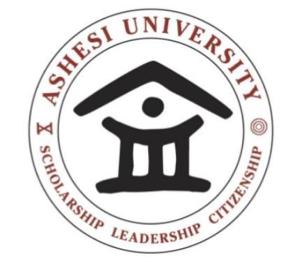
****

**PROJECT REPORT: ALGORITHM IMPLEMENTATION DETAILS**

**GROUP 1:**

**LESLIE TETTEY, KYEI BAFFOUR LAWERNCIA, HAWA ELIASU, INARES KENE TSANGUE**

**24922027**

**INTRODUCTION TO ARTIFICIAL INTELLIGENCE**

**CS 254\_C**

**DENNIS ASAMOAH OWUSU**

**MAY 1ST, 2025**

**TECHNICAL WRITE UP**

For our project we decided to use the machine learning model LSTM. LSTM (Long Short-Term Memory) is a type of Recurrent Neural Network (RNN) designed to handle time-series data. Time series data is data where past values influence future values. Unlike traditional neural networks, LSTMs have a memory mechanism that allows them to remember past patterns over longer sequences.

We used LSTM because SARS infection counts over time are sequential, meaning that the current day’s numbers are influenced by the last few days. LSTMs are good at learning such dependencies, which is ideal for forecasting future cases.

LSTM learns to minimize a loss function (usually mean squared error in regression) by adjusting internal weights using a process called backpropagation through time. It has memory cells and gates (input, forget, output) that control the flow of information through time steps. The goal is to find a function f(X) = y, where X is a window of past time points, and y is the next value to predict.

In LSTM, sequential order matters, it assumes that order and spacing of data points carry meaning. Data Normalization is also critical since neural nets are sensitive to large value ranges and more data helps the model performs better with more history.

We used the programming language python and certain libraries from python that served the purpose of our project. The first library we use used was Pandas. It is the standard library for working with structured data like csv files. It was used to read the SARS dataset from the csv file. It also handled data cleaning such as filling missing values, fixing column names, and filtering by country.We also perform grouping and summarizing, such as aggregating cases by country. Some of the functions we used were: pd.read\_csv(), df.fillna(), df.groupby()

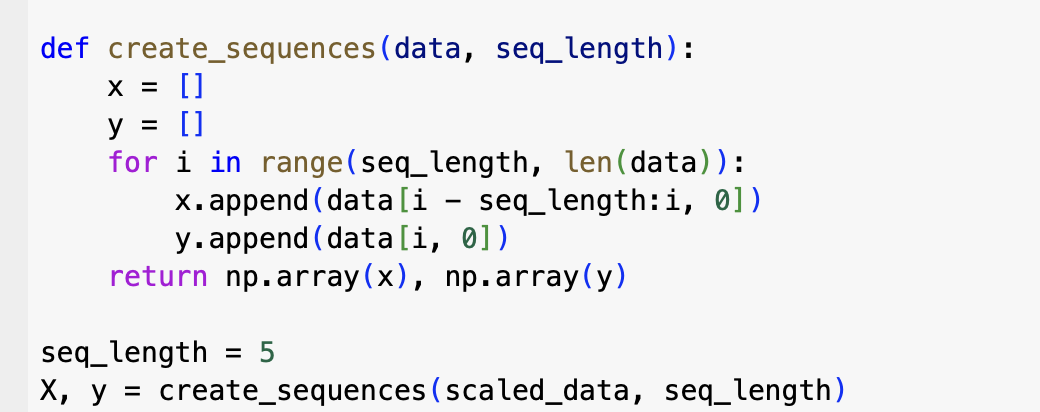
We also used torch from PyTorch which is a deep learning framework used to build and train the LSTM model. PyTorch provided us with Tensor objects for storing data, neural network models like LSTM layers and linear layers and training infrastructure such as loss functions, optimizers and back propagation.

We also used Numpy which was essential for numerical operations on arrays and matrices. We used it for handling and reshaping data before its fed into the LSTM. It was also used for creating sequences of data. Some functions we used were np.array() which converts lists to arrays.

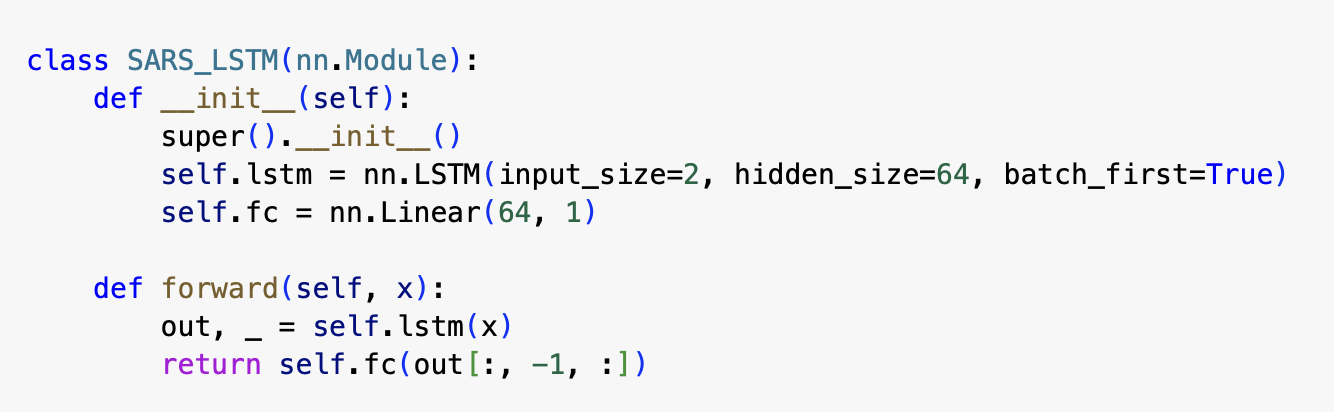
We used the MinMaxScaler to scale features numerical features into a normalized range, typically 0 to 1. This was crucial because LSTM performed better when inputs are normalized and without scaling, large input values can cause unstable training.

In implementing the steps for our algorithm we started with Data cleaning and processing. We started by removing duplicates, filling missing values, and cleaning temperature and humidity data. We then selected data for Germany and selected 4 input features: cumulative SARS cases, temperature, humidity, and public awareness. We then normalised the input features using the MinMaxScaler.

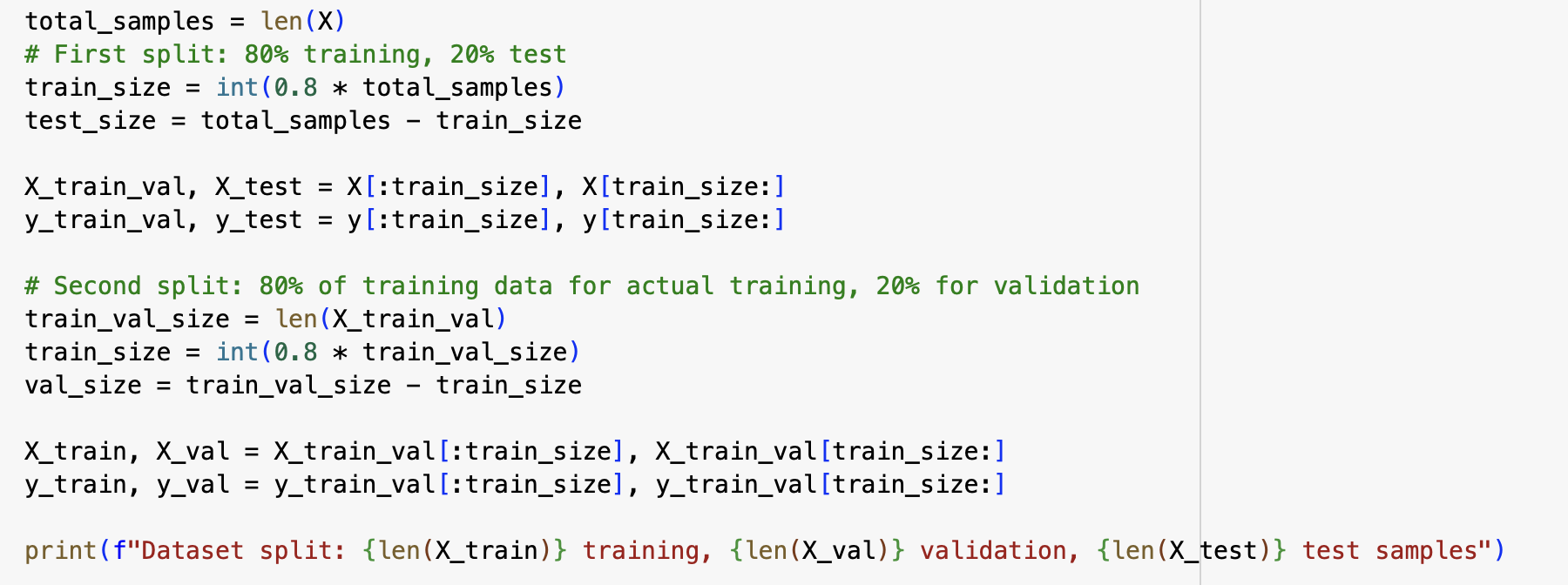
ChatGPT advised us to create input sequences which we used to create sliding window of 5 days to predict the next day’s case count.



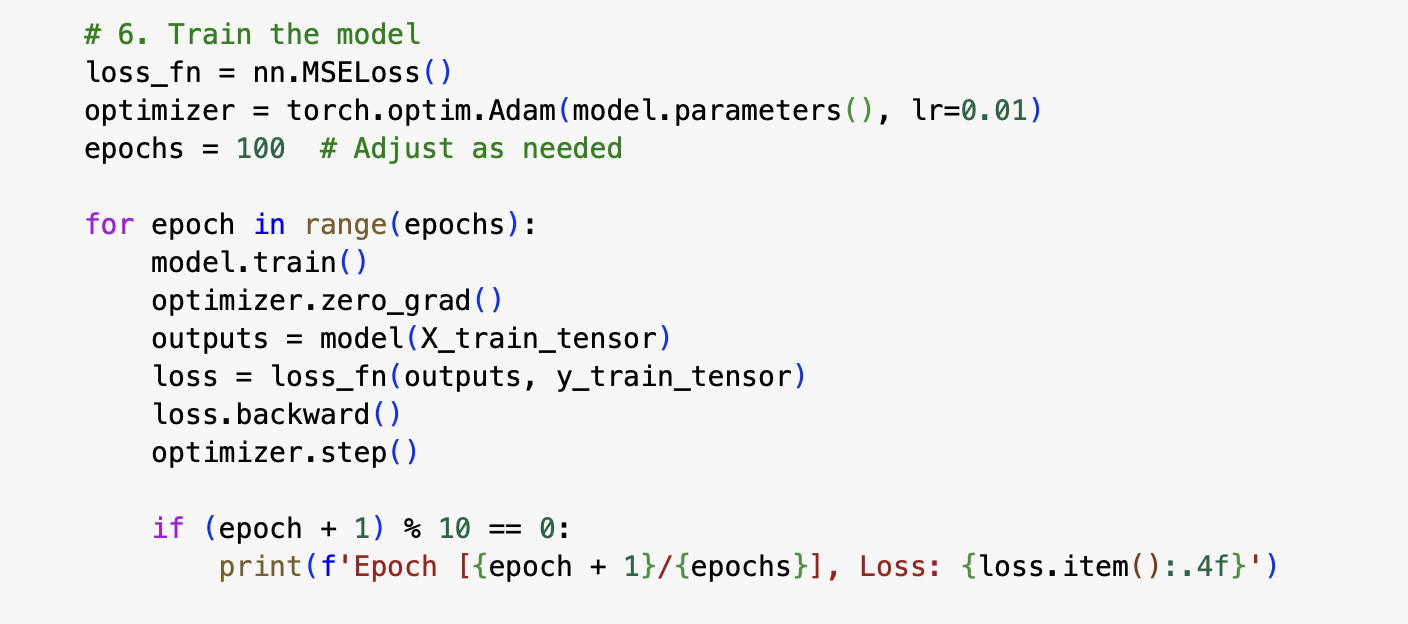
We then defined the LSTM model. ChatGPT also provided us with a simple 1-layer LSTM with 32 or 64 hidden units, followed by a fully connected output layer.



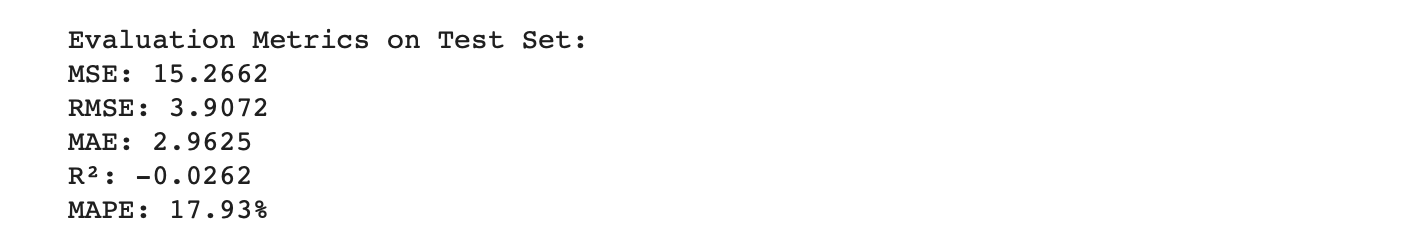
We then split the the data 80 20. 80 for the training set and 20 for the test set and then we split the 80 of the training data into 80 for actual training and 20 for validation.



We trained the model using 100 epoch using Mean Squared Error (MSE) loss and the Adam optimizer also referenced from ChatGPT.



To check our models performance we used certain evaluation metrics. Since our problem was a regression problem (prediciting a number, not a category) we used Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) which is more robust to outliers, Mean Squared Error (MSE), Mean Absolute Perentage Error (MAPE) and R-squared (Co-efficient of determination) showing how well the model explains the variance.



To improve the model’s performance, we adjusted several key settings or hyperparameters and observed how they affected the results.

For Sequence Length we tried using 5, 10, and 15 previous days to predict the next day’s cases. Using more past days like 10 or 15 gave the model slightly more context, which helped it perform better.

For Hidden Units (complexity of the LSTM) we reduced the number of hidden units from 64 to 32. This made the model simpler and reduced overfitting, which slightly improved accuracy on unseen data.

With training epochs increasing the number of training epochs beyond 100 didn't lead to major improvement. Too many epochs started to make the model overfit.

We added more features like humidity and public awareness level. These features didn’t improve the prediction much, possibly because they don’t strongly affect the number of SARS cases. The model performed only slightly better with tuning, but its accuracy was still limited. This suggests that the main issue is the small dataset, not just the model settings. A larger dataset or a different model might be needed for better results.

Lastly, to make our findings more interactive and accessible, we used **Streamlit** to build a simple web app that displays the model's predictions and performance. This allowed us to visualize SARS case trends and model outputs directly in a browser, making it easier for others to explore the results. We used code examples and guidance provided by **Claude AI** to help set up the Streamlit interface and integrate it with our trained LSTM model.

**Note:** Large Language Models (LLMs) such as ChatGPT and Claude AI were consulted to guide the code implementation and the udnerstanding of the model in the machine learning algorithm as well as debugging.

**References**

**Smith, J. (2020, June 15). The math behind LSTM.** Towards Data Science**.** <https://towardsdatascience.com/the-math-behind-lstm-9069b835289d/>​

Five practical applications of the LSTM model for time series with code\*. Towards Data Science. <https://towardsdatascience.com/five-practical-applications-of-the-lstm-model-for-time-series-with-code-a7aac0aa85c0/>

Brownlee, J. (n.d.). Time series forecasting with LSTM neural network Python. Towards Data Science. <https://towardsdatascience.com/time-series-forecasting-with-lstm-neural-network-python-3ee4f6f6f23f>

TensorFlow. (n.d.). Time series forecasting with recurrent neural networks. <https://www.tensorflow.org/tutorials/structured_data/time_series>

Scikit-learn. (n.d.). Preprocessing data. <https://scikitlearn.org/stable/modules/preprocessing.html>

Lundberg, S. M., & Lee, S.-I. (n.d.). SHAP (SHapley Additive exPlanations). SHAP Documentation. <https://shap.readthedocs.io/en/latest/>

Data preprocessing for machine learning. Towards Data Science. <https://towardsdatascience.com/data-preprocessing-for-machine-learning-5fa2a8fbd318>

ChatGPT

Claude AI