. We shall briefly explain some of the relevant concept of ML in the subsections [**10**].

2.1.1 Model

Machine learning model is a mathematical representation of real-world problems. It is a type of mathematical model $f(X;w)$ that, after being "trained" on a given dataset X, can be used to make predictions or classifications on new data. Examples are linear regression, neural network, decision tree, etc. The main component of ML models are:

- **Parameters (w):** These are the numbers that the model learns from the training data. For instance, in a linear regression model, these parameters are the coefficients which multiply the variables.

- **Hyperparameters:** They are the external configurations set before training the model, like the learning rate in gradient descent or the number of layers in a neural network.

2.1.2 Loss

The loss is a function $\mathscr{L}(y, f(X;w))$ that measures how well the models predictions match the true values y. It quantifies the error between the predicted output and the actual output; and thus measures the network's performance in numerical terms [**11**]. The choice of a loss function depends on various factors, including the nature of the problem, the distribution of the data, and the desired characteristics of the model. Different loss functions emphasize different aspects of model performance and may be more suitable for specific applications. Common loss functions include:

- Mean squared error : MSE is a fundamental metric in the realm of machine learning, particularly in the domain of regression analysis. The mean squared error is a crucial metric for evaluating the performance of predictive models. The mathematical formulation is given below as:

$$\mathscr{L}(y, f(X;w)) = \frac{1}{k}\sum_{i=1}^{k}(y_i - f(X_i;w))^2. \tag{2.1}$$

- Cross-entropy loss : It is used in classification tasks, and it can be computed mathematically as:

$$\mathscr{L}(y, f(X;w)) = -\frac{1}{k}\sum_{i=1}^{k}[y_i \log f(X_i;w) + (1-y_i)\log(1-f(X_i;w))]. \tag{2.2}$$

2.1.3 Loss minimization

Loss minimization is the process of changing the model's parameters w to reduce the loss function. The objective is to optimise the model's performance by identifying the combination of parameters that provide the least amount of loss. This process is also known as optimization. It is defined as:

$$w^* = arg\min_{w} \mathscr{L}(y, f(X;w)). \tag{2.3}$$

2.1.4 Gradient descent

Gradient descent is known as one of the most commonly used optimization algorithms to train machine learning models by means of minimizing errors between actual and expected results. Again, gradient descent is also used to train neural networks. The higher the gradient, the steeper the slope. Notices that, if the slope is zero, the model stops learning. Gradient descent can be formulated mathematically as:

$$w_{i+1} = w_i - k\frac{\partial L}{\partial w} \quad , \qquad i = 1,2,3,\ldots \tag{2.4}$$

w_{i+1} is the next expected position of the model parameter, w_i is the previous position, k is the learning rate, and $\frac{\partial L}{\partial w}$ is the gradient of the loss function with respect to w. Figure 2.1 gives an overview of gradient descent [**12**].

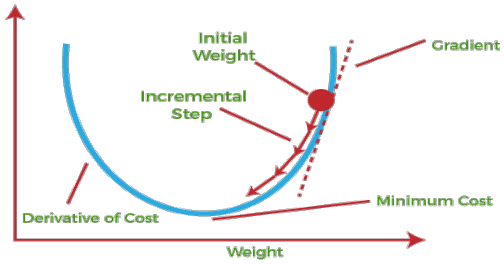

Figure 2.1: Gradient descent in machine learning. Source: `https://www.javatpoint.com/gradient-descent-in-machine-learning`

2.2 Evaluation of machine learning models

Model evaluation is a process that uses certain measures to help analyse model performance [**13**]. The evaluation also helps to analyse the main weaknesses of the model. There are various metrics such as precision, accuracy, recall, F1 score, area under the curve, confusion matrix, mean absolute error and root mean square error. But in this work, we shall be focusing on mean absolute error, R-squared, and root mean square error.

- **Training set:** [**14**] Training set refers to the dataset used in training a machine learning model. It includes the input and the corresponding expected output, where the inputs are the features or attributes that the model will use to make predictions, and the outputs are the target values or labels that the model is expected to predict. The model learns from this data by finding patterns and relationships between the inputs and outputs. Type of training data hugely determines the ability of the model to generalize. Note that it is very important to allocate a higher percentage of the data to the training set.

- **Validation set :** Validation set is a portion of the data from the training set often set aside as to tune the model's hyperparameters and prevent over-fitting. In machine learning, a validation set is a subset of the dataset used to evaluate the performance of a model during training. The primary purpose of the validation set is to provide an unbiased evaluation of the model while tuning its hyperparameters, thus helping to prevent over-fitting and ensure the model's generalizability. When comparing multiple models, validation helps in selecting the best model. The one that performs best on the validation set is usually chosen as the final model.

- **Test set:** Once a machine learning model is built (with training data), one needs unseen data to test the model. This data is called testing data. It is used to evaluate the performance of the machine learning model. After training, the model makes predictions on the test set, and these predictions are compared to the actual outputs to assess the model's performance. Figure 2.2 is an example of how data is split in data preprocessing for machine learning.

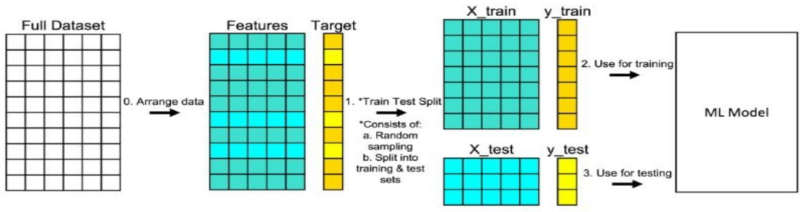

Figure 2.2: Image showing the splitting procedure in ML. Source: `https://builtin.com/data-science/train-test-split`

- **Root mean squared error (RMSE):** When we talk about RMSE in machine learning, we are essentially addressing its role as a performance measure for algorithms that involve prediction or forecasting. It provides an estimate of how far the predicted values $\hat{y}_i$ deviate on average, from the actual values y_i in the dataset. It is valuable because it retains the same units as the input, making it easier to interpret. The RMSE is given by:

$$RMSE = \sqrt{\frac{1}{k}\sum_{i=1}^{k}(y_i - \hat{y}_i)^2} \tag{2.5}$$

- **Mean absolute error (MAE):** MAE is the average of the absolute differences between the predicted values and the actual values. MAE takes the average of absolute errors for a group of predictions and observations as a measurement of the magnitude of errors for

the entire group. MAE can also be referred as L1 loss function. It is often considered more interpretable because it represents the average error in the same units as the original data. The mean absolute error is mathematically given as:

$$MAE = \frac{1}{k}\sum_{i=1}^{k}|y_i - \hat{y}_i| \tag{2.6}$$

- **R-Squared :** The R^2 measures the proportion of the variance in the dependent variable that is predictable from the independent variables. A good R^2 values for population prediction, close to 1 indicates that a large proportion of the variance is explained by the model. Below is the formulation of R-squared:

$$SSR = \sum_{i=1}^{k}(y_i - \hat{y}_i)^2 \tag{2.7}$$

$$SS = \sum_{i=1}^{k}(y_i - \bar{y})^2, \tag{2.8}$$

where $\bar{y}$ is the mean of the true values,

$$R^2 = 1 - \frac{SSR}{SS}. \tag{2.9}$$

2.3 Feed forward neural network

One of the most basic kinds of artificial neural networks (ANNs) is a feedforward neural network (FNN), sometimes referred to as a feedforward network. There are no cycles in the connections made between the nodes in this network. It is called "feed forward" because the information moves in only one direction, i.e., forward from the input nodes, through the hidden nodes (if any) to the output nodes. This is in contrast to recurrent neural networks, where connections between nodes can create cycles and allow information to be fed-back into the network.

2.3.1 Structure of feed forward neural network

Below is a detailed breakdown of the structure of the feed forward neural network:

1. **Layers :** Feed forward network has two main types of layers, each will be mentioned and explain briefly below:

- Hidden layer : the layers that lie before the output layers are known as hidden layers. The nodes (neurons) that make up each hidden layer weighs the inputs and then routes the outputs through an activation function. These layers are in charge of identifying intricate patterns in the data. The basic units that process inputs and produce outputs are called nodes, or neurons. In a hidden layer, every node adds up all of its inputs, weighs it, and then runs the output through an activation function.
- Output layer : the output layer produces the final output of the network. The number of neurons in the output layer depends on the type of problem being solved (e.g., one neuron for binary classification, multiple neurons for multi-class classification, or regression).

2. **Activation function :** Activation functions introduce non-linearity into the network, enabling it to learn complex patterns. Common activation functions include:
   - Rectified linear units (ReLU): $y(t) = max(0,t)$
   - Sigmoid : $y(t) = \dfrac{1}{1+e^{-t}}$
   - Hyperbolic tangent (tanh) : $y(t) = \tanh(t) = \dfrac{e^{t}-e^{-t}}{e^{t}+e^{-t}}$
   - Leaky ReLU : $y(t) = max(\alpha t, t)$, where α is a small positive constant and t is the input to the neurons.

2.3.2 Forward propagation

In forward propagation, the input data passes through the network layer by layer until it reaches the output layer. Each neuron's output in one layer becomes the input for the neurons in the next layer. The forward pass computes the predicted output of the network. The forward flow of data is designed to avoid data moving in a circular motion, which does not generate an output. Pre-activation and activation occur at every hidden and output layer node of a neural network during forward propagation. The weighted sum calculation is the pre-activation function. By adding the bias to the weighted sum and then applying the activation function, a non-linear relationships is created in the data.

2.3.3 Backpropagation and training

[**15**] Backpropagation is an algorithm specifically designed to compute the partial derivatives of the loss function with respect to the weights of a neural network, essentially calculating the

gradient. The gradients obtained through backpropagation are utilized in gradient descent to optimize the loss function, thereby adjusting the weights and improving the model's performance. This process is crucial for training neural networks efficiently.

2.4 Recurrent neural network

An artificial neural network that functions best with data that arrives in a specific order is called a recurrent neural network (RNN). RNNs are helpful for tasks like voice recognition, language translation, time series, and captioning photos. This is due to their ability to convert input sequences into output sequences through processing. The "memory" that RNNs possess is one unique feature. They are able to retain data from earlier inputs in the present processing step as a result. Since a large portion of the data in the world is sequential, extra consideration must be given while developing predictive models. [**16**]

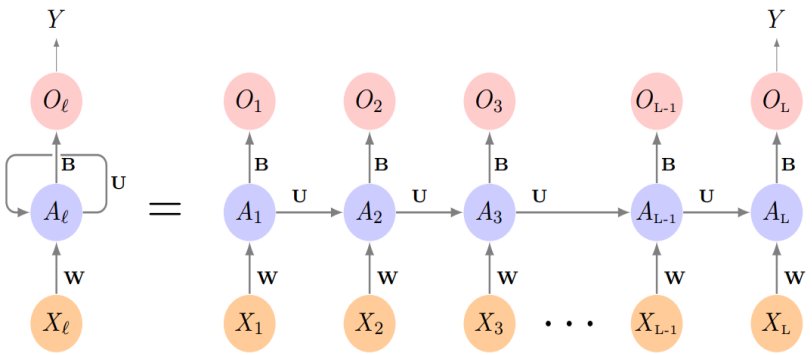

Figure 2.3: Diagram illustrating the basic recurrent neural network. Source: *Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. An Introduction to Statistical Learning with Applications in R. Second edition, page 422.*

Figure 2.3 shows the structure of the basic recurrent neural network. As it can be seen from the figure, there are a series of vectors in the input $\{x_l\}_1^l$. The input sequence X is processed sequentially by the network; each X_l flows into the hidden layer, which additionally receives the activation vector A_{l-1} from the element before it as input. Every element of the sequence is processed using the same sets of weights W, U, and B. From the current activation A_l, the output layer generates a series of predictions O_l; usually, only the last of these, O_L, is significant. The network is represented succinctly to the left of the equal sign, and is unrolled into a more explicit version to the right [**17**].

2.4.1 Key concepts of RNN

1. Sequential data : RNNs work especially well in applications involving sequential data. RNNs are able to retain information over many time steps because they feature connections that loop back on themselves, in contrast to feedforward neural networks. For instance, we will be working with sequential data from 39 African nations in this work.

2. Hidden state : An RNN's hidden state serves as a kind of memory that stores data about previous processing operations. Based on the prior hidden state and the current input, the hidden state is updated at each time step.

3. Weight sharing : The weights in an RNN are shared across all time steps. This is what enables RNNs to generalize well to sequence data of varying lengths.

2.4.2 Architecture of recurrent neural network

RNNs have same input and output architecture as any other deep neural architecture. However, difference arise in the way information flows from input to output. Unlike deep neural network where we have different weight matrices for each dense network in RNN, the weight across the network remains the same. Figure 2.3 represents the structure of a basic RNN with sequence $X = X_1, X_2, X_3, \ldots, X_L$ as input, a simple output Y , and a hidden-layer sequence $\{A_l\}_1^L = A_1, A_2, A_3, \ldots, A_L$. As the sequence is processed one vector X_l at a time, the network updates the activations A_l in the hidden layer, taking as input the vector X_l and the activation vector A_{l-1} from the previous step in the sequence. Each A_l feeds into the output layer and produces a prediction O_l for Y. Assume that the hidden layer is made up of K units $A_l^T = \{A_{l1}, A_{l2}, A_{l3}, \ldots, A_{lk}\}$ and that each vector X_l of the input sequence has p components $X_l^T = \{X_{l1}, X_{l2}, \ldots, X_{lp}\}$. In Figure 2.3, for the input layer, we use a matrix W to represent the collection of $K \times$ (p+1) shared weights w_{kj}. In a similar manner, U is a K×K matrix that contains the weights u_{ks} of the hidden-to-hidden layers, and B is a K + 1 vector representing the output layer's weights β_k. Below is a basic equations governing an RNN,

$$A_{lk} = g(w_{k0} + \sum_{j=1}^{p} w_{kj} X_{lj} + \sum_{s=1}^{K} U_{ks} A_{L-1,s}). \tag{2.10}$$

The output O_l is computed as:

$$O_l = \beta_0 + \sum_{k=1}^{K} \beta_k A_{lk}. \tag{2.11}$$

2.4.3 Characteristics of simple RNNs

1. Memory of past inputs : By maintaining a hidden state, RNNs can remember information from previous time steps, making them suitable for sequential data.

2. Challenges with long sequences : Simple RNNs can struggle with long-term dependencies due to issues like vanishing gradients, where the influence of older inputs diminishes exponentially as the sequence length increases.

Chapter 3

DATA IMPUTATION

Missing data is a prevalent problem in data analysis and machine learning that can have a big influence on model performance and outcomes. The techniques of substituting values for missing values in a dataset are known as data imputation. This chapter will examine the idea of missing data, the reasons for it, and the fundamental techniques for handling it with data imputation.

3.1 Missing data and the concept of data imputation

The statement "without data, there is no machine learning" is one unquestionable truth. Finding a trustworthy and accurate source of population data as well as other variables that influence a country's population is necessary, because the foundation of this thesis is population forecast. The World Bank and United Nation are trustworthy source from which the data was gathered.

3.1.1 Understanding missing data

Missing data, also known as missing values, occurs when no data value is stored for a variable in an observation. Missing data is one serious situation that can affect the performance of a model. It can happen for various reasons, including: data loss, human error, and non-response. Data imputation is the technique used to fill in the missing values in a dataset. The goal is to provide a complete dataset that can be used for analysis without the biases or inaccuracies that can arise from simply ignoring missing data. Imputation methods aim to maintain the statistical properties of the original data.

The quantity and kind of missing data, the type of data, and the particular needs of the study all influence the imputation method selection. Data imputation can be done in a number of ways, from straightforward procedures to intricate algorithms.

3.2 Basic methods of data imputation

One of the simplest methods of imputation involves replacing missing values with the mean, median, or mode of the available data [**18**]. There are others like; linear interpolation, k-nearest neighbour (KNN), last observation carried forward (LOCF), next observation carried backward (NOCB) and regression imputation. We shall be focusing on mean and linear interpolation as far as this thesis is concern.

3.2.1 Linear interpolation

There are a number of common methods that are routinely used to help mitigate the problem of missing data. One is linear interpolation. It is commonly used to fill in missing values in time series data. It refers to interpolating or predicting missing data points using a linear function that best fits the known data. It can be computed mathematically as:

$$y = y_1 + (x - x_1)\frac{(y_2 - y_1)}{(x_2 - x_1)} \tag{3.1}$$

Where (x_1, y_1) are coordinate of the first data points, (x_2, y_2) are coordinates of the second data point, and x is the point on which interpolation is performed and y is the interpolated value.

3.2.2 Mean

Mean imputation is a technique used in statistics and data analysis to handle missing data within a dataset. When a dataset has missing values, mean imputation fills in these gaps with the mean (average) value of the non-missing data for that particular variable. In this thesis, the dataset is split into training, validation and test set according the percentage 80%, 10% and 10% respectively; the mean from each column (feature) of the train data was calculated to fill the missing values in both the validation and test set. A simple mathematical formulation is given below:

$$Mean(\bar{x}) = \frac{x_1 + x_2 + \cdots + x_n}{n}, \tag{3.2}$$

where $x_1, x_2, \ldots, x_n$ are data points or observed values.
Some of the merits of this imputation method is:

- It is easy to understand.

- Easy to implement and also keeps the number of records the same, thereby avoiding the loss of data.

And some of the demerits are:

- It reduces the variability in the dataset, as the imputed values are the same for each missing value within a variable.
- correlations can be distorted between variables, potentially leading to incorrect conclusions.

We shall discuss it a bit more in the next chapter.

3.2.3 Indicators

An indicator is a binary variable (taking values of 0 or 1) that signifies the presence or absence of a specific values or attributes within a dataset. Additional columns corresponding to the number of features in our data were created to indicate 0's for non-missing values and 1 otherwise. Indicators are used for various purposes, including:

- Preserving information: indicators preserves the information about missing data by labelling missing values with an indicator column. This information may be crucial for the model to understand missingness-related patterns or correlations.
- Preventing data loss: instead of discarding rows with missing values, creating an indicator column allows us to use the available data more effectively. This is particularly important when the dataset is small or the missing data is substantial.

Chapter 4

PREDICTION OF POPULATION SIZE

This chapter provides an overview of the importance of population prediction, description of predictive task, dataset description, model description, quality metrics and implementation of code.

With the recent advancement in artificial intelligence, especially recurrent neural networks (RNNs), and the growing availability of open-source data, it is possible to build better predictions for population future sizes at a disaggregate geographical level. These predictions can help solve various real-world problems, such as infrastructure and residential planning and political boundary delimitation studies. The proposed project aims to deal with estimate building-based population prediction by making use of a data-driven approach and cutting-edge artificial intelligence, the recurrent neural network (RNN). This method is simple, straightforward, and promising. However, the data errors and assumptions built into the method may affect the results.

Population prediction is a difficult problem since the population can change due to a variety of factors such as births, deaths, and people moving in and out of the area. Planning a range of services, such as power, water, transport, schools, and medical centers if the population exceeds a certain size, can be costly. Conversely, the building of facilities such as services and infrastructure is expensive, and it would be a waste of resources if the facilities that are provided are not used. Therefore, a better estimation of the population would result in better community planning and development projects.

Although all kinds of predictive modeling have been employed for these attempts, the recent advances in the deep learning field, particularly regarding recurrent neural networks, or RNNs, with their in-built feedback mechanisms for the process of predictions, have shown the capability to drive this development to a level surpassing what has been achieved before, achieving goals that resolve some setbacks with conventional approaches.

4.1 Description of predictive task

The predictive task involves forecasting the population size of a given region over a specified period. The prediction model will take into account historical population data along with other relevant demographic and socio-economic factors. The primary objective is to minimize the error between the predicted population size and the actual observed values.
Some specific goals of predictive task are:

- Creating a predictive model that can manage big datasets and generate precise projections.
- Determining the main causes of population shifts.
- Evaluating the model's performance using standard quality metrics.

4.2 Dataset description

The dataset used for this study comprises historical population data along with various demographic and socio-economic indicators. The data is sourced from international organizations such as the United Nations and World Bank. These are annual data (63 data points per country) covering the years 1960 to 2022. 39 African nations from the region's West, East, South, North, and Central are taken into consideration. The lack of availability and sufficient data points for this endeavour was the reason why certain African countries were left out. Furthermore, it is well known that large datasets are necessary for machine learning models to learn and develop accurate patterns for predictions. Countries considered in this studies are Algeria, Angola, Benin, Burkina Faso, Burundi, Cameroon, Chad, Congo Republic, Cote D'voire, Democratic Republic of Congo, Egypt, Equatorial Guinea, Eswatini, Gabon, Gambia, Ghana, Guinea-Bissau, Kenya, Lesotho, Lybia, Mali, Morocco, Mozambique, Namibia, Niger, Nigeria, Rwanda, Senegal, Sierra Leone, South Africa, Sub-Sahara, Sudan, Tanzania, Togo, Tunisia, Uganda, Zambia and Zimbabwe.

4.2.1 Features of the dataset

Prediction of population size is influenced by many factors, in this study's we will consider only ten. The table below shows the respective countries with their features and the number of missing values.

Table 4.1: Analysis of missing values for each country.

No.	Country	GDP	EGS	IGS	RG	U G	I	Ind	AFF	MT	P
1	Algeria	1	0	1	1	1	1	39	39	0	0
2	Angola	21	40	44	1	1	21	35	35	20	0
3	Benin	1	0	1	1	1	1	0	0	0	0
4	Burkina Faso	1	3	9	1	1	1	0	0	0	0
5	Burundi	1	0	38	1	1	1	10	10	0	0
6	Cameroon	1	5	1	1	1	1	5	5	0	0
7	Chad	1	0	5	1	1	1	0	0	0	0
8	Congo Rep	1	0	1	1	1	1	0	0	0	0
9	Cote D'voire	1	0	37	1	1	1	0	0	0	0
10	DR Congo	1	34	1	1	1	1	34	34	0	0
11	Egypt	1	0	1	1	1	1	0	0	5	0
12	Equatorial Guinea	21	45	46	1	1	21	46	46	4	0
13	Eswatini	11	1	2	1	1	11	6	1	0	0
14	Gabon	1	0	1	1	1	1	0	0	0	0
15	Gambia	8	6	45	1	1	7	6	6	6	0
16	Ghana	1	0	47	1	1	1	5	0	0	0
17	Guinea	27	26	47	1	1	27	26	26	26	0
18	Guinea Bissau	11	12	45	1	1	11	12	12	10	0
19	Kenya	1	0	1	1	1	1	0	0	0	0
20	Lesotho	1	26	3	1	1	17	3	0	0	0
21	Lybia	40	33	46	1	1	40	42	42	30	0
22	Mali	8	7	8	1	1	8	7	7	7	0
23	Morocco	7	0	1	1	1	7	5	5	0	0
24	Mozambique	21	32	22	1	1	32	32	32	31	0
25	Namibia	21	20	21	1	1	21	20	20	20	0
26	Niger	1	0	31	1	1	1	1	1	0	0
27	Nigeria	1	1	23	1	1	1	21	21	0	0
28	Rwanda	1	0	1	1	1	1	5	5	0	0
29	Senegal	1	0	1	1	1	1	0	0	0	0
30	Sierra Leone	1	5	9	1	1	1	5	5	0	0
31	South Africa	1	0	1	1	1	1	0	0	0	0
32	Sub-Sahara	1	0	22	1	1	1	21	21	0	0
33	Sudan	1	0	1	1	1	1	0	0	0	0
34	Tanzania	29	30	31	1	1	29	30	30	28	0
35	Togo	1	0	1	1	1	1	0	0	0	0

Notice that GDP, EGS, IGS, RG, UG, I, IND, AFF, MT, and P represents annual GDP growth, exports of goods and services, import of goods and services, rural growth, urban growth, inflation, GDP deflator, industry (including construction), value added (% of GDP), agriculture, forestry, and fishing, value added (% of GDP), merchandise trade, and population respectively. Due to these missing values, it was necessary to introduce the imputation method. Below is a

Table 4.1 (cont.)

No.	Country	GDP	EGS	IGS	RG	U G	I	Ind	AFF	MT	P
36	Tunisia	6	5	6	1	1	6	5	5	5	0
37	Uganda	23	0	23	1	1	23	0	0	0	0
38	Zambia	1	34	13	1	1	1	0	0	0	0
39	Zimbabwe	1	16	18	1	1	1	10	6	0	0

brief explanation of the features used in this work:

1. **Annual GDP growth (GDP):** This gauges a nation's overall economic health. Better living conditions and employment prospects are frequently correlated with higher GDP growth, which may promote immigration and higher birth rates and hence population expansion

2. **Exports of goods and services (EGS):** A healthy export industry is a sign of a healthy economy, which might draw job seekers. Additionally, infrastructure improvements and better public services might result from export-led growth, which can affect population dynamics.

3. **Import of goods and services (IGS):** A high import volume may indicate robust economic activity and consumer demand. This might draw labourers from other areas or nations, changing the distribution and size of the population.

4. **Rural growth (RG):** Variations in these rates shed light on rural migration trends and economic prospects. Growing rural areas may be a sign of improved living standards or more employment prospects in the agricultural or rural industries.

5. **Urban growth (UG):** The growth of the urban population is a crucial aspect of population studies. People from rural areas and other countries are drawn to cities because they often provide better healthcare, education, work prospects, and living conditions.

6. **Inflation, GDP deflator (annual %) (I):** This gauges inflation and shifts in price levels. Excessive inflation can reduce savings and purchasing power, which may have an impact on birth rates and migration trends as people look for more stable economic conditions.

7. **Industry (including construction), value added (% of GDP) (Ind):** This shows how much the construction and industrial sectors contribute to the economy. A robust industrial sector can influence population growth by generating a large number of employment that draw workers and their families.

8. **Agriculture, forestry, and fishing, value added (% of GDP) (AFF):** The impact of the agriculture, forestry, and fishing sector's value added on population growth is multifaceted and context-dependent. It encompasses economic, social, and environmental dimensions, all of which interplay to shape population dynamics.

9. **Merchandise trade (% of GDP) (MT):** This percentage illustrates how significant trade is to the economy as a whole. Elevated trade volumes may suggest economic transparency and assimilation into the worldwide marketplace, which may result in shifts in the population due to immigration and economic prospects.

10. **Population (P):** Population is the term typically used to refer to the number of people in a single area. Governments conduct a census to quantify the size of a resident population within a given jurisdiction.

4.3 Model description

The key objective of the RNN model in this study is to predict the population size of some selected African countries based on historical data which has been explained earlier in our previous section.
Recurrent Neural Networks (RNNs) are particularly well-suited for sequential data due to their inherent design and characteristics in the following ways: temporal dynamics, memory of previous input, variable length sequence, and parameter sharing.

4.3.1 Important libraries

- **os:** Manages file system interactions. Useful for loading data from files, saving model checkpoints, and organizing directories for input and output data.

- **Pandas (pd):** Provides data structures and data analysis tools.It is essential for data preprocessing, cleaning, and manipulation. It helps in loading datasets into DataFrame objects, which makes it easier to handle time-series data and perform operations like merging, filtering, and aggregation.

- **Numpy (np):** Supports large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. Crucial for numerical operations and handling array-based data. RNNs often require operations on matrices, and numpy provides efficient tools for such computations.

- **Scikit-learn (sklearn):** Provides simple and efficient tools for data mining and data analysis. It is useful for preprocessing data (e.g., scaling, encoding), splitting datasets, and evaluating models using metrics. While sklearn is not used for building RNNs, it supports tasks like feature selection, model validation, and splitting data into training and test sets.

- **Tensorflow (tf):** is an open-source library for machine learning and deep learning developed by Google. It is used for building and training machine learning models, particularly neural networks. tensorflow provides a flexible platform for implementing both high-level and low-level machine learning models.

- **Keras:** It is a high-level API that simplifies the creation of RNN layers (e.g., SimpleRNN, LSTM, GRU) and models. Facilitates model compilation, training, and evaluation with user-friendly functions and methods.

- **matplotlib and pyplot (plt):** is a plotting library for the Python programming language and its numerical mathematics extension numpy. It is employed in the production of animated, interactive, and static visualisations. It is crucial for displaying data, identifying trends, and presenting findings.

- **LearningRateScheduler:** It is a callback in keras, this allows adjustment in the learning rate when training the model. Helps in optimizing the training process by dynamically changing the learning rate according to a predefined schedule. This can lead to faster convergence and improved model performance.

4.3.2 Model architecture

As various architect of our model has be mentioned earlier, we shall have a specific description in this section. Below is a detailed explanation of how the model is designed for this work.

1. Input : It accepts sequences of normalized population counts. The input shape is having a dimension of (50,11), where 50 is the number of time steps and 11 is the number of features in the x-training set i.e., all the columns in the training data except the target column.

2. Recurrent layer: A simple recurrent neural network with 64 hidden units and a Leaky ReLU (with $\alpha = 0.3$) activation function is used.

3. Dense layer: A fully connected layer with one neuron and a Leaky ReLU (with $\alpha = 0.3$) activation function to predict the next year's population is also used.

4.3.3 Training

Model training is carried out after all preprocessing steps are done. Below are few specifications engaged during the process:

1. Loss function: In this experiment, we used the mean squared error (MSE) as the loss function.

2. Optimizer: The Adam optimizer is used.

3. Hyperparameters: A Learning rate of 0.001, epoch of 3000, and a batch size of 128 is employed in the training.

4. Training procedure: As explained in chapter 2, the training, validation and test set is split in the percentage 80, 10 and 10 respectively. Each consists of 39 (countries) sequences of training_size in years, but each is shifted in time. The evaluation is performed only on the part of the test set which is not overlapping with training and validation sets.

4.3.4 Quality metrics

To evaluate the quality of the population prediction models, various metrics are used:

- Mean absolute error (MAE): MAE measures the average magnitude of errors in the predictions, without considering their direction.

- Mean squared error (MSE): MSE calculates the average of the squares of the errors, giving more weight to larger errors.

- Root mean squared error (RMSE): The square root of MSE, providing an error metric in the same units as the population data.

- R-squared (R^2): Indicates the proportion of the variance in the dependent variable that is predictable from the independent variables.

These metrics help in assessing the accuracy and reliability of the prediction models, guiding further improvements and adjustments.

4.3.5 Implementation

Below are code snippets illustrating the recurrent neural network implementation.

1. Mean imputation.

```
import os
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import SimpleRNN, Dense, ...
    LeakyReLU
from tensorflow.keras.optimizers import Adam
from sklearn.metrics import mean_squared_error
from matplotlib import pyplot as plt
from tensorflow.keras.callbacks import ...
    LearningRateScheduler

# Get files from directory
directory = 'E:\\Data'
csv_files = [file for file in os.listdir(directory) if ...
    file.endswith('.csv')]

#Percentage of data for each split
train_percent = 0.8
val_percent = 0.1
test_percent = 0.1

#Creating empty DataFrame to stack datasets
stacked_train = pd.DataFrame()
stacked_val = pd.DataFrame()
stacked_test = pd.DataFrame()

for file in csv_files:
file_path = os.path.join(directory, file)
df = pd.read_csv(file_path)
df = df.iloc[:,:11]
df['Population'] = df.loc[:,'Population'] / 100000
df['target'] = df.loc[:,'Population'].shift(-1)
df = df.iloc[:-1]

# Calculating the number of rows
total_rows = len(df)
val_size = int(val_percent*total_rows)
test_size = int(test_percent*total_rows)
train_size = total_rows-val_size-test_size

```

```
train = df.iloc[:train_size, :]
val = df.iloc[val_size:train_size + val_size, :]
test = df.iloc[val_size + test_size:, :]

# Using mean of training set for each column to fill the missing values in the test and validation set
mean_value = train.mean()
train_imputed = train.fillna(mean_value)
val_imputed = val.fillna(mean_value)
test_imputed = test.fillna(mean_value)

# Stacking datasets
stacked_train = pd.concat([stacked_train, train_imputed], ignore_index=False)
stacked_val = pd.concat([stacked_val, val_imputed], ignore_index=False)
stacked_test = pd.concat([stacked_test, test_imputed], ignore_index=False)


# Read stacked_train data files from path
stacked_train_data = pd.read_csv(os.path.join(stacked_train_folder_path, 'stacked_train.csv'))

target_column = 'Population'
x_stacked_train = stacked_train_data.drop(target_column, axis=1)
y_stacked_train = pd.DataFrame(stacked_train_data[target_column])

# Reshaping the data for RNN input (num_samples, num_time_step, num_features)
x_stacked_train = x_stacked_train.values.reshape(-1, train_size, x_stacked_train.shape[1])

y_stacked_train = y_stacked_train.values.reshape(-1, train_size)

stacked_val_data = pd.read_csv(os.path.join(stacked_val_folder_path, 'stacked_val.csv'))
x_stacked_val = stacked_val_data.drop(target_column, axis=1)
y_stacked_val = pd.DataFrame(stacked_val_data[target_column])

# Reshaping validation data
x_stacked_val = x_stacked_val.values.reshape((-1, train_size, x_stacked_val.shape[1]))
y_stacked_val = y_stacked_val.values.reshape(-1, ...
```

```
    train_size)
77
78 # Setting path to folder containg stacked csv files
79 stacked_test_folder_path = 'test_imputed_folder'
80
81
82 # Read the stacked test data
83 stacked_test_data = ...
    pd.read_csv(os.path.join(stacked_test_folder_path, ...
    'stacked_test.csv'))
84 x_stacked_test = stacked_test_data.drop(target_column, ...
    axis=1)
85 y_stacked_test = ...
    pd.DataFrame(stacked_test_data[target_column])
86
87 x_stacked_test = x_stacked_test.values.reshape((-1, ...
    train_size, x_stacked_test.shape[1]))
88
89 y_stacked_test = y_stacked_test.values.reshape(-1, ...
    train_size)
90
91 # Defining my RNN model
92
93 def build_rnn_model(hidden_units, dense_units, ...
    input_shape, activation):
94 model = Sequential()
95 model.add(SimpleRNN(hidden_units, ...
    input_shape=input_shape, activation=activation[0], ...
    return_sequences=True))
96 model.add(Dense(units=dense_units, ...
    activation=activation[1]))
97 model.compile(loss='mean_squared_error', ...
    optimizer=Adam(learning_rate=0.001))
98 return model
99
100 input_shape = (x_stacked_train.shape[1], ...
    x_stacked_train.shape[2])
101 model = build_rnn_model(hidden_units=64, dense_units ...
    =1, input_shape=input_shape, ...
    activation=[LeakyReLU(alpha=0.3), LeakyReLU(alpha=0.3)])
102 history = model.fit(x_stacked_train, y_stacked_train, ...
    epochs = 3000, batch_size=128, ...
    validation_data=(x_stacked_val,y_stacked_val))
103
104 # Making prediction on the training set and computing ...
    the mean absolute error
105 train_pred = ...
    model.predict(x_stacked_train).reshape((39,-1))
106 np.mean(np.abs(train_pred-y_stacked_train))
107
108 val_pred = ...
```

```
    model.predict(x_stacked_val).reshape((39,-1))[:,-val_size:-1]
np.mean(np.abs(val_pred-y_stacked_val[:,-val_size:-1]))

test_pred = ...
    model.predict(x_stacked_test).reshape((39,-1))[:,-test_size:
-1]
np.mean(np.abs(test_pred-y_stacked_test[:,-test_size:-1]))


#Calclating the naive forecast error.
np.mean(np.abs(y_stacked_test[:,-test_size:-1]-
x_stacked_test[:,-test_size:-1,-1]))

for i in range(y_stacked_test.shape[0]):
y_pred = test_pred[i]
y_true = y_stacked_test[i,-test_size:-1]
sum_squares_residuals = np.sum((y_true-y_pred)**2)
sum_squares = np.sum((y_true-np.mean(y_pred))**2)
R2 = 1-sum_squares_residuals/sum_squares
print(f"Country: {csv_files[i].split('.')[0]} R2: ...
    {R2:.3f} Mean: {np.mean(y_pred):.3f}")

# Plotting training and validation loss
plt.plot(history.history['loss'], label= 'Training Loss')
plt.plot(history.history['val_loss'], label= ...
    'Validation Loss')
plt.legend()
plt.title('Training and Validation Loss over epochs')
plt.show()
```

2. Linear interpolation.

```
import os
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import SimpleRNN, Dense, ...
    LeakyReLU
from tensorflow.keras.optimizers import Adam
from sklearn.metrics import mean_squared_error
from matplotlib import pyplot as plt
from tensorflow.keras.callbacks import ...
    LearningRateScheduler

# Get files from directory
directory = 'E:\\Data'
```

```
csv_files = [file for file in os.listdir(directory) if file.endswith('.csv')]

#Percentage of data for each split
train_percent = 0.8
val_percent = 0.1
test_percent = 0.1

# Creating empty Dataframe to stack datasets
stacked_train = pd.DataFrame()
stacked_val = pd.DataFrame()
stacked_test = pd.DataFrame()

for file in csv_files:
file_path = os.path.join(directory, file)
df = pd.read_csv(file_path)

df = df.interpolate().ffill().bfill()

df = df.iloc[:, :11]
df['Population'] = df.loc[:, 'Population']/100000
df['target'] = df.loc[:, 'Population'].shift(-1)
df = df.iloc[:-1]

# Calculating the number of rows and sizes of splits ... respectively
total_rows = len(df)
val_size = int(val_percent*total_rows)
test_size = int(test_percent*total_rows)
train_size = total_rows-val_size-test_size

train = df.iloc[:train_size, :]
val = df.iloc[val_size:train_size + val_size, :]
test = df.iloc[val_size + test_size:, :]

# Stacking datasets
stacked_train = pd.concat([stacked_train, train], ... ignore_index=False)
stacked_val = pd.concat([stacked_val, val], ... ignore_index=False)
stacked_test = pd.concat([stacked_test, test], ... ignore_index=False)


# Read the stacked train data
stacked_train_data = ... pd.read_csv(os.path.join(stacked_train_folder_path, ... 'stacked_train.csv'))

target_column = 'Population'
x_stacked_train = ...
```

```
    stacked_train_data.drop(target_column, axis=1)
y_stacked_train = ...
    pd.DataFrame(stacked_train_data[target_column])

# Reshaping the data for RNN input(num_samples, ...
    num_time_steps, num_features)
x_stacked_train = x_stacked_train.values.reshape((-1,  ...
    train_size, x_stacked_train.shape[1]))

y_stacked_train = ...
    y_stacked_train.values.reshape(-1,train_size)

stacked_val_data = ...
    pd.read_csv(os.path.join(stacked_val_folder_path, ...
    'stacked_val.csv'))
x_stacked_val = stacked_val_data.drop(target_column, ...
    axis=1)
y_stacked_val = ...
    pd.DataFrame(stacked_val_data[target_column])

# Reshaping validation data
x_stacked_val = x_stacked_val.values.reshape((-1, ...
    train_size, x_stacked_val.shape[1]))

y_stacked_val = y_stacked_val.values.reshape(-1, ...
    train_size)

# Setting the path to the test folder containing ...
    stacked csv files for test set
stacked_test_folder_path = 'test_interpolate_folder'

# Read the stacked test data
stacked_test_data = ...
    pd.read_csv(os.path.join(stacked_test_folder_path, ...
    'stacked_test.csv'))
x_stacked_test = stacked_test_data.drop(target_column, ...
    axis=1)
y_stacked_test = ...
    pd.DataFrame(stacked_test_data[target_column])

# Reshaping the stacked test data
x_stacked_test = x_stacked_test.values.reshape((-1, ...
    train_size, x_stacked_test.shape[1]))


y_stacked_test = y_stacked_test.values.reshape(-1, ...
    train_size)

# Defining my model RNN

def build_rnn_model(hidden_units, dense_units, ...
```

```
    input_shape, activation):
93 model = Sequential()
94 model.add(SimpleRNN(hidden_units, ...
    input_shape=input_shape, activation=activation[0], ...
    return_sequences=True))
95 model.add(Dense(units=dense_units, ...
    activation=activation[1]))
96 model.compile(loss='mean_squared_error', ...
    optimizer=Adam(learning_rate=0.001))
97 return model
98
99 input_shape = (x_stacked_train.shape[1], ...
    x_stacked_train.shape[2])
100 # Building and training RNN model on the stacked train ...
    data
101 model = build_rnn_model(hidden_units = 64, ...
    dense_units=1, input_shape=input_shape, ...
    activation=[LeakyReLU(alpha=0.3), LeakyReLU(alpha=0.3)])
102 history = model.fit(x_stacked_train, y_stacked_train, ...
    epochs=3000, batch_size=128, ...
    validation_data=(x_stacked_val,y_stacked_val))
103
104 # Making prediction on the training set and computing ...
    the mean absolute error
105 train_pred = ...
    model.predict(x_stacked_train).reshape((39, -1))
106 np.mean(np.abs(train_pred-y_stacked_train))
107
108 val_pred = model.predict(x_stacked_val).reshape((39, - ...
    1))[:,-val_size:-1]
109 np.mean(np.abs(val_pred-y_stacked_val[:,-val_size:-1]))
110
111 test_pred = model.predict(x_stacked_test).reshape((39, ...
    -1))[:,-test_size:-1]
112 np.mean(np.abs(test_pred-y_stacked_test[:,-test_size:-1]))
113
114 #Calculating the naive forecast error.
115 np.mean(np.abs(y_stacked_test[:,-test_size:-1]-
116 x_stacked_test[:,-test_size:-1,-1]))
117
118 for i in range(y_stacked_test.shape[0]):
119 y_pred = test_pred[i]
120 y_true = y_stacked_test[i,-test_size:-1]
121 sum_squares_residuals = np.sum((y_true - y_pred)**2)
122 sum_squares = np.sum((y_true - np.mean(y_true))**2)
123 R2 = 1 - sum_squares_residuals / sum_squares
124 print(f"Country: {csv_files[i].split('.')[0]} R2: ...
    {R2:.3f} Mean: {np.mean(y_true):.3f}")
125
126 # Plotting Training and Validation Loss
127 plt.plot(history.history['loss'], label = 'Training Loss')
```

```
plt.plot(history.history['val_loss'], label = 'Validation Loss')
plt.legend()
plt.title('Training and Validation Loss over epochs')
plt.show()
```

3. Linear interpolation + indicators

```
import os
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import SimpleRNN, Dense, LeakyReLU
from tensorflow.keras.optimizers import Adam
from sklearn.metrics import mean_squared_error
from matplotlib import pyplot as plt
from tensorflow.keras.callbacks import LearningRateScheduler

# Get files from directory
directory = 'E:\\Data'
csv_files = [file for file in os.listdir(directory) if file.endswith('.csv')]

#Percentage of data for each split
train_percent = 0.8
val_percent = 0.1
test_percent = 0.1

# Creating empty DataFrame to stacked my datasets
stacked_train = pd.DataFrame()
stacked_val = pd.DataFrame()
stacked_test = pd.DataFrame()

for file in csv_files:
file_path =os.path.join(directory, file)
df = pd.read_csv(file_path)
df= df.iloc[:,:11]
df['Population'] = df.loc[:, 'Population'] / 100000
df['target'] = df.loc[:, 'Population'].shift(-1)
df = df.iloc[:-1]

df_indicator = df.interpolate().ffill().bfill()

# Creating a new DataFrame for imputation
```

```
df_interpolated = df_indicator.copy()
# Adding new columns filled with ones and zeros
for column in df.columns:
new_column_name = column + "_indicator"
df_interpolated[new_column_name] = 1
df_interpolated[new_column_name] = ...
    df_interpolated[new_column_name] * ...
    df[column].isna().astype(int)
# Calculating the number of rows and sizes of splits ...
    respectively
total_rows = len(df_interpolated)
val_size = int(val_percent*total_rows)
test_size = int(test_percent*total_rows)
train_size = total_rows - val_size - test_size

# Splitting the dataset into training, validation and ...
    testing sets
train = df_interpolated.iloc[:train_size, :]
val = df_interpolated.iloc[val_size:train_size + ...
    val_size, :]
test = df_interpolated.iloc[val_size + test_size:, :]
print(df.shape, train.shape, val.shape, test.shape)

# Stacking datasets
stacked_train = pd.concat([stacked_train, train], ...
    ignore_index=True)
stacked_val = pd.concat([stacked_val, val], ...
    ignore_index=True)
stacked_test = pd.concat([stacked_test, test], ...
    ignore_index=True)

# Read the stacked train data
stacked_train_data = ...
    pd.read_csv(os.path.join(stacked_train_folder_path, ...
    'stacked_train.csv'))
stacked_val_data = ...
    pd.read_csv(os.path.join(stacked_val_folder_path, ...
    'stacked_val.csv'))
stacked_test_data = ...
    pd.read_csv(os.path.join(stacked_test_folder_path, ...
    'stacked_test.csv'))

target_column = 'Population'
x_stacked_train = ...
    stacked_train_data.drop(target_column, axis=1)
y_stacked_train = ...
    pd.DataFrame(stacked_train_data[target_column])

#Reshaping the data for RNN input(num_sample, ...
    num_time_steps, num_features)
x_stacked_train = x_stacked_train.values.reshape((-1, ...
```

```
    train_size, x_stacked_train.shape[1]))
y_stacked_train = y_stacked_train.values.reshape(-1, ...
    train_size)

x_stacked_val = stacked_val_data.drop(target_column, ...
    axis=1)
y_stacked_val = ...
    pd.DataFrame(stacked_val_data[target_column])

# Reshaping validation data
x_stacked_val = x_stacked_val.values.reshape(-1, ...
    train_size, x_stacked_val.shape[1])
y_stacked_val = y_stacked_val.values.reshape(-1, ...
    train_size)

x_stacked_test = stacked_test_data.drop(target_column, ...
    axis=1)
y_stacked_test = ...
    pd.DataFrame(stacked_test_data[target_column])

# Reshaping the stacked test data
x_stacked_test = x_stacked_test.values.reshape((-1, ...
    train_size, x_stacked_test.shape[1]))

y_stacked_test = y_stacked_test.values.reshape(-1, ...
    train_size)

# Defining my model RNN

def build_rnn_model(hidden_units, dense_units, ...
    input_shape, activation):
model = Sequential()
model.add(SimpleRNN(hidden_units, ...
    input_shape=input_shape, activation=activation[0], ...
    return_sequences=True))
model.add(Dense(units=dense_units, ...
    activation=activation[1]))
model.compile(loss='mean_squared_error', ...
    optimizer=Adam(learning_rate=0.004))
return model

input_shape = (x_stacked_train.shape[1], ...
    x_stacked_train.shape[2])
# Building and training the RNN model on the stacked ...
    train data
model = build_rnn_model(hidden_units=64, dense_units = ...
    1, input_shape=input_shape, ...
    activation=[LeakyReLU(alpha=0.3), LeakyReLU(alpha=0.3)])
history = model.fit(x_stacked_train, y_stacked_train, ...
    epochs = 3000, batch_size = 128, ...
    validation_data=(x_stacked_val,y_stacked_val))
```

```
103
104 # Making prediction on the training set and computing ...
       the mean absolute error
105 train_pred = ...
       model.predict(x_stacked_train).reshape((39,-1))
106 np.mean(np.abs(train_pred-y_stacked_train))
107
108 val_pred = ...
       model.predict(x_stacked_val).reshape((39,-1))[:,-val_size:-1]
109 np.mean(np.abs(val_pred-y_stacked_val[:,-val_size:-1]))
110
111 test_pred = ...
       model.predict(x_stacked_test).reshape((39,-1))[:,-test_size:
112 -1]
113 np.mean(np.abs(test_pred-y_stacked_test[:,-test_size:-1]))
114
115 #Calculating the naive forecast error.
116 np.mean(np.abs(y_stacked_test[:,-test_size:-1]-
117 x_stacked_test[:,-test_size:-1,-1]))
118
119 for i in range(y_stacked_test.shape[0]):
120 y_pred = test_pred[i]
121 y_true = y_stacked_test[i,-test_size:-1]
122 sum_squares_residuals = np.sum((y_true - y_pred)**2)
123 sum_squares = np.sum((y_true - np.mean(y_true))**2)
124 R2 = 1 - sum_squares_residuals/sum_squares
125 print(f"Country: {csv_files[i].split('.')[0]}) R2: ...
       {R2:.3f} Mean: {np.mean(y_pred):.3f}")
126
127 # Plotting Training and Validation Loss
128 plt.plot(history.history['loss'], label = 'Training Loss')
129 plt.plot(history.history['val_loss'], label = ...
       'Validation Loss')
130 plt.legend()
131 plt.title('Training and Validation Loss over Epochs')
132 plt.show()
```

Chapter 5

EXPERIMENTAL RESULTS

This work used the methodology depicted in Fig 5.1 to experimentally assess the impact of missing value imputation in the time series forecasting problem. The first stage is to use data preprocessing techniques including data reduction, cleansing, and transformation to create a refined dataset. The missing values in the dataset are then filled in using imputation techniques. We used three different imputation techniques for this process: mean substitution, linear interpolation, and linear interpolation plus corresponding indicators. Once each imputed dataset was trained, the various forecasting models were used. Finally, loss functions were used to compare and assess the forecasting models' performance.

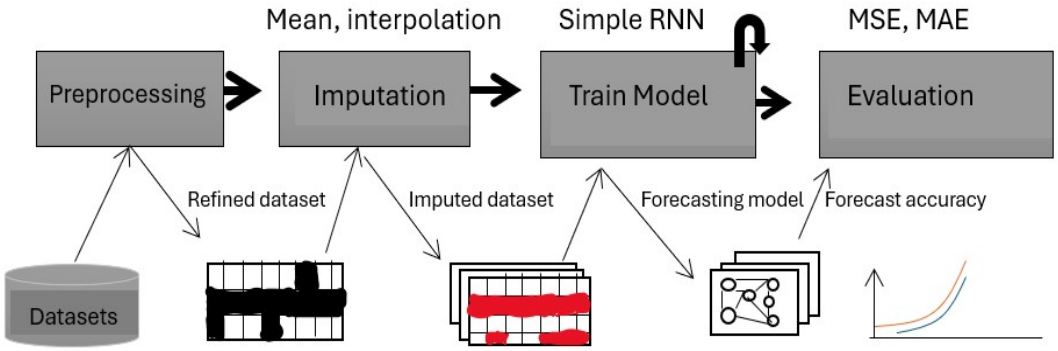

Figure 5.1: A design for the experimental procedure for this work.

5.1 Aggregate results

We show here the aggregate results for all countries. We show results for different imputation methods when used with RNN model. We compare these forecasting methods with one naive forecasting method.

A naive forecast is a simple prediction method where the forecast for a future value is assumed to be the same as the most recent observed value. In the context of time series data, this means

predicting that the next value in the series will be the same as the last observed value.

Table 5.1: A table showing prediction accuracies of the RNN model on all three imputation methods.

Method	Naive forecast error	MAE train pred	MAE val pred	MAE test pred
Mean	14.398	1.246	1.190	1.547
Inter	14.398	1.295	1.597	2.050
Inter + ind	14.398	3.838	3.863	4.257

5.2 Per country analysis

In this section, the analysis of the results obtained for all the countries involved will be discussed, and various tables will be shown accordingly. Below are tables for the results attained for all three imputation methods evaluated by the RNN model.

5.2.1 Visual summary

The graph shows the losses obtained during the training of the RNN model on the training and validation set. After the 3000 epochs, the loss on training (train) sets are 4.8812, 4.1485 and 16.0477 for mean, interpolation and interpolation + indicators imputation respectively. And that of validation (val) are 6.3309, 11.3007 and 19.4815 for mean, interpolation and interpolation + indicators imputation respectively. Below gives pictures of how all losses declined with increasing epochs.

Note that interpolation and indicators are shorten in the Table 5.5 as inter and ind respectively. And mean absolute error as MAE.

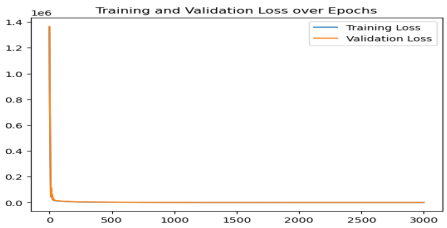

Figure 5.2: A graph plot showing the loss over epochs on training and validation sets for mean imputation method.

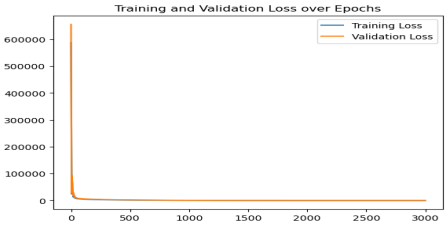

Figure 5.3: A graph plot showing the loss over epochs on training and validation sets for interpolation method.

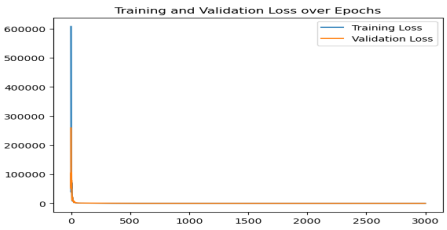

Figure 5.4: A graph plot showing the loss over epochs on training and validation sets for interpolation + indicators.

1. **Mean imputation method.**

Table 5.2: Results of R^2 values and mean population sizes of mean imputation

Country	R^2	**mean population (hundred thousands)**
Algeria	0.982	417.751
Angola	0.978	314.863
Benin	0.998	119.301
Burkina Faso	1.000	204.037
Burundi	0.983	115.238
Cameroon	0.997	251.414
Chad	0.982	156.935
Congo Rep	0.484	54.458
Cote d'Ivoire	0.999	255.089
DR Congo	0.978	876.680
Egypt	0.937	1029.919
Equatorial Guinea	0.406	15.701
Eswatini	-1.121	11.717
Gabon	0.291	22.467
Gambia	0.111	24.461
Ghana	0.995	308.324
Guinea	0.983	125.066
Guinea-Bissau	-3.374	19.330
Kenya	0.991	498.151
Lesotho	0.816	22.121
Libya	-1.017	65.124
Mali	0.998	199.791
Morocco	0.800	356.336
Mozambique	0.997	294.754
Namibia	0.798	24.127
Niger	0.988	227.344
Nigeria	0.976	1995.116
Rwanda	0.992	124.977
Senegal	1.000	155.957
Sierra Leone	0.979	78.497
South Africa	0.763	569.957
Sub-Saharan Africa	1.000	10929.046
Sudan	0.983	421.646
Tanzania	0.994	582.650
Togo	0.962	80.133

Table 5.2 (cont.)

Country	R^2	**mean population (hundred thousands)**
Tunisia	0.850	118.673
Uganda	0.991	417.341
Zambia	0.988	179.205
Zimbabwe	0.679	152.588

2. **Linear interpolation method.**

Table 5.3: Results of R^2 values and mean population sizes of African countries

Country	R^2	**mean population (hundred thousands)**
Algeria	0.986	419.120
Angola	0.987	312.838
Benin	0.962	119.462
Burkina Faso	0.994	203.957
Cameroon	0.996	250.910
Chad	0.972	156.108
Congo Rep	-0.884	54.426
Cote d'Ivoire	0.998	255.030
DR Congo	0.956	871.123
Egypt	0.983	1036.796
Equatorial Guinea	0.882	15.002
Burundi	0.957	115.295
Eswatini	-20.838	11.609
Gabon	0.197	21.908
Gambia	0.038	24.452
Ghana	0.985	308.700
Guinea	0.784	125.619
Kenya	0.997	499.467
Lesotho	0.306	21.985
Libya	-0.280	64.723
Guinea-Bissau	-16.292	19.251
Mali	0.997	199.474
Morocco	0.835	359.112
Mozambique	0.996	294.307

Table 5.3 (cont.)

Country	R^2	**mean population (hundred thousands)**
Namibia	0.525	24.059
Niger	0.992	226.028
Nigeria	0.992	1984.365
Rwanda	0.989	125.349
Senegal	0.998	155.842
Sierra Leone	0.779	78.627
South Africa	0.846	574.584
Sub-Saharan Africa	0.997	10930.593
Sudan	0.970	419.457
Tanzania	0.991	580.673
Togo	0.799	80.493
Tunisia	0.758	119.282
Uganda	0.987	415.489
Zambia	0.989	178.420
Zimbabwe	-1.419	150.561

3. **Linear interpolation + indicators.**

Table 5.4: Results of R^2 values and mean population sizes of African countries

Country	R^2	**Mean population (hundred thousands)**
Algeria	0.780	413.994
Angola	0.985	311.057
Benin	0.338	115.488
Burkina Faso	0.778	200.257
Burundi	0.517	112.036
Cameroon	0.911	247.979
Chad	0.772	152.639
Congo Rep	-3.672	50.565
Cote d'Ivoire	0.818	251.109

Table 5.4 (cont.)

Country	R^2	**Mean population (hundred thousands)**
DR Congo	0.998	872.273
Egypt	0.958	1031.381
Equatorial Guinea	-24.001	11.541
Eswatini	-592.966	8.361
Gabon	-14.245	19.107
Gambia	-10.171	21.438
Ghana	0.734	303.953
Guinea	0.180	121.551
Guinea-Bissau	-33.574	15.549
Kenya	0.999	499.430
Lesotho	-70.553	18.693
Libya	-7.350	61.412
Mali	0.865	196.207
Morocco	-0.636	351.988
Mozambique	0.902	290.507
Namibia	-26.072	21.029
Niger	0.971	223.976
Nigeria	0.920	2003.880
Rwanda	0.143	121.377
Senegal	0.644	152.288
Sierra Leone	-1.499	74.511
South Africa	-0.316	564.477
Sub-Saharan Africa	0.999	10916.004
Sudan	0.996	418.770
Tanzania	0.997	581.885
Togo	-0.867	76.727
Tunisia	-7.188	114.466
Uganda	1.000	415.339
Zambia	0.802	175.029
Zimbabwe	-0.672	145.216

Table 5.5: A summary of the best prediction of each country by the recurrent neural network based on the imputation methods.

Country	R^2	**Mean population (hundred thousands)**	**Best method**
Algeria	0.986	419.120	Interpolation
Angola	0.987	312.838	Interpolation
Benin	0.998	119.301	Mean
Burkina Faso	1.00	204.037	Mean
Burundi	0.983	115.238	Mean
Cameroon	0.997	251.414	Mean
Chad	0.982	156.935	Mean
Congo Rep	0.484	54.458	Mean
Cote d'Ivoire	0.999	255.089	Mean
DR Congo	0.998	872.273	Inter + ind
Egypt	0.983	1036.796	Interpolation
Equatorial Guinea	0.882	15.002	Interpolation
Eswatini	-1.121	11.717	Mean
Gabon	0.291	22.467	Mean
Gambia	0.111	24.461	Mean
Ghana	0.995	308.324	Mean
Guinea	0.983	125.066	Mean
Guinea-Bissau	-3.374	19.330	Mean
Kenya	0.999	499.430	Inter + ind
Lesotho	0.816	22.121	Mean
Libya	-0.280	64.723	Interpolation
Mali	0.998	199.791	Mean
Morocco	0.835	359.112	Interpolation
Mozambique	0.997	294.754	Mean
Namibia	0.798	24.127	Mean
Niger	0.992	226.028	Interpolation
Nigeria	0.992	1984.365	Interpolation
Rwanda	0.992	124.977	Mean
Senegal	1.00	155.957	Mean
Sierra Leone	0.979	78.497	Mean
South Africa	0.846	574.584	Interpolation
Sub-Saharan Africa	1.00	10929.046	Mean
Sudan	0.996	418.770	Inter + ind
Tanzania	0.997	581.885	Inter + ind
Togo	0.962	80.133	Mean
Tunisia	0.850	118.673	Mean
Uganda	1.00	415.339	Inter + ind
Zambia	0.989	580.673	Interpolation
Zimbabwe	0.679	152.588	Mean

Chapter 6

CONCLUSION

In this thesis, we have explored the application of recurrent neural networks (RNNs) for predicting the population sizes of African countries. The primary motivation behind this study was to leverage advanced machine learning techniques to address the challenges of population forecasting in a diverse and dynamic region of Africa. The main conclusions and future work directions are listed below.

- Model Performance: Strong prediction power was shown by the RNN model in most of the African countries on the mean imputation method. The model was able to accurately represent the trends and patterns in population increase, according to performance criteria like the mean population values and R^2 values. And interpolation + indicators performed poorly according to Table 5.4.

- Country specific insights: Tables 5.5 show the best imputation method for each country based on its R^2 and mean population values. From the Table 5.5, it can be seen that the RNN model performed very well and have higher predictive accuracy on countries like Burkina Faso, DR Congo, Senegal, Sub-Saharan Africa, Cote d'Ivoire, Benin,Mozambique, Ghana, Algeria,Niger, Egypt, Nigeria, Kenya, Uganda, etc. And performed very poor on countries like Guinea-Bissau, Libya, Eswatini, Gambia, Gabon. Others need improvement to perform much better, for instance Zimbabwe, Tunisia, Namibia, etc. If we examine from Table 4.1, the magnitude of the missing values are mega, but the predictions by the model looks very powerful.

- Future work: While the results of this study are promising, there are several areas for future work. Further enhancements to the RNN model, such as incorporating additional data sources (e.g., economic indicators, migration patterns) and experimenting with more sophisticated architectures like long short-term memory (LSTM) networks or gated recurrent units (GRUs), could improve prediction accuracy.

- Final remarks: Machine learning holds great promise for solving intricate demographic problems, as evidenced by the use of RNNs to forecast population sizes in African nations. The knowledge acquired from this research can be used as a basis for better strategic planning and decision-making, which will ultimately aid in the sustainable development of the African continent. Predictive models' capabilities will advance along with data collecting and technology, opening the door to ever more precise and useful population estimates.

Bibliography

[1] Carillo, Maria Rosaria, Vincenzo Lombardo, and Alberto Zazzaro. "The rise and fall of family firms in the process of development." Journal of Economic Growth 24 (2019): 43-78.

[2] Ritchie, Hannah, Veronika Samborska, and Max Roser. "Urbanization." Our world in data (2024).

[3] Graham, Victoria. "Youth Participation in Anglophone Africa. In Brittle Democracies: Comparing Politics in Anglophone Africa", edited by Heather A. Thuynsma (2020).

[4] World Demographics. Available at: https://www.worldometers.info/demographics/world-demographics/#median-age.

[5] Freire, Maria Emilia. "Urbanization and green growth in Africa." The Growth Dialogue 1 (2013): 1-38.

[6] Unemployment Rate in Africa. Available at: https://www.statista.com/statistics/1319860/unemployment-rate-in-africa.

[7] Woeste, Peter. "The World-an Old-Age Home." (2019): 9.

[8] Uinted Nations General Assembly. Transforming our World: The 2030 Agenda for Sustainable Development. A/RES/70/1.

[9] Ayodele, Taiwo Oladipupo. "Types of machine learning algorithms." New advances in machine learning 3, no. 19-48 (2010): 5-1.

[10] Basic Concepts in Machine Learning. Available at: https://www.javatpoint.com/basic-concepts-in-machine-learning.

[11] Loss Function. Available at: https://www.datacamp.com/tutorial/loss-function-in-machine-learning.

[12] Gradient Descent in Machine Learning. Available at: https://www.javatpoint.com/gradient-descent-in-machine-learning.

[13] Machine Learning Model Evaluation. Available at: https://www.geeksforgeeks.org/machine-learning-model-evaluation/.

[14] The Difference Between Training Data vs Test Data in Machine Learning. Available at: https://www.obviously.ai/post/the-difference-between-training-data-vs-test-data-in-machine-learning.

[15] Backpropagation. Available at: https://builtin.com/machine-learning/backpropagation-neural-network#:~:text=Backpropagation.

[16] Hewamalage, Hansika, Christoph Bergmeir, and Kasun Bandara. "Recurrent neural networks for time series forecasting: Current status and future directions." International Journal of Forecasting 37, no. 1 (2021): 388-427.

[17] Gareth, James, Witten Daniela, Hastie Trevor, and Tibshirani Robert. An introduction to statistical learning: with applications in R. Spinger, 2013.

[18] Kaiser, Jiří. "Dealing with Missing Values in Data." Journal of Systems Integration (1804-2724) 5, no. 1 (2014).