Final Project

Weather Prediction

Haritha Dhanlalji Parmar, Vyshnavi Reddy Mungi

School of Science and Engineering

Saint Louis University

Data Analytics and IOT

Professor Anna Marbut

Introduction

Weather prediction is the technique of forecasting future weather conditions for a specific area or region using scientific methodologies and equipment. Weather prediction is a complicated and dynamic process that requires the collecting of data from a variety of sources, including weather satellites, weather stations, and other sensors that monitor various weather variables such as temperature, air pressure, and humidity.

The information gathered from various sources is then examined with mathematical models, statistical methodologies, and computer simulations to provide forecasts of future weather conditions. Individuals and organizations can use these forecasts to plan for and respond to extreme weather disasters such as hurricanes, floods, and droughts.

Weather forecasting is also becoming more crucial as a result of climate change, which is creating more extreme weather occurrences. Accurate weather forecasts can assist people and governments in planning for and mitigating the consequences of severe catastrophes.

The project's goal is to deliver real-time weather predictions using machine learning algorithms and IoT devices, which will allow farmers, transportation businesses, and emergency response teams to make educated decisions depending on weather conditions. The initiative also intends to minimize the expenses of weather prediction by employing IoT devices for data collecting and analysis instead of traditional weather stations.

Overall, the weather prediction project is a promising use of IoT and machine learning technology that has the potential to significantly affect numerous businesses by delivering more accurate and timely weather forecasts.

Final Project: Weather Prediction

Data Source

The SMHI Weather dataset from the Weather Predictor project is the one you've chosen. This collection comprises historical weather data for several regions of Sweden gathered by the Swedish Meteorological and Hydrological Institute (SMHI).

The SMHI dataset covers a wide range of variables, including temperature, air pressure, precipitation, and wind speed, making it ideal for weather prediction using machine learning techniques. The dataset is derived from genuine IoT sensors and devices, verifying the data's validity. SMHI Weather dataset contains a large amount of data with 221k rows and 5 columns when taken as raw data.

In addition, the dataset contains a time series variable, which is required for time series analysis and forecasting with machine learning algorithms. The dataset may be used to train machine learning models like LSTM and Linear Regression to reliably forecast future weather conditions.

IOT System Design

Our Iot System Design consists of sensors, edge processing, data storage and processing, networking, user interface.

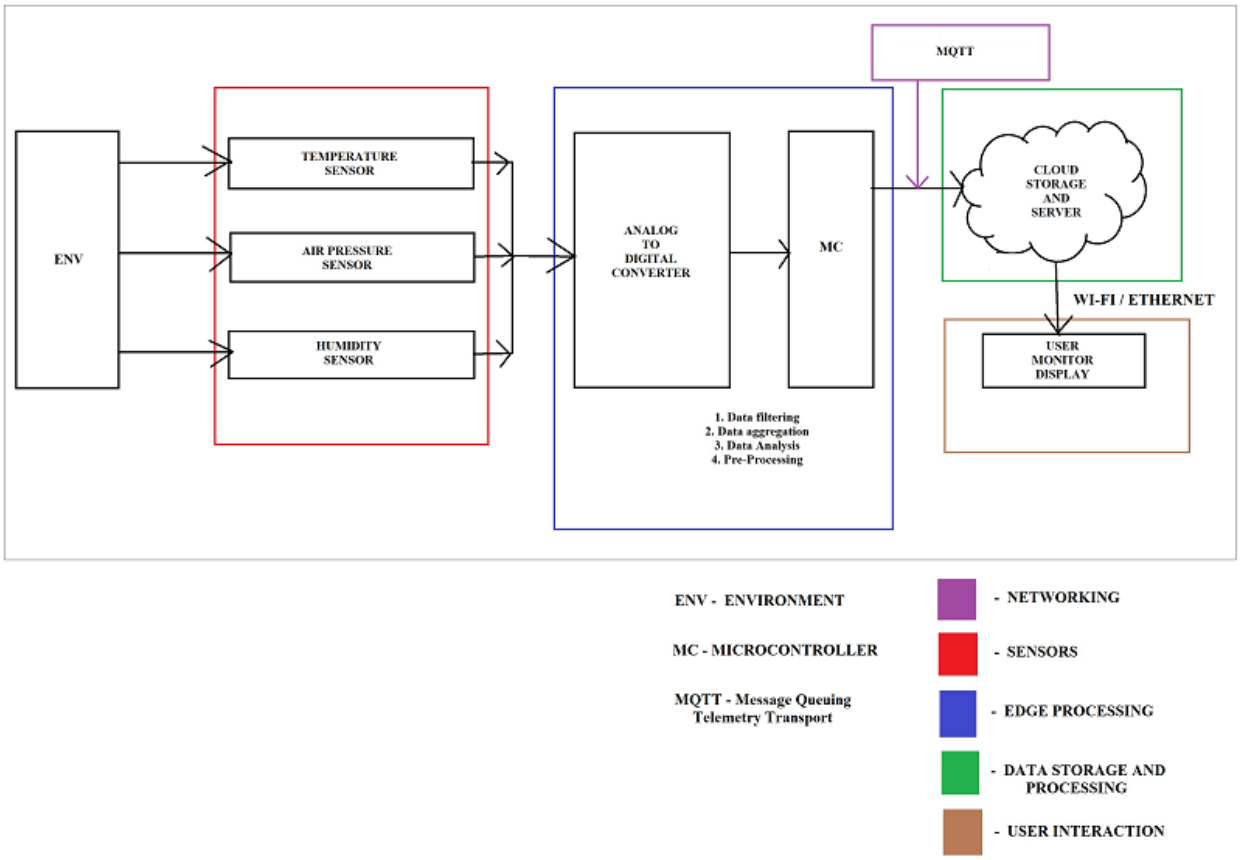

Fig. 1. Weather Prediction System Design

Sensors:

Temperature sensor: The project's temperature sensor is most likely a thermistor or a thermocouple. A thermistor is a type of resistor whose resistance varies with temperature. A thermocouple is a device made up of two distinct metals that are linked at one end. The voltage generated by the junction between the metals is proportional to the temperature difference between the two ends.

The temperature range of the sensor will be determined by the type of sensor utilized. Temperatures as low as -200°C and as high as 2000°C may be measured by certain thermistors

and thermocouples. However, at high temperatures, the sensor's accuracy and resolution typically decrease.

Temperature sensors are limited by their sensitivity to external influences such as electromagnetic interference and self-heating. Furthermore, thermistors and thermocouples may need to be calibrated to maintain accuracy over time.

Air Pressure sensor: The air pressure sensor utilized in the project is most likely a piezoresistive or capacitive sensor. A piezoresistive sensor detects changes in resistance induced by deformation of a thin-film diaphragm to measure pressure. A capacitive sensor detects changes in capacitance generated by the deformation of a flexible membrane to measure pressure.

The pressure range of the sensor will be determined by the type of sensor employed. Some air pressure sensors can measure pressures ranging from 0.1 kPa to 700 kPa. However, at high pressures, the sensor's accuracy and resolution typically decrease.

Air pressure sensors include limitations such as sensitivity to temperature changes and susceptibility to mechanical stress and vibration. Calibration may also be necessary to maintain accuracy over time.

Humidity Sensor: The humidity sensor utilized in the project is most likely a capacitive or resistive sensor. A capacitive humidity sensor detects variations in capacitance due by water vapor absorption into a thin-film polymer. A resistive humidity sensor monitors resistance variations induced by humidity-sensitive material changes.

The humidity range of the sensor will be determined by the type of sensor utilized. Some humidity sensors can detect relative humidity levels ranging from 0% to 100%. However, at high humidity levels, the sensor's accuracy and resolution will typically decrease.

Humidity sensors have limitations such as their sensitivity to temperature fluctuations and susceptibility to contamination by chemicals or dust. Calibration may also be necessary to maintain accuracy over time.

Microcontroller:

The microcontroller is a vital component of the IoT system that receives sensor data from numerous sensors. The raw data from the sensors is processed by the microcontroller, which converts the analog impulses to digital signals using an Analog to Digital Converter (ADC). The microcontroller conducts some initial data filtering and aggregation, which reduces the amount of data that must be communicated to the cloud server, lowering network traffic and delay. The microcontroller also monitors the system's power consumption and ensures that it runs effectively and dependably.

After processing the data, the microcontroller sends the aggregated and filtered data to the cloud server using the MQTT messaging protocol. The microcontroller must be designed to guarantee that the data is appropriately sent to the server and that any problems that may arise during transmission are handled. The microcontroller must also have enough memory and processing power to handle the volume of data being delivered. Overall, the microcontroller is critical in the IoT system, and its performance can have a major influence on the overall efficiency and dependability of the system.

Final Project: Weather Prediction

Edge processing:

Edge processing entails filtering, aggregating, and preparing data at the network's edge before sending it to a cloud server for additional analysis and storage. Data filtering is the process of reducing noise or outliers from raw data in order to guarantee that only relevant data is transferred to the cloud server. This reduces the amount of the dataset while simultaneously improving the accuracy of the analysis by deleting extraneous data.

Data aggregation is the process of merging data from various sensors into a single dataset that may be used for larger-scale analysis. In a weather forecast system, for example, data from several sensors positioned around a city may be aggregated to create a more thorough picture.

Data preparation entails converting raw data into a format suitable for analysis and storage. Normalizing the data, changing it to a certain data type, or scaling the data to a specified range are all examples of operations that may be performed. Preprocessing data at the edge can assist to minimize the processing time and resources needed for cloud analysis and storage.

Networking:

MQTT (Message Queuing Telemetry Transport) is a lightweight messaging protocol intended for use in Internet of Things (IoT) devices with limited processing power and memory. It is built on a publish/subscribe approach, in which devices can send messages to a central broker, who subsequently sends the messages to all devices that have subscribed to the relevant topics. This method enables effective and flexible device connection, as well as decreased bandwidth and power consumption. MQTT may also be protected using Transport Layer Security (TLS) to safeguard data in transit and supports Quality of Service (QoS) levels to ensure

reliable message delivery. MQTT, in general, provides a dependable, low-latency, and scalable networking solution for IoT systems.

Cloud Storage and Server:

The cloud storage and server are critical in storing and analyzing the huge volume of data produced by the IoT system. The data is easily available with cloud-based storage, and the cloud server can analyse the data in real-time, making it easier to spot trends and patterns in the data. The cloud server also supports the use of machine learning techniques such as LSTM and Linear Regression, which are trained on data and used to forecast future weather conditions. This helps users to make educated decisions based on the system's data analysis and forecasts.

Visualizations:

Visualization is important in the weather prediction project since it helps to graphically communicate the insights and forecasts produced from the machine learning models. Data visualizations built using technologies like Tableau may aid in the identification of trends, patterns, and correlations that may not be immediately obvious in raw data. Decision-makers may rapidly identify areas that require attention and take informed choices by portraying data in various forms such as charts, graphs, and maps. Visualization is also used in the weather prediction project to communicate insights and forecasts to stakeholders in an easy-to-understand style, allowing them to make educated decisions based on the facts given.

Data Cleaning

Data Cleaning of a project report explains how the time series data is cleaned and investigated. An explanation of the data cleaning procedure, which often entails getting rid of duplicates, fixing mistakes, and dealing with missing data, should come first in this part. Next, exploratory data analysis (EDA) methods can be used to examine the cleaned data.

The "Load and clean the data step entails loading a time series dataset into a pandas DataFrame and carrying out fundamental cleaning operations, such as combining the Date and Time columns into a single Datetime column and checking that the data types of the other columns are accurate.

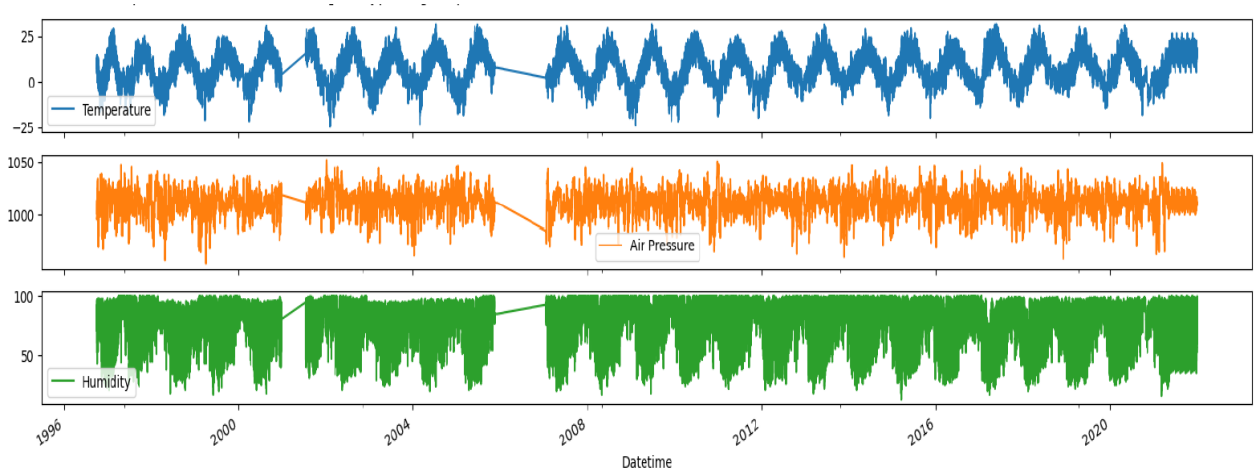

Fig. 2. Data Visualized after cleaning and processing.

To better comprehend patterns, correlations, and trends, the "Visualizing Data" component of data analysis focuses on developing powerful visual representations of the data using line charts and computing monthly averages.

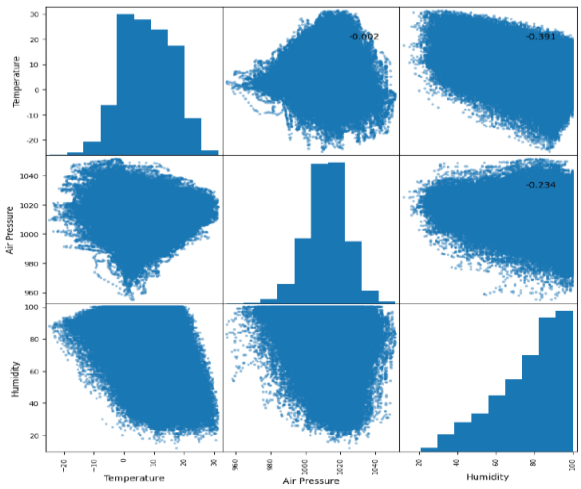

Fig. 3. Correlation Matrix for three variables Temperature, Air Pressure and Humidity.

The emphasis in the "Data Covariance and Correlation" area of data analysis is on understanding the connections between variables in the dataset. Covariance and correlation are two statistical metrics that may be used to estimate the strength of a relationship between two variables.

Linear Regression Model

Linear regression is a statistical approach for modeling the connection between one or more independent variables and a dependent variable. It is utilized in this example to forecast temperature values based on data from IoT devices. The data is put into a pandas DataFrame and preprocessed by converting it to epoch/unix time.

A rolling window of data is used to train a linear regression model. The linear regression technique and sample weights are used to train the model, which are changed by a parameter called mu. Different mu values are examined to find the one that produces the greatest model performance.

In [5]:
```
# Setting the ph to be 10 days, as the data set values are recorded with an interval of 1 hr

ts = pd.DataFrame(df.unix)
ys = pd.DataFrame(df.Temperature)
ph = 10*24 #100 days
ph_index = round(ph/24)
mu = 0.01
#let's limit the number of samples in our model to 5000 just for speed
n_s = 5000
# Arrays to hold predicted values
tp_pred = np.zeros(n_s-1)
yp_pred = np.zeros(n_s-1)
```

In [6]:
```
#using linear regression to predict a rolling temperature for our dataset

for i in range(2, n_s+1):
    ts_tmp = ts[0:i]
    ys_tmp = ys[0:i]

    ns = len(ys_tmp)
    weights = np.ones(ns)*mu
    for k in range(ns):
        weights[k] = weights[k]**k
    weights = np.flip(weights, 0)
    #perform linear regression on "available" data using the mu-adjusted weights
    lm_tmp = LinearRegression()
    model_tmp = lm_tmp.fit(ts_tmp, ys_tmp, sample_weight=weights)
    m_tmp = model_tmp.coef_
    q_tmp = model_tmp.intercept_
    #use ph to make the model prediction according to the prediction time
    tp = ts.iloc[i-1,0] + ph #predicted time
    yp = m_tmp*tp + q_tmp
    tp_pred[i-2] = tp
    yp_pred[i-2] = yp
```

Fig. 4. Linear Regression Code to predict temperature.

The model's performance is assessed by measuring the mean squared error (MSE) between the anticipated and actual temperature readings. A lower MSE suggests that the model performed better.

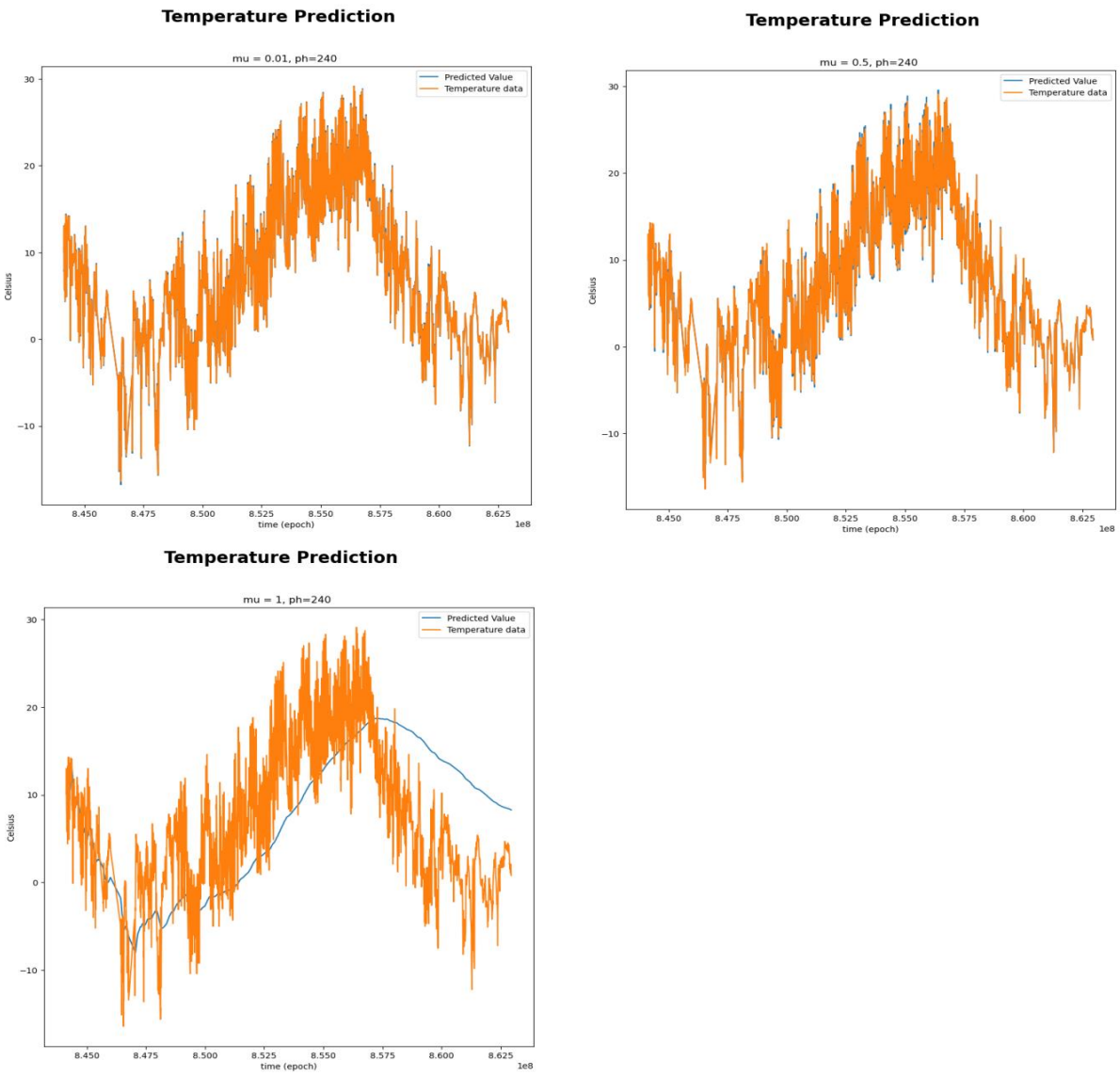

Fig. 5. Various temperature prediction visualizations for different values of mu.

The following is a report on the performance of the linear regression model with various mu values:

- The model performs well with an MSE of 0.0131 when mu=0.01. This means that the projected temperatures are close to the actual temperatures.

- The model performs badly when mu=0.5, with an MSE of 0.278. This means that the projected temperature values are significantly different from the actual temperature data. This might be due to the high value of mu, which causes the model to overfit.
- The model performs very poorly when mu=1, with an MSE of 66.29. This means that the projected temperature values are significantly different from the actual temperature data. This might be due to the unusually high value of mu, which causes the model to be severely overfitted.

As a result, it is critical to select an adequate value of mu in order for the linear regression model to work well. A value of mu that is too high or too low might cause the model to perform poorly. In general, a smaller mu value is desired since it leads to higher model generalization.

Long Short-Term Memory Model

Long Short-Term Memory (LSTM) is a recurrent neural network (RNN) architecture that can learn long-term relationships in sequential input. LSTMs are especially well-suited for time-series data processing and are frequently employed in disciplines such as natural language processing, audio recognition, and weather forecasting.

An LSTM model is trained on historical weather data before it can be used to forecast humidity in weather prediction. Here's a high-level summary of the procedure:

The data is read from a CSV file and a random selection of it is used for training and validation. The Keras library is used to create the LSTM model. The model has one input layer with the shape (seq_length, nb_features), where seq_length is the length of each sequence (i.e., the number of previous time steps utilized to generate a prediction) and nb_features is the

number of features in the input data (i.e., the number of input variables). Because there is just one characteristic (humidity) in this scenario, nb_features = 1. The input layer is followed by two LSTM layers of 7 and 5 units, with a dropout layer in between to minimize overfitting. One unit in the output layer has a linear activation function.

```
model_path = 'LSTM_model1.h5'
# build the network
nb_features = len(feat_cols) #number of features included in the training data
nb_out = 1 #expected output length
model = Sequential()

model.add(LSTM(input_shape=(seq_length, nb_features),units=7,return_sequences=True))
model.add(Dropout(0.001))
model.add(LSTM(units=5,return_sequences=False))
model.add(Dropout(0.001))
model.add(Dense(units=nb_out))
model.add(Activation("linear"))
optimizer = keras.optimizers.Adam(learning_rate = 0.9)
model.compile(loss='mean_squared_error', optimizer=optimizer,metrics=['mse'])
print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 20, 7)             252

 dropout (Dropout)           (None, 20, 7)             0

 lstm_1 (LSTM)               (None, 5)                 260

 dropout_1 (Dropout)         (None, 5)                 0

 dense (Dense)               (None, 1)                 6

 activation (Activation)     (None, 1)                 0

=================================================================
Total params: 518
Trainable params: 518
Non-trainable params: 0
_________________________________________________________________
None
```

```
history = model.fit(seq_arrays, seq_labs, epochs=6000, batch_size=15,validation_split=0.9, verbose=0,
callbacks = [keras.callbacks.EarlyStopping(monitor='val_loss',min_delta=0, patience=50, verbose=0, mode='min'),
keras.callbacks.ModelCheckpoint(model_path,monitor='val_loss', save_best_only=True,mode='min', verbose=0)])
```

Fig. 6. Model training using LSTM technique.

The model is trained on training data using the mean squared error loss function and the Adam optimizer with a learning rate of 0.9. The mean squared error measure is used to monitor the training process. If the validation loss does not improve after 50 epochs, the training is terminated, and the best model is stored.

The model is assessed on the validation data using the mean squared error metric. Finally, the model is used to create predictions on the test data, and the mean squared error metric is used to assess the predictions. The training and validation loss and mean squared error metrics are presented over the training epochs to visualize the training process.

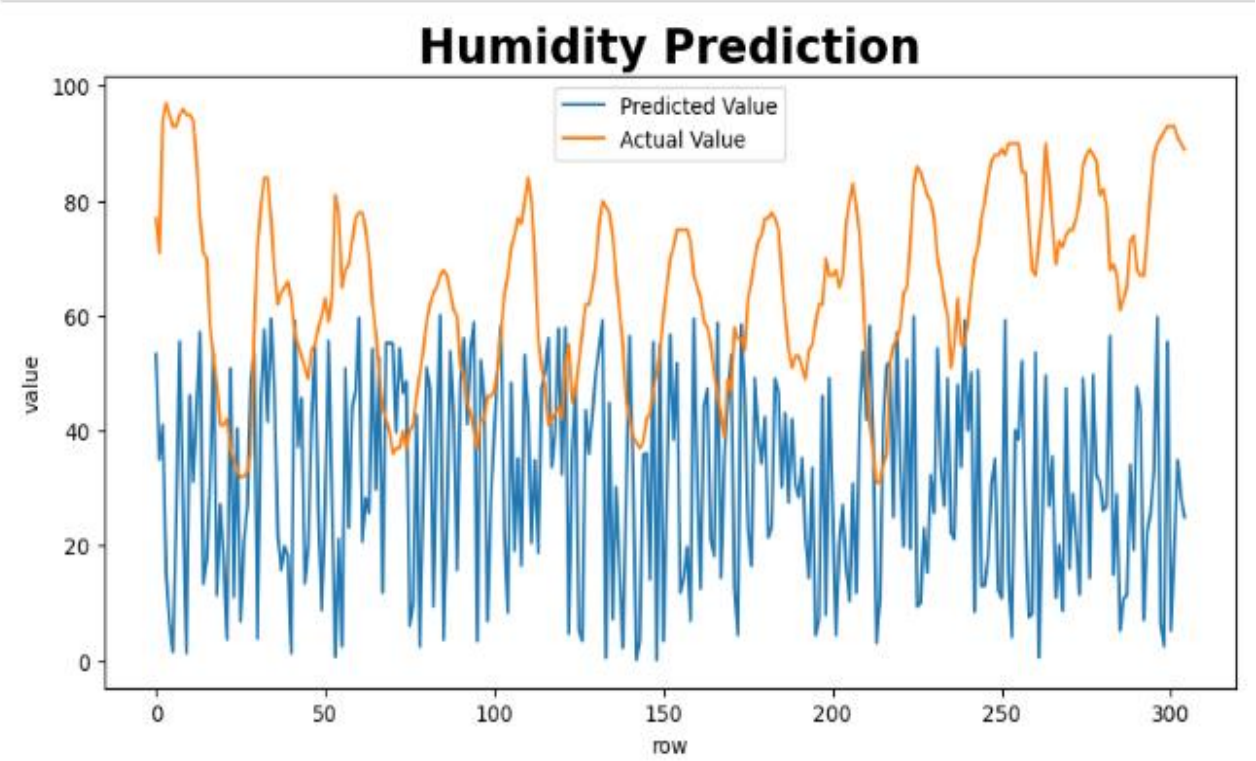

Fig. 7. Humidity Prediction using LSTM techniques.

Tableau Visualtization

Tableau is a sophisticated data visualization program that enables users to connect, view, and share data in a simple and dynamic manner. Tableau allows users to quickly build interactive dashboards, charts, graphs, and maps to explore and analyze data from a variety of sources.

Tableau connects to data sources like as spreadsheets, databases, and cloud services, and users may drag and drop data fields into a visualization canvas. Users may then select from a number of visualization formats to generate interactive charts, graphs, and maps that can be modified with different colors, labels, and formatting choices.

Tableau users may also build computed fields, filters, and parameters to analyze and segment data in a variety of ways. Tableau also has strong data blending capabilities, allowing users to mix and analyze data from numerous sources.

Temperature Prediction:

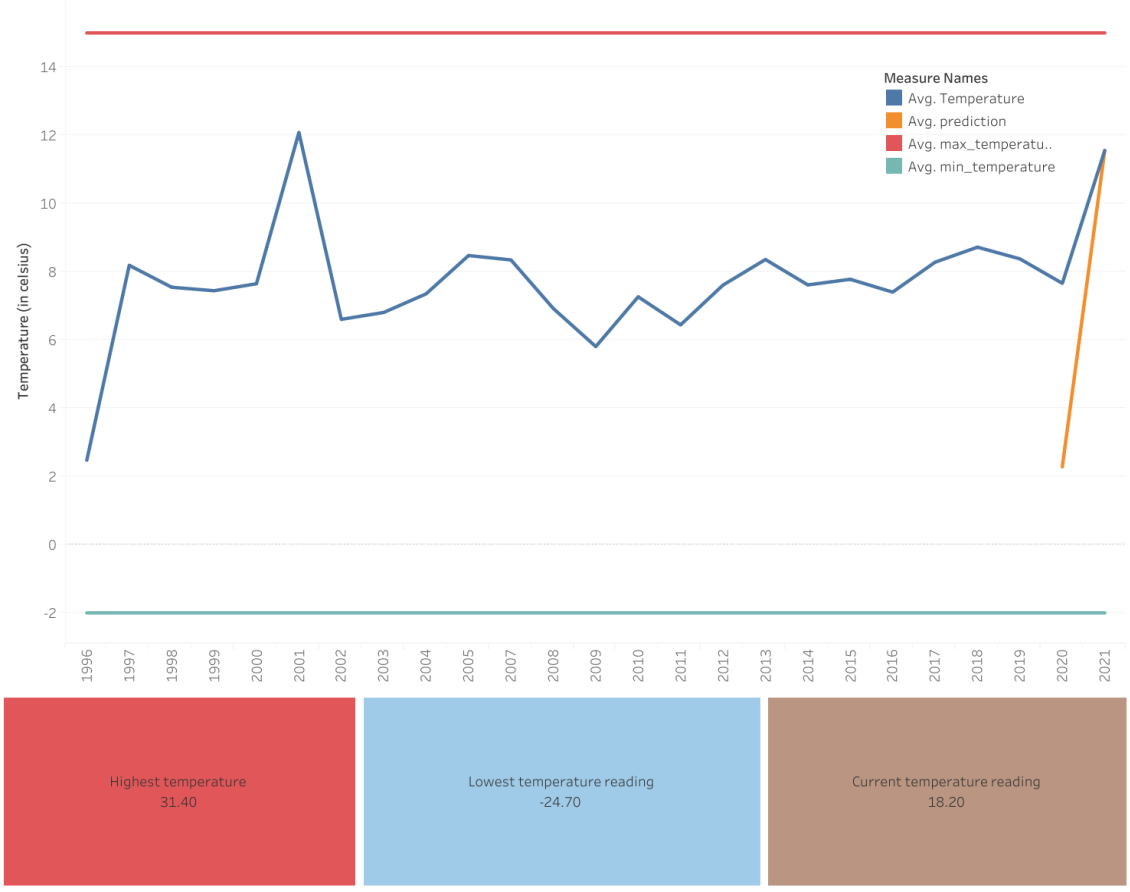

Fig. 7. Temperature Prediction visualization using Tableau dashboard.

When predicting temperature with linear regression, the data after linear regression performance is taken, and then a calculated field named prediction prediction temperature is created when it is null, and then two more fields named max temp and min temp are created giving the ranges to them, and finally all of these fields are added to the measure values and measure values are added to rows of the sheet. Later, colors are applied to each field in the display, and a parameter is set and filtered with it. Then three more sheets are produced, one with the lowest temperature reading, another with the highest temperature reading, and a third with the average temperature; the fonts, styles, and colors are changed. The sheets are then uploaded to the dashboard and tweaked to appear appropriate for visualization.

Humidity Prediction:

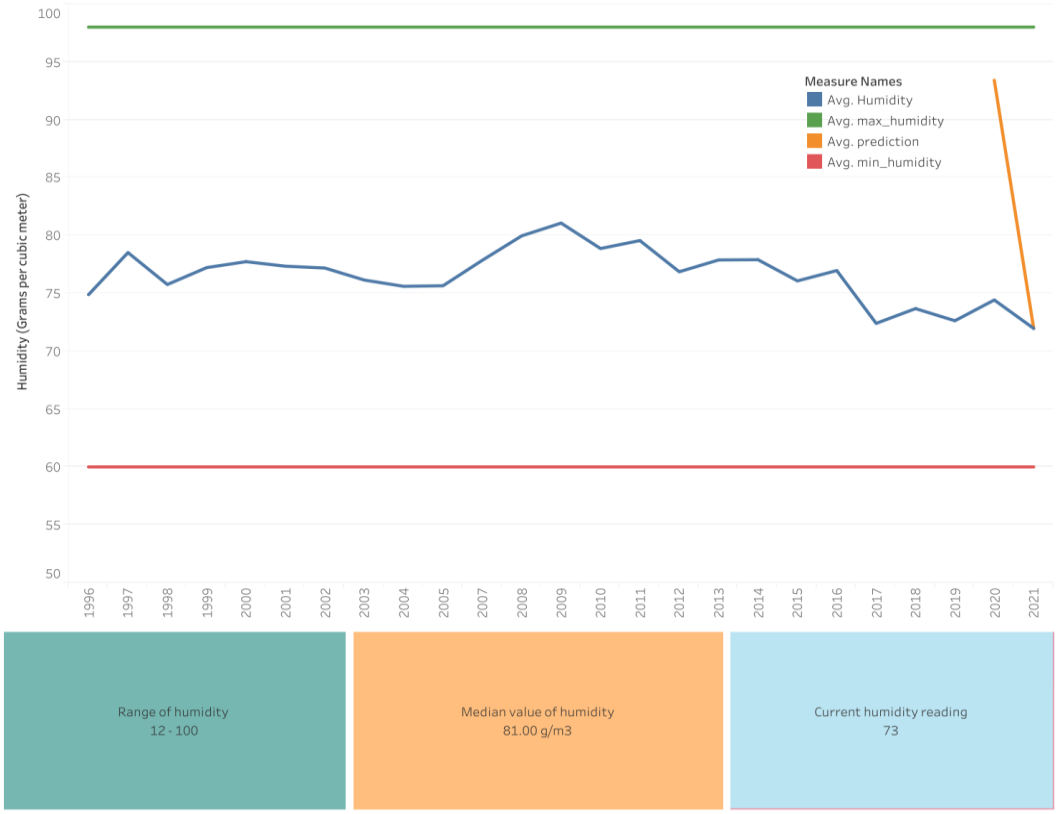

Fig. 8. Humidity Prediction visualization using Tableau dashboard.

When predicting humidity using an LSTM model, the data is first collected after the model has run, and when it is null, a calculated field called prediction. prediction humidity is created. Next, two more fields named max humidity and min humidity are created, giving them ranges. Finally, all of these fields are added to the measure values, and measure values are added to rows of the sheet. Each field in the display is afterwards given a color, and along with it, a parameter is set and a filter is applied. Then three additional pages are created, each with a different set of typefaces, styles, and colors. These sheets provide the median humidity reading, the range of humidity reading, and the humidity count when it exceeds 98g/m3. After that, the sheets are uploaded to the dashboard and adjusted to look good for visualization.

Conclusion

As a result, we were able to estimate temperature and humidity for weather prediction using both linear regression and LSTM models. The models have performed well in terms of accuracy, and Tableau has been used to display the data to give the anticipated values clear and illuminating visuals.

We can now base our judgments on reliable weather forecasts thanks to a mix of data cleaning, preprocessing, modeling, and visualization approaches. Several stakeholders, like weather forecasters, farmers, and transportation corporations, who depend on accurate weather predictions to make educated decisions, will be significantly impacted by the models' accuracy. Overall, these methods have helped to shed light on weather forecasting and have the potential to enhance numerous businesses' decision-making procedures.

References

Jaiswal, O. N. (2020). Weather Forecasting using Linear Regression In Machine Learning. Easychair.org. https://easychair.org/publications/preprint/Q2DV

Srivastava, S., & Lessmann, S. (2018). A comparative study of LSTM neural networks in forecasting day-ahead global horizontal irradiance with satellite data. Solar Energy, 162, 232–247. https://doi.org/10.1016/j.solener.2018.01.005

Sharaff, Aakanksha & Roy, Samuel. (2018). Comparative Analysis of Temperature Prediction Using Regression Methods and Back Propagation Neural Network. 739-742. 10.1109/ICOEI.2018.8553803.

N. Sri Lakshmi, P. Ajimunnisa, V. Lakshmi Prasanna, T. YugaSravani, & M. RaviTeja. (2021, June 10). Prediction of Weather Forecasting by Using Machine Learning. Papers.ssrn.com. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3902660