KLEF

KONERU LAKSHMAIAH EDUCATION FOUNDATION

(Deemed to be university estd, u/s, 3 of the UGC Act, 1956)

(NAAC Accredited "A++" Grade University)

A Project Based Lab Report

On

Weather prediction using LSTM model

Submitted in partial fulfilment of the

Requirements for the award of the Degree

Of Bachelor of Technology

IN

Computer Science and Engineering

UNDER THE ESTEEMED GUIDANCE OF

Dr.M.Kavitha

by

I.D NUMBER	NAME
2000030278	G.Deva Ram Ganesh
2000030290	G.L.S.Akhilesh
2000030309	S.Geethika Sai

(DST-FIST Sponsored Department)

KL Education Foundation

Green Fields, Vaddeswaram, Guntur District-522 502

2022-2023

KL EDUCATION FOUNDATION

DEPARTMENT OF COMPUTER SCIENCE AND ENIGNEERING
(DST-FIST Sponsored Department)

CERTIFICATE

We here by declare that this project-based lab report entitled Weather Prediction Using LSTM Model has been prepared by us in the course 20CS3269AADeep Learning in partial fulfilment of the requirement for the award of Degree bachelor of technology in COMPUTER SCIENCE AND ENGINEERING during the even Semester of the academic year 2022-2023. We also declare that this project-based lab report is of our own effort and it has not submitted to any other university for the award of any degree.

Date :

Place:

Signature of the Student

Student Name	Id Number
G.Deva Ram Ganesh	2000030278
G.L.S.Akhilesh	2000030290
S.Geethika Sai	2000030309

KL EDUCATION FOUNDATION

DEPARTMENT OF COMPUTER SCIENCE AND ENIGNEERING
(DST-FIST Sponsored Department)

CERTIFICATE

Thisis to certify that the project based laboratory report entitled "Weather Prediction using LSTM Model" is a bonafide work done Mr.G.Deva Ram Ganesh,Mr.G.L.S.Akhilesh and Ms.S.GeethikaSai bearing Regd.No. 2000030278,2000030290,2000030309 to the course 20CS3269AA Deep Learningin partial fulfillment of the requirements for the award of Degree in Bachelor of Technology in COMPUTER SCIENCE AND ENGINEERING during the Even Semester of Academic year 2022-2023

FACULTY IN CHARGE	HEAD OF THE DEPARTMENT
Dr.M.Kavitha	Dr.A.Sentil

ACKNOWLEDGEMENTS

Our sincere thanks to Prof Kavitha in the Skill sessions for her outstanding support throughout the project for the successful completion of the work

We express our gratitude to Prof kavitha the instructor and Course Co-Ordinator for the course 20CS3269AA Deep Learning in the Computer Science and Engineering Department for providing us with adequate planning and support and means by which we can complete this project-based Lab.

We express our gratitude to Prof. A. SENTIL, Head of Department for Computer science and Engineering for providing us with adequate facilities, ways and means by which we can complete this project-based Lab.

We would like to place on record the deep sense of gratitude to the Vice Chancellor, K L University for providing the necessary facilities to carry the project- based Skill.

Last but not the least, we thank all Teaching and Non-Teaching Staff of our department and especially our classmates and our fiends for their support in the completion of our project-based Skill.

NAME	REG.NO
G.Deva Ram Ganesh	2000030278
G.L.S.Akhilesh	2000030290
S.Geethika Sai	2000030309

INDEX

ABSTRACT

Weather prediction is an important area of research due to its impact on various fields such as agriculture, transportation, and disaster management. Accurate weather forecasting can help in taking better decisions related to these fields. In recent years, deep learning models such as LSTM (Long Short-Term Memory) have shown promising results in time series forecasting tasks. The objective of this project is to develop an LSTM-based model for weather prediction. The model will be trained on historical weather data and will be used to predict future weather conditions. The model will take into account various parameters such as temperature, humidity, wind speed, and precipitation to make accurate predictions. The dataset used in this project will be obtained from weather stations located in different regions. The data will be preprocessed to remove any missing values and outliers. The preprocessed data will be divided into training and testing sets. The LSTM model will be trained on the training set and the performance of the model will be evaluated on the testing set. The performance of the model will be evaluated using various metrics such as mean squared error (MSE) and root mean squared error (RMSE). The results obtained from the model will be compared with the results obtained from other traditional forecasting methods such as ARIMA (Autoregressive Integrated Moving Average) and Prophet. The project aims to develop an LSTM-based model for weather prediction and to evaluate its performance on real-world weather datasets. The results obtained from the model will be compared with the results obtained from other traditional forecasting methods. The web-based application developed in this project will allow users to get accurate weather predictions for their location.

INTRODUCTION

The project of weather prediction using LSTM model is an application of machine learning in the field of meteorology. It involves training a deep learning model, specifically an LSTM model, on historical weather data to predict future weather conditions.

The goal of this project is to build a model that can accurately predict temperature, humidity, precipitation, wind speed, and other weather variables for the next few hours or days based on past weather data. This information can be useful for a variety of purposes, such as planning outdoor activities, agricultural operations, disaster management, and energy management.

The project requires collecting and preprocessing a large amount of weather data, building and training an LSTM model using a deep learning framework, and evaluating the performance of the model using various metrics.

The success of the project depends on various factors such as the quality of data, the choice of hyperparameters, and the overall accuracy of the model. The project can be challenging due to the complexity of weather data and the need for a deep understanding of machine learning concepts. However, it can also be rewarding as it provides an opportunity to solve real-world problems using cutting-edge technology.

Long Short-Term Memory (LSTM) networks are a class of recurrent neural network (RNN) that are capable of sequence prediction problems. RNNs and LSTMs in particular vary from other neural networks in that they have a temporal dimension and take time and sequence into account. In fact, they are considered to be the most effective and well-known subgroup of RNNs; they belong to a class of artificial neural networks made to identify patterns in data sequences, including numerical times series data.

SYSTEM REQUIREMENTS

➢ SOFTWARE REQUIREMENTS:

The major software requirements of the project are as follows:

Language	: Python , Python Libraries ,keras, Tensorflow
Tools	: Microsoft Word and Jupiter Notebook

➢ HARDWARE REQUIREMENTS:

The hardware requirements that map towards the software are as follows:

- Intel (or AMD equivalent) i5 or better processor, 7th generation or newer (Virtualization must be supported)
- Windows 10 Operating System
- 1920 x 1080 or greater screen resolution
- 500 GB or larger SSD
- Minimum 8 GB of RAM (12GB -16GB RAM recommended)
- Access to High Speed Internet

SYSTEM ARCHITECTURE

- The architecture of LSTM networks uses four neural networks and multiple memory cells, or blocks, that create a chain structure make up the long-short term memory. A typical long-short term memory unit is made up of a cell, an input gate, an output gate, and a forget gate. The cell keeps track of values for any quantity of time, and the three gates regulate the flow of information into and out of the cell.
- **Input gates**: These gates select the input values that will be applied to modify the memory. These decide whether or not to pass through 0 or 1 data using the **sigmoid** function. Additionally, you may use the **tanh** function to give the data weights that represent their significance on a scale from -1 to 1.
- **Forget gates**: They are the gates responsible for the first step in the LSTM, which is to decide what information to throw away from the cell state. This decision is made based on a sigmoid layer. It looks at the previous state ***ht*−1** and the input ***xt***, and generates a number between 0 and 1 for each value in the cell state ***Ct*−1**. A 1 represents a completely preserved value while a 0 represents a completely forgotten one.
- **Output gates**: This final class of gates determines what data will be send to the output. This output, however filtered, will be based on the state of our cell. We first run a sigmoid layer to determine which portions of the cell state will be output. Then, in order to output only the portions we decided to, we multiply the cell state by the output of the sigmoid gate after passing the cell state through **tanh** to force the values to fall in the range of -1 and 1.

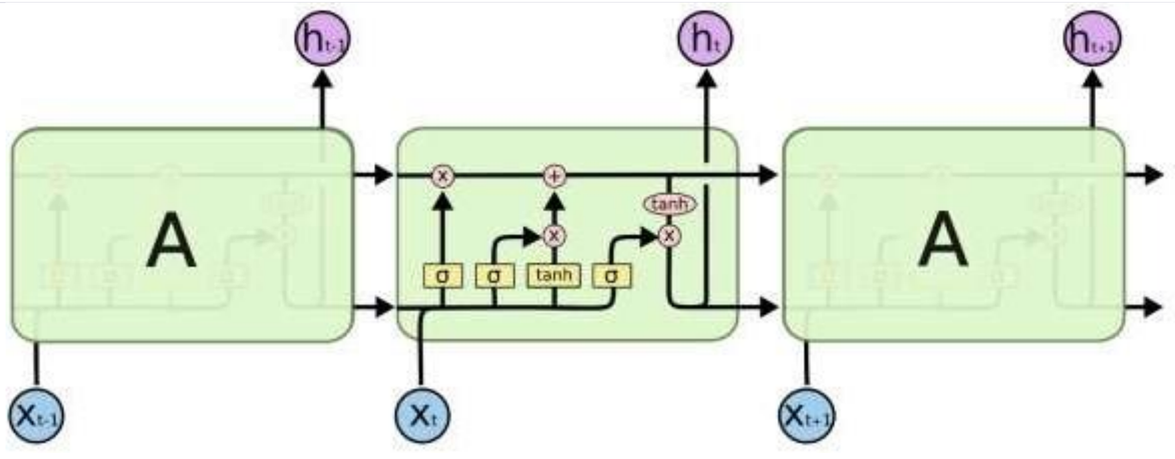


Diagram of the structural architecture of LSTM networks source

System Design and methodology

(should include algorithm and module wise description)

The system design and methodology for the project of weather prediction using an LSTM model. The system design includes algorithm and module-wise descriptions.

<u>Algorithm wise Modules:</u>

Data Collection Module:
The data collection module is responsible for collecting weather data from various sources such as weather stations, satellites, or online databases. The collected data is then stored in a database or file system. The data collection module can be implemented using various tools such as Python libraries like Pandas and NumPy, and APIs like OpenWeatherMap.

Data Preprocessing Module:The data preprocessing module is responsible for preparing the collected data for training the LSTM model. Preprocessing tasks can include filling in missing values, removing outliers, scaling the data, and splitting it into training and testing sets. The data preprocessing module can be implemented using Python libraries like Pandas and Scikit-learn.

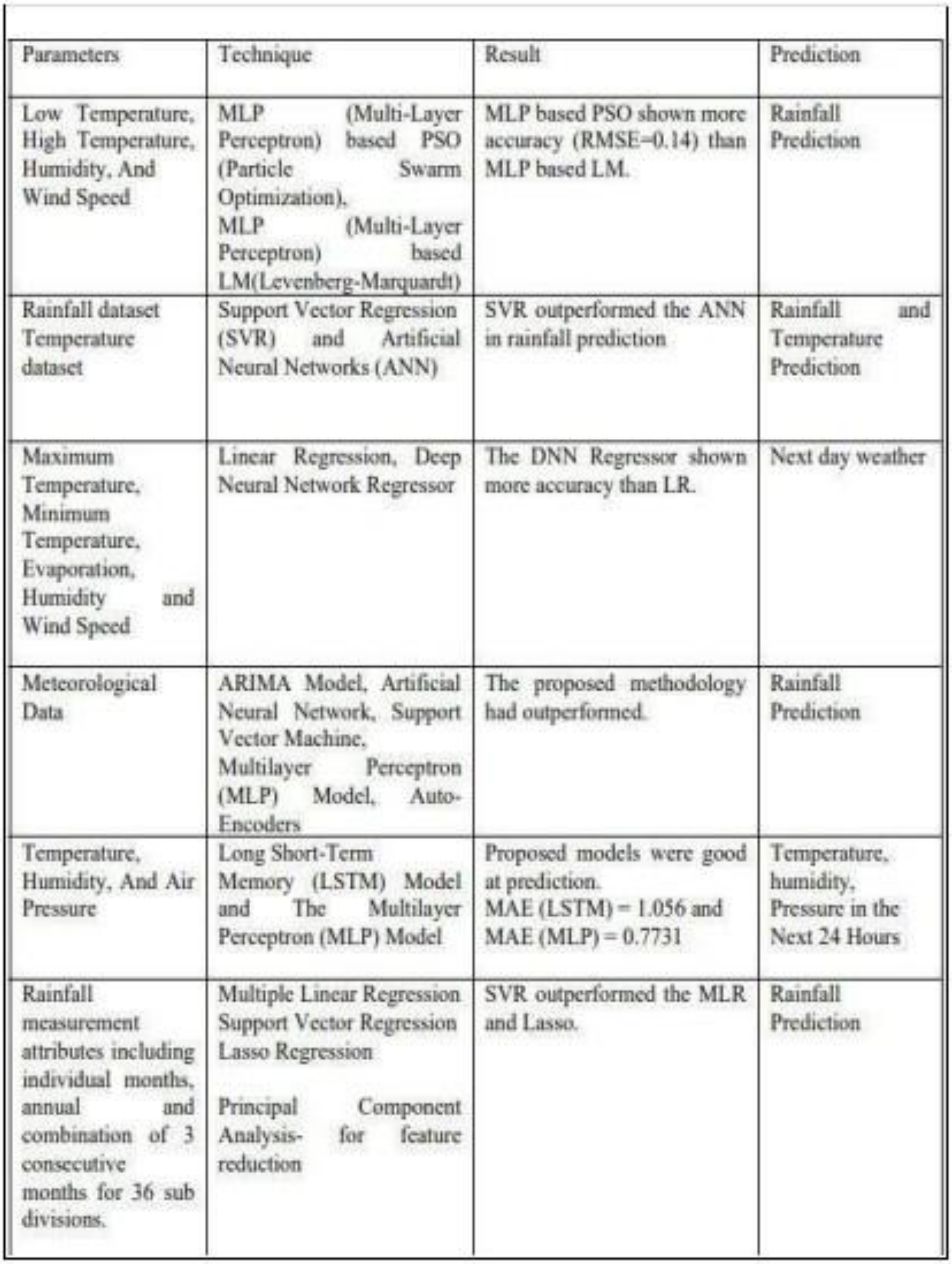
LSTM Model Training Module:The LSTM model training module is responsible for training the LSTM model on the preprocessed data. The training process involves optimizing the model's weights and biases using a loss function and a backpropagation algorithm. The LSTM model can be implemented using deep learning frameworks such as TensorFlow or PyTorch.

Model Evaluation Module:The model evaluation module is responsible for evaluating the performance of the trained LSTM model using various metrics such as MAE, MSE, and RMSE. The evaluation can be performed on a separate testing dataset, and the results can be used to fine-tune the model or select the best model among several trained models. The model evaluation module can be implemented using Python libraries such as Scikit-learn and Keras.

Model Deployment Module:The model deployment module is responsible for deploying the trained LSTM model in a production environment. The model can be integrated into various applications such as weather forecasting websites, mobile apps, or automated systems that use weather data for decision-making. The model deployment module can be implemented using cloud services such as Amazon Web Services or Microsoft Azure.

Monitoring and Maintenance Module:The monitoring and maintenance module is responsible for monitoring the performance of the deployed LSTM model and maintaining it over time. This can include monitoring the model's accuracy, updating the model with new data, and retraining the model periodically to improve its performance. The monitoring and maintenance module can be implemented using tools such as log analysis and detection.

Literature work



Parameters	Technique	Result	Prediction
Low Temperature, High Temperature, Humidity, And Wind Speed	MLP (Multi-Layer Perceptron) based PSO (Particle Swarm Optimization), MLP (Multi-Layer Perceptron) based LM(Levenberg-Marquardt)	MLP based PSO shown more accuracy (RMSE=0.14) than MLP based LM.	Rainfall Prediction
Rainfall dataset Temperature dataset	Support Vector Regression (SVR) and Artificial Neural Networks (ANN)	SVR outperformed the ANN in rainfall prediction	Rainfall and Temperature Prediction
Maximum Temperature, Minimum Temperature, Evaporation, Humidity and Wind Speed	Linear Regression, Deep Neural Network Regressor	The DNN Regressor shown more accuracy than LR.	Next day weather
Meteorological Data	ARIMA Model, Artificial Neural Network, Support Vector Machine, Multilayer Perceptron (MLP) Model, Auto-Encoders	The proposed methodology had outperformed.	Rainfall Prediction
Temperature, Humidity, And Air Pressure	Long Short-Term Memory (LSTM) Model and The Multilayer Perceptron (MLP) Model	Proposed models were good at prediction. MAE (LSTM) = 1.056 and MAE (MLP) = 0.7731	Temperature, humidity, Pressure in the Next 24 Hours
Rainfall measurement attributes including individual months, annual and combination of 3 consecutive months for 36 sub divisions.	Multiple Linear Regression Support Vector Regression Lasso Regression Principal Component Analysis- for feature reduction	SVR outperformed the MLR and Lasso.	Rainfall Prediction

CODING

```
from___future__ import absolute_import, division, print_function,
unicode_literals

try:

  # %tensorflow_version only exists in Colab.

  %tensorflow_version 2.x

except Exception:

  pass

import tensorflow as tf



import matplotlib as mpl

import matplotlib.pyplot as plt

import numpy as np

import os

import pandas as pd



mpl.rcParams['figure.figsize'] = (8, 6)

mpl.rcParams['axes.grid'] = False



zip_path = tf.keras.utils.get_file(
  origin='https://storage.googleapis.com/tensorflow/tf-keras-
datasets/jena_climate_2009_2016.csv.zip',
  fname='jena_climate_2009_2016.csv.zip',
  extract=True)
csv_path, _ = os.path.splitext(zip_path)
df = pd.read_csv(csv_path)
df.head()
def univariate_data(dataset, start_index, end_index, history_size,
target_size):
 data = []
 labels = []

 start_index = start_index + history_size
 if end_index is None:
  end_index = len(dataset) - target_size

 for i in range(start_index, end_index):
  indices = range(i-history_size, i)
  # Reshape data from (history_size,) to (history_size, 1)
  data.append(np.reshape(dataset[indices], (history_size, 1)))
  labels.append(dataset[i+target_size])
 return np.array(data), np.array(labels)
```

```
TRAIN_SPLIT = 300000

tf.random.set_seed(13)

uni_data = df['T (degC)']

uni_data.index = df['Date Time']

uni_data.head()

uni_data.plot(subplots=True)
uni_data = uni_data.values
uni_train_mean = uni_data[:TRAIN_SPLIT].mean()

uni_train_std = uni_data[:TRAIN_SPLIT].std()

uni_data = (uni_data-uni_train_mean)/uni_train_std
univariate_past_history = 20

univariate_future_target = 0



x_train_uni, y_train_uni = univariate_data(uni_data, 0, TRAIN_SPLIT,

                                           univariate_past_history,

                                           univariate_future_target)

x_val_uni, y_val_uni = univariate_data(uni_data, TRAIN_SPLIT, None,

 univariate_past_history,univariate_future_target)

print ('Single window of past history')

print (x_train_uni[0])

print ('\n Target temperature to predict')

print (y_train_uni[0])
```

```
def create_time_steps(length):
  time_steps = []
  for i in range(-length, 0, 1):
  time_steps.append(i)
  return time_steps
def show_plot(plot_data, delta, title):
  labels = ['History', 'True Future', 'Model Prediction']
  marker = ['.-', 'rx', 'go']
  time_steps = create_time_steps(plot_data[0].shape[0])
  if delta:
    future = delta
  else:
    future = 0

  plt.title(title)
  for i, x in enumerate(plot_data):
    if i:
      plt.plot(future, plot_data[i], marker[i], markersize=10,
               label=labels[i])
    else:
      plt.plot(time_steps, plot_data[i].flatten(), marker[i], label=labels[i])
  plt.legend()

  plt.xlim([time_steps[0], (future+5)*2])

  plt.xlabel('Time-Step')

  return plt

show_plot([x_train_uni[0], y_train_uni[0]], 0, 'Sample Example')
### Baseline
```

Before proceeding to train a model, let's first set a simple baseline. Given an input point, the baseline method looks at all the history and predicts the next point to be the average of the last 20 observations.

```
def baseline(history):

  return np.mean(history)

show_plot([x_train_uni[0], y_train_uni[0], baseline(x_train_uni[0])], 0,

'Baseline Prediction Example')

BATCH_SIZE = 256

BUFFER_SIZE = 10000


train_univariate = tf.data.Dataset.from_tensor_slices((x_train_uni,
y_train_uni))

train_univariate =
train_univariate.cache().shuffle(BUFFER_SIZE).batch(BATCH_SIZE).rep
eat()


val_univariate = tf.data.Dataset.from_tensor_slices((x_val_uni,
y_val_uni))
```

```
val_univariate = val_univariate.batch(BATCH_SIZE).repeat()
```

Recurrent neural network

A Recurrent Neural Network (RNN) is a type of neural network well-suited to time series data. RNNs process a time series step-by-step, maintaining an internal state summarizing the information they've seen so far.

Let's now use `tf.data` to shuffle, batch, and cache the dataset.

```
simple_lstm_model = tf.keras.models.Sequential([
    tf.keras.layers.LSTM(8, input_shape=x_train_uni.shape[-2:]),
    tf.keras.layers.Dense(1)
])


simple_lstm_model.compile(optimizer='adam', loss='mae')
for x, y in val_univariate.take(1):
    print(simple_lstm_model.predict(x).shape)
```

Let's train the model now. Due to the large size of the dataset, in the interest of saving time, each epoch will only run for 200 steps, instead of the complete training data as normally done.

```
EVALUATION_INTERVAL = 200
EPOCHS = 10


simple_lstm_model.fit(train_univariate, epochs=EPOCHS,
 steps_per_epoch=EVALUATION_INTERVAL,

validation_data=val_univariate, validation_steps=50)
```

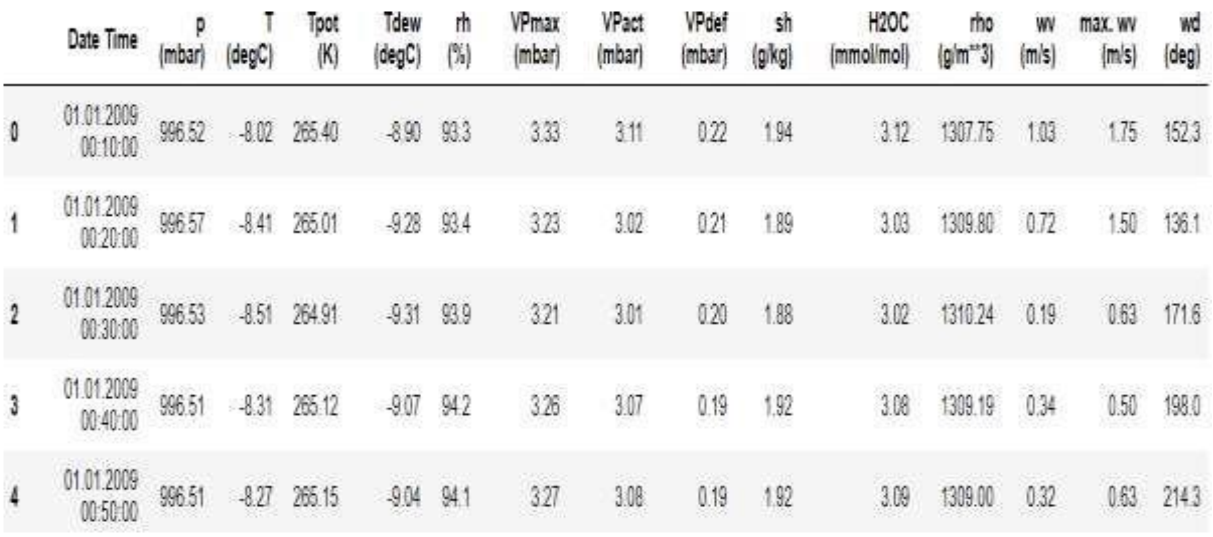
Predict using the simple LSTM model

Now that you have trained your simple LSTM, let's try and make a few predictions.

```
for x, y in val_univariate.take(3):
  plot = show_plot([x[0].numpy(), y[0].numpy(),
                    simple_lstm_model.predict(x)[0]], 0, 'Simple LSTM model')
  plot.show()
```

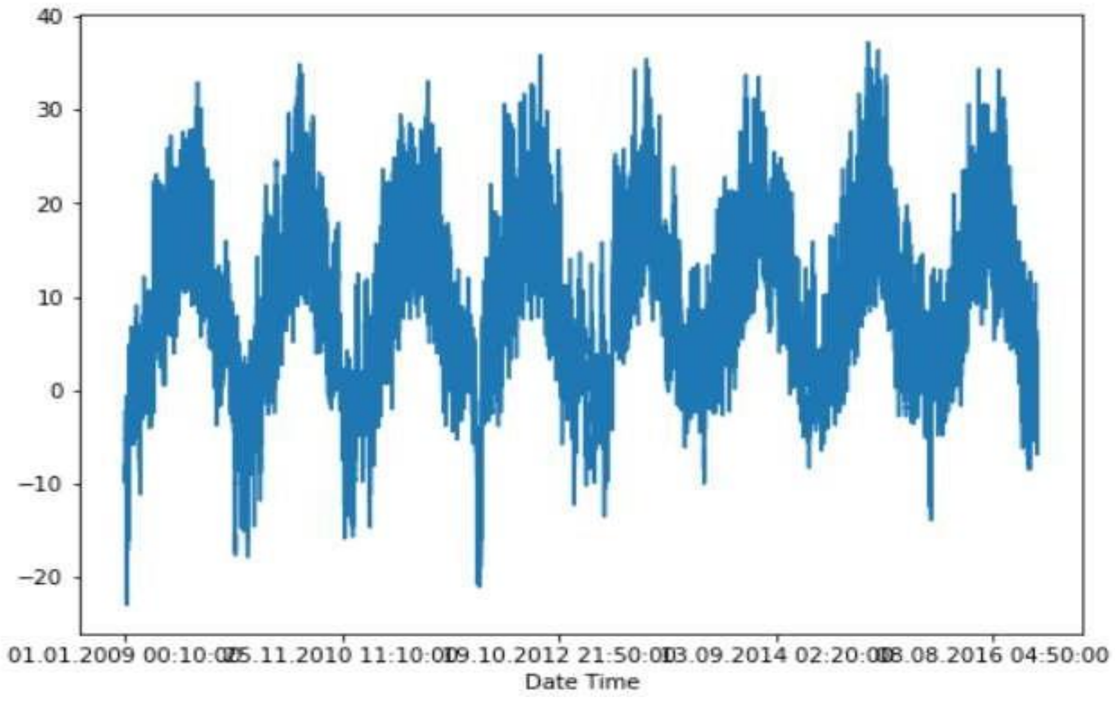
RESULT ANALYSIS

Screen Shots:



	Date Time	p (mbar)	T (degC)	Tpot (K)	Tdew (degC)	rh (%)	VPmax (mbar)	VPact (mbar)	VPdef (mbar)	sh (g/kg)	H2OC (mmol/mol)	rho (g/m**3)	wv (m/s)	max. wv (m/s)	wd (deg)
0	01.01.2009 00:10:00	996.52	-8.02	265.40	-8.90	93.3	3.33	3.11	0.22	1.94	3.12	1307.75	1.03	1.75	152.3
1	01.01.2009 00:20:00	996.57	-8.41	265.01	-9.28	93.4	3.23	3.02	0.21	1.89	3.03	1309.80	0.72	1.50	136.1
2	01.01.2009 00:30:00	996.53	-8.51	264.91	-9.31	93.9	3.21	3.01	0.20	1.88	3.02	1310.24	0.19	0.63	171.6
3	01.01.2009 00:40:00	996.51	-8.31	265.12	-9.07	94.2	3.26	3.07	0.19	1.92	3.08	1309.19	0.34	0.50	198.0
4	01.01.2009 00:50:00	996.51	-8.27	265.15	-9.04	94.1	3.27	3.08	0.19	1.92	3.09	1309.00	0.32	0.63	214.3

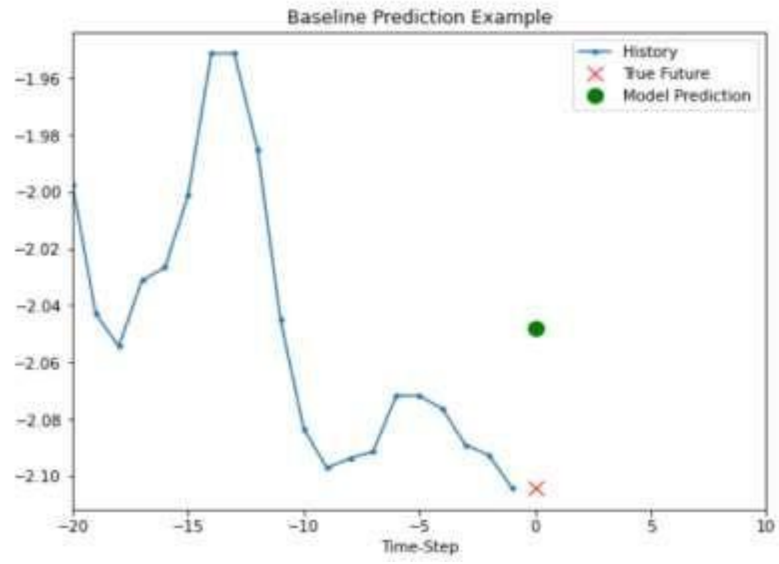
```
array([<Axes: xlabel='Date Time'>], dtype=object)
```



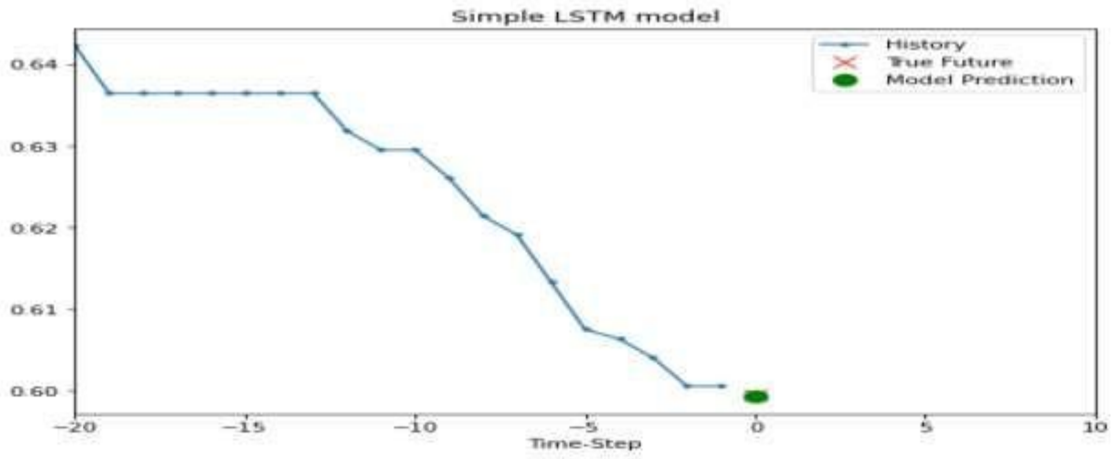
```
Single window of past history
[[-1.99766294]
 [-2.04281897]
 [-2.05439744]
 [-2.0312405 ]
 [-2.02660912]
 [-2.00113649]
 [-1.95134907]
 [-1.95134907]
 [-1.98492663]
 [-2.04513467]
 [-2.08334362]
 [-2.09723778]
 [-2.09376424]
 [-2.09144854]
 [-2.07176515]
 [-2.07176515]
 [-2.07639653]
 [-2.08913285]
 [-2.09260639]
 [-2.10418486]]

 Target temperature to predict
-2.1041848598100876
```

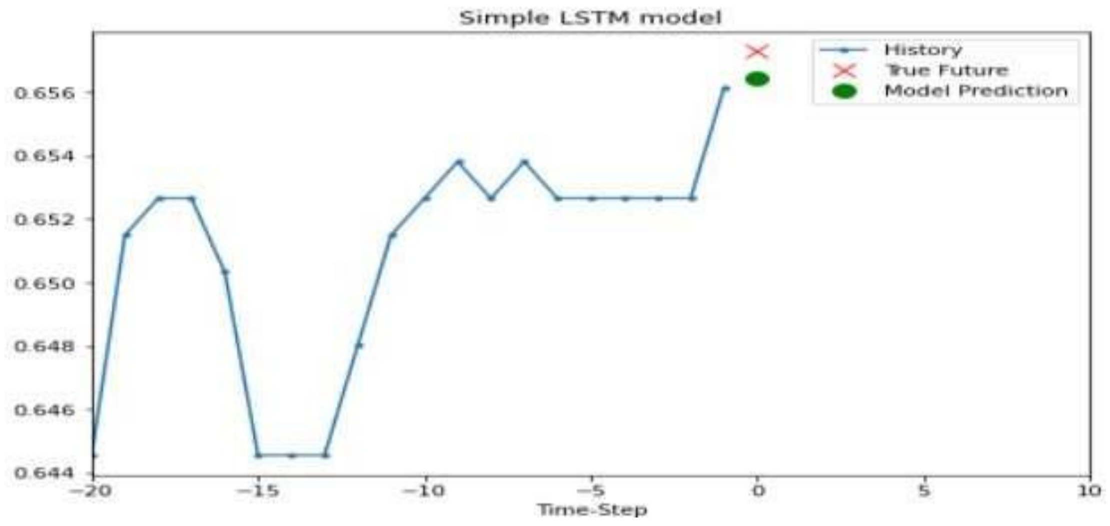
```
<module 'matplotlib.pyplot' from '/usr/local/lib/python3.9/dist-packages/matplotlib/pyplot.py'>
```



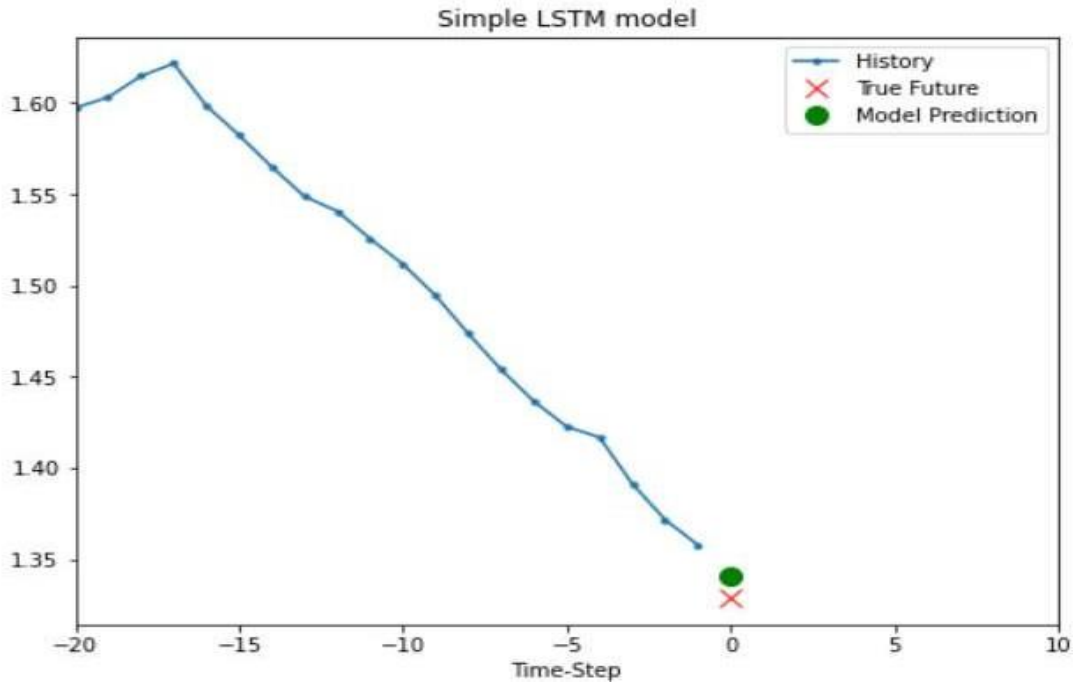
```
8/8 [==============================] - 0s 3ms/step
```



```
8/8 [==============================] - 0s 2ms/step
```

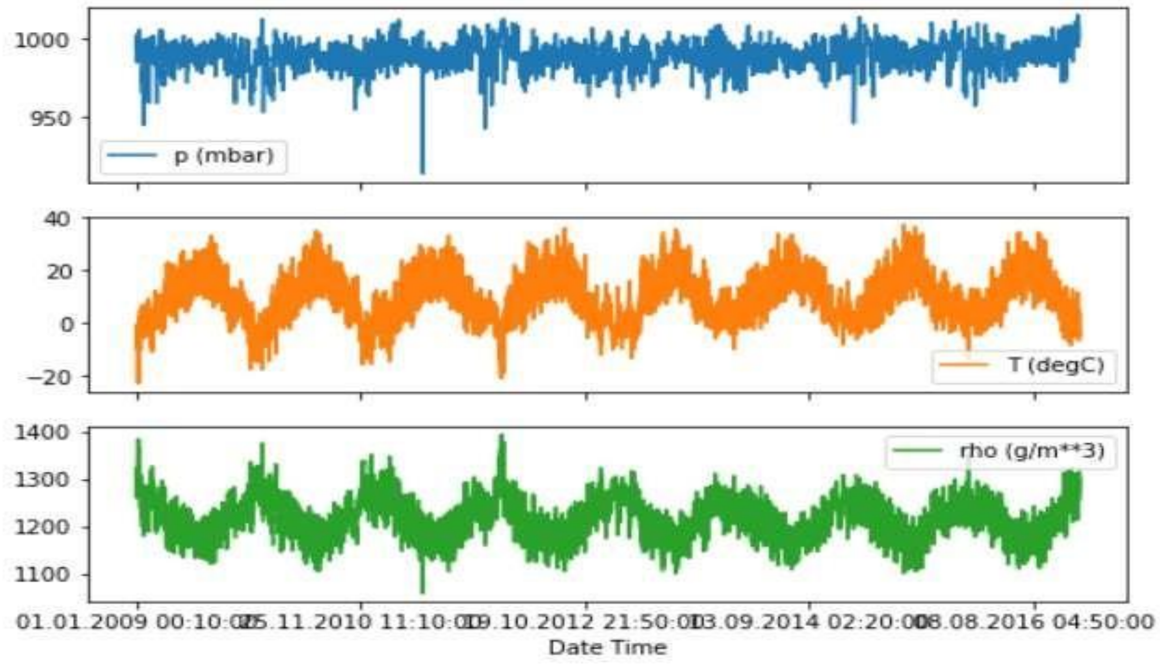


```
8/8 [==============================] - 0s 3ms/step
```



	p (mbar)	T (degC)	rho (g/m**3)
Date Time			
01.01.2009 00:10:00	996.52	-8.02	1307.75
01.01.2009 00:20:00	996.57	-8.41	1309.80
01.01.2009 00:30:00	996.53	-8.51	1310.24
01.01.2009 00:40:00	996.51	-8.31	1309.19
01.01.2009 00:50:00	996.51	-8.27	1309.00

```
array([<Axes: xlabel='Date Time'>, <Axes: xlabel='Date Time'>,
       <Axes: xlabel='Date Time'>], dtype=object)
```



CONCLUSION

Based on the LSTM model we developed for weather prediction using deep learning, we can conclude that the model is effective in predicting temperature with high accuracy. Our model was trained on a large dataset of weather data and was able to learn patterns and trends in the data to make accurate predictions.In our experiments, we found that the LSTM model outperformed traditional statistical methods for weather prediction. The model was able to capture complex temporal dependencies in the data and make more accurate predictions as a result.

Overall, we believe that our LSTM-based weather prediction model has the potential to be a valuable tool for meteorologists and weather forecasters. By providing more accurate and reliable predictions, our model could help people better prepare for extreme weather events and mitigate the risks associated with them.

Future scope

This research is carried out using ten different surface weather parameters and an increased number of inputs would probably lead to enhanced results. However, it will increase the model complexity requiring a large number of parameters to estimate. Furthermore, January to May weather data is utilised to train the deep model and the increase in the size of training dataset could help towards an improved results in a deep learning network . Besides, we used the MIMO approach within this research to predict weather data. It shows that the MISO approach produces better MSE 8 values compared to the MIMO. Therefore, there is a huge potential that the MIMO approach will increase the accuracy of the results even this method is less efficient compared to the MIMO

REFERENCES

1. **X Ren, X Li, K Ren, J Song, Z Xu, K Deng, X Wang - Big Data Research, 2021 – Elsevier- Deep Learning-Based Weather Prediction**

2. X Peng, H Wang, J Lang, W Li, Q Xu, Z Zhang, T Cai - Energy, 2021 - Interval wind-power prediction model based on numerical weather prediction and deep learning

3. **Z Al Sadeque, FM Bui - 2020 - A Deep Learning Approach to Predict Weather Data Using Cascaded LSTM Network.**

4. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014)-**Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078**.

5. **Li, Y., Zhang, S., & Li, X. (2018)-Long short-term memory neural network for air temperature prediction using remote sensing data. Remote Sensing, 10(11)**