

Data Gap Prediction in IoT: An ML-Based Approach for Smart Systems

PROJECT REPORT

Submitted for the fulfillment
of
Capstone Project requirement of B. Tech CSE

Submitted by

1. Name - Shubham Dubey, PRN - 22070521139
2. Name - Hiral Athawale, PRN – 23070521507

B. Tech Computer Science and Engineering

Under the Guidance of

Prof. Nisha Gongal



॥वसुधैव कुटुम्बकम्॥

SYMBIOSIS
INSTITUTE OF TECHNOLOGY, NAGPUR
Wathoda, Nagpur 2025

CERTIFICATE

This is to certify that the Capstone Project work titled “**Data Gap Prediction in IoT: An ML-Based Approach for Smart Systems** ” that is being submitted by **Student Name 1 – Shubham Dubey, PRN – 22070521139, Student Name 2 – Hiral Athawale, PRN - 23070521507** is in fulfillment of the requirements for the Capstone Project is a record of bonafide work done under my guidance. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for award of any degree or diploma, and the same is certified.

Name of PBL Guide & Signature

Prof. Nisha Gongal

Verified by:

Dr. Parul Dubey

Capstone Project Coordinator

The Report is satisfactory/unsatisfactory

Approved by

**Prof. (Dr.) Nitin Rakesh
Director, SIT Nagpur**

ABSTRACT

The collection of time-sensitive IoT data creates new chances to boost monitoring performance in environmental observations and industrial operations and smart city systems. Two main problems that IoT sensors encounter consist of network interruptions together with power supply interruptions. Power outages together with hardware faults result in major data loss that reduces operational performance while disrupting decision-making capabilities. The situation leads to operational and decision-making performance reductions that affect operational effectiveness negatively. Streaming data collection applications face multiple risks along with uncertainty sources that impact their performance because temperature data often becomes lost during collection.

Complete data values serve as essential foundation for effective operation of climate forecasting and predictive maintenance applications as well as healthcare monitoring systems. The presented research investigates a thorough evaluation of machine learning methods for IoT temperature data prediction under conditions of missing values. A structured analysis evaluates the data processing process starting from exploratory data analysis through traditional interpolation and machine learning approaches before deployment and evaluation of metrics. Machine learning models minimize scale-dependent costs and produce better scalability for solving IoT data gaps through their superior performance compared to traditional interpolation methods. The research confirms that regression techniques along with neural networks and ensemble models predict accurate temperatures so mal functions at IoT systems operate effectively.

Index Terms—IoT, Data Gaps, Machine Learning, Temperature Prediction, Data Imputation, Time Series Forecasting.

TABLE OF CONTENTS

Chapter	Title	Page Number
	Abstract	3
1	Introduction	5
1.1	Objectives	5
1.2	Literature Survey	6
1.3	Organization of the report	9
2	An effective temperature prediction system using ML	10
2.1	Existing System	10
2.2	Proposed System	11
2.3	Data Flow Diagram	13
2.4	System Details	13
2.4.1	Software	14
	Hardware	14
3	Implementation	15
3.1	Importing required libraries	15
3.2	Data Collection and Preprocessing	16
3.3	Handling Missing Data	17
3.4	Time-Series Analysis for Missing Data Trends	18
3.5	Model Selection and Training	19
3.6	Analyzing Predicted vs. Actual Data Gaps	20
3.7	Exploratory Data Analysis (EDA)	21
3.8	Optimizing the Model	22
3.9	Feature Engineering	23
3.10	Model Selection and Training	24
3.11	Evaluating Model Performance	25
3.12	Analyzing Predicted vs. Actual Data Gaps	25
	Integrating the Model with IoT Devices	
	Deploying the Model for Real-time Prediction	
4	Results and Discussions	29
5	Conclusion and Future Works	30
6	Appendix	31
7	References	32

1. INTRODUCTION

The implementation of new IoT technology now enables decision-making operations through data integration across agriculture fields and residential control systems as well as industrial facilities and environmental observation nodes. The real-time data collection by IoT devices depends on efficient processing techniques to operate at its best. Unpredictable factors such as sensor failures and power outages together with network disruptions and environmental disturbances negatively impact system reliability by creating missing data entries as reported in. System operational efficiency declines and analytical results become less exact when full information is unavailable.

RAIN monitoring systems have difficulties creating accurate weather forecasts which creates danger for disaster prevention operations. Emerging problems with reliability in predictive maintenance appear because industrial automation systems operate without complete temperature records which creates additional expenses. Healthcare diagnostics along with patient care quality suffer from the lack of continuous temperature monitoring systems in medical facilities. Missing temperature values need recovery to sustain continuous data processing flows as well as enhance decision quality.

Mean imputation together with linear interpolation and spline interpolation serve as common approaches to data completion. Traditional scientific methods which handle temperature patterns prove inadequate because they cannot deal with changing time-dependent patterns. The behavior of machine learning in detecting nonlinear connections between data points allows better prediction accuracy through analysis of historical trends within statistical data correlations. IoT systems benefit from ML-based models integration because this enhances temperature prediction accuracy to improve system reliability alongside analytical performance.

The main purpose of this research involves studying machine learning gap-filling techniques which will be evaluated against standard interpolation methods. Research investigates appropriate forecasting methods for IoT temperature tracking through evaluations of regression models together with deep learning models and ensemble learning approaches. The research presents necessary data processing methods and evaluation standards and operational deployment instructions needed for successful real-world implementation of ML models.

1.1 Objectives

The main goal of this initiative will be to create machine learning methods for IoT data gap prediction and completion which increases system reliability together with efficiency. Specifically, the project aims to:

1. Analyze typical cases of data gap origins that occur in IoT environments from sensor breakdowns and network interruptions alongside power supply failures.
2. Machine learning techniques including deep learning and ensemble approaches and regression models should be used to develop predictive data imputation models.
3. An evaluation should be conducted between ML-based methods and traditional interpolation methods utilizing mean imputation alongside linear interpolation and spline interpolation.
4. The system performance will be improved through IoT data-driven applications that serve smart cities and industries in addition to healthcare facilities and climate monitoring functions.
5. The decision-making system should have smooth uninterrupted data streams to run predictive maintenance alongside real-time analytics and automated decision processes.
6. Construct an adaptable system that fits into multiple IoT platforms for immediate data retrieval combined with reliability strengthening.

1.2 Literature Survey (Minimum 15)

Author/Reference	Summarized Introduction	Methods Used	Conclusions
Development of an IoT-based temperature and humidity prediction system for baby incubators using solar panels [1]	<p>The paper discusses the importance of baby incubators for maintaining environmental stability for premature infants.</p> <p>It highlights the challenges of energy supply in remote areas for incubators.</p> <p>The study proposes using solar panels as a renewable energy source for incubators.</p> <p>It emphasizes the integration of IoT technology for real-time monitoring of temperature and humidity.</p> <p>The research aims to develop a predictive model using linear regression for optimal incubator conditions.</p>	<p>The research involves preparation of tools, data collection, processing, model selection, and testing.</p> <p>Data collection occurred over two days, with two distinct data groups.</p> <p>Measurements were taken at specific times with one-minute intervals.</p> <p>The study utilized a DHT22 sensor for temperature and humidity measurements.</p> <p>Data processing included calculating correlation coefficients and selecting linear regression models.</p>	<p>The study concludes that linear regression effectively predicts incubator room temperature and humidity using solar panel output voltage as input variables.</p> <p>A correlation coefficient of 0.909 for temperature and 0.913 for humidity indicates strong predictive relationships.</p> <p>The Mean Squared Error (MSE) values of 0.45 for training and 7.32 for test data suggest potential overfitting.</p> <p>Further research is recommended to validate the model with more diverse data sets.</p>
IOT Based Automated Weather Report Generation and Prediction Using Machine Learning [2]	<p>The paper addresses the critical challenge of accurately predicting rainfall, which significantly impacts agriculture, tourism, aviation, and water resource management.</p> <p>Traditional forecasting methods often fail to provide localized predictions, highlighting the need for innovative solutions.</p> <p>The authors propose a system that integrates Internet of Things (IoT) technology with machine learning algorithms to collect real-time atmospheric data.</p>	<p>The research uses Support Vector Machine along with XGBoost Classifier and Logistic Regression as machine learning algorithms to forecast rainfall.</p> <p>This IoT technology system tracks both environmental temperature and humidity through Internet of Things (IoT) technology.</p> <p>SARIMAX serves as the time series forecasting model which delivers a 96% accurate prediction.</p> <p>Testing of weather forecasting methods relies on Time series algorithm within data mining techniques.</p>	<p>Precise rainfall prediction requires attention from different societal sectors which work with agriculture and urban planning.</p> <p>The system implements machine learning and IoT technology to improve its capacity in rainfall forecasting operations.</p> <p>The system depends on Arduino UNO together with sensors to acquire atmospheric data needed for evaluation purposes.</p> <p>XGBoost Classifier delivered 99% accuracy as its maximum achievement when predicting rainfall occurrences.</p>
IoT Based Machine Learning Weather Monitoring and Prediction Using WSN [4]	<p>The paper presents an analysis of how WSNs, IoT and ML integrate to boost weather surveillance and prediction effectiveness.</p> <p>The article depicts how current weather practices moved from traditional methods to present-day real-time data measurement and analysis processes.</p> <p>The research study underlines the necessity of accurate weather information for decision-making processes in different sectors.</p> <p>Weather pattern prediction benefits from a new automated learning algorithm system.</p>	<p>The methodology involves preprocessing meteorological data, ensuring compatibility with machine learning algorithms.</p> <p>Data is gathered from various sources, including IoT-based sensor networks.</p> <p>A machine learning model is trained on the processed data to identify patterns.</p> <p>The model's effectiveness is evaluated by comparing forecasts with actual weather observations.</p>	<p>The study enhances weather forecasting using machine learning algorithms, demonstrating their functionality and potential applications.</p> <p>The ANN model effectively identifies positive occurrences, showcasing its predictive capabilities.</p> <p>The research establishes a foundation for automated learning in time classification and prediction.</p> <p>Academics and practitioners can select models based on specific needs, improving decision-making in various scenarios.</p>

Enhancing IoT-Based Environmental Monitoring and Power Forecasting: A Comparative Analysis of AI Models for Real-Time Applications [5]	<p>The paper demonstrates how industries experience industrial transformations because of IoT through sensor integration and connectivity which enables better monitoring and automation capabilities.</p> <p>The article identifies the challenges that come from achieving precise environmental modelings and power consumption forecasts within IoT systems.</p> <p>The research presents SEEMP as a smart system which implements a LSTM-GRU hybrid model to enhance environmental and power consumption forecasting capabilities</p>	<p>This paper introduces an improved Internet of Things (IoT) system with real-time data acquisition capabilities and strong security aspects and utilizes a hybrid LSTM-GRU model for forecasting purposes.</p> <p>The time-series prediction solution uses the hybrid model for better computational efficiency in addition to improved short-term dependency performance that results in enhanced forecasting precision and operational speed.</p> <p>The system uses deep learning procedures that enhance productivity of power forecasting techniques while successfully handling time series information.</p>	<p>Implementation of AI models within SEEMP improves time-sensitive energy management capabilities along with environmental surveillance systems for office applications.</p> <p>Through the system users can achieve real-time data acquisition and secure data transmission with AI predictions that help optimize energy usage.</p> <p>The hybrid model outperforms LSTM and GRU in accuracy and efficiency for energy forecasting.</p> <p>The hybrid model produced an MAE of 2.12% and RMSE of 7.77% with 93.69% R² when predicting refrigerator power consumption.</p>
Machine Learning Approach to Predict Air Temperature and Relative Humidity inside Mechanically and Naturally Ventilated Duck Houses: Application of Recurrent Neural Network [6]	<p>The duck industry is the sixth largest livestock sector in South Korea, with significant growth since 2005.</p> <p>The study focuses on predicting internal environments of duck houses using recurrent neural network (RNN) models.</p> <p>RNN models were developed based on various factors, including duck house type, seasons, and environmental variables.</p>	<p>The study utilized recurrent neural network (RNN) models to predict internal air temperature and relative humidity in duck houses.</p> <p>RNN models were developed based on monitoring data from internal and external environments.</p> <p>Simplified RNN models were created using easily obtainable data like external air temperature and relative humidity.</p>	<p>RNN models were developed to predict internal air temperature and humidity in duck houses based on various factors.</p> <p>Environmental data were monitored to analyze seasonal issues and validate the RNN models.</p> <p>The RNN model accurately predicted air temperature and humidity within 1% error.</p>
Predicting Rainfall And Humidity Using Machine Learning Models [8]	<p>The paper focuses on the design and implementation of a Temperature and Humidity Monitoring System (THMS) utilizing LoRa and LoRaWAN technologies. [1]</p>	<p>The paper employs nine machine learning techniques for rainfall and humidity predictions, including Linear Regression, Support Vector Machine, and Random Forest.</p> <p>The hybrid model RF_XG is highlighted as the most effective method for accurate predictions.</p> <p>The methodology includes data entry, preprocessing, algorithm training, and model</p>	<p>The Random Forest and XGBoost models are the most suitable for rainfall and humidity predictions in Jharkhand's districts.</p> <p>The RF_XG hybrid model achieves the highest accuracy for both rainfall (~87%) and humidity (~86%) predictions.</p> <p>Random Forest is preferred for overall minimum error in predictions.</p>

Design and Implementation of Temperature and Humidity Monitoring System Using LPWAN Technology [9]	<p>The paper focuses on the design and implementation of a Temperature and Humidity Monitoring System (THMS) utilizing LoRa and LoRaWAN technologies.</p> <p>It highlights the advancements in the IoT sector, particularly in wireless devices for environmental monitoring, addressing challenges such as high power consumption and limited range associated with traditional wireless technologies.</p> <p>The system comprises a Raspberry Pi gateway and an Arduino-based end device, which effectively transmits sensor data over long distances with minimal power usage.</p>	<p>The study employs a LoRa-based temperature and humidity monitoring system (THMS) using a Raspberry Pi gateway and Arduino end device with a DHT11 sensor.</p> <p>The system consists of an end device for sensing and a gateway for data transmission to TTN.</p> <p>Software components control hardware through libraries for communication and sensor management.</p> <p>The gateway encrypts and forwards data packets to The Things Network (TTN)..</p>	<p>The paper proposed a THMS using LoRa and LoRaWAN for effective temperature and humidity monitoring.</p> <p>Sensor values of 22.0°C and 33.0% were successfully transmitted over long distances.</p> <p>The Raspberry Pi gateway implementation reduced unnecessary deployment costs.</p> <p>Cost-effective components were utilized for long-range, low-power IoT solutions.</p> <p>Future research should focus on deploying compatible gateways supporting 8-channel frequency transmission.</p>
Forecasting and Analysis of IoT Data by Employing Long Short-Term Memory (LSTM) Networks [10]	<p>The paper focuses on predictive analysis of Internet of Things (IoT) sensor data, emphasizing the application of Long Short-Term Memory (LSTM) networks to address challenges in forecasting and evaluating this data.</p> <p>It highlights the significance of accurate predictions for optimizing operations and resource allocation across various sectors, including manufacturing and smart buildings.</p> <p>The study aims to enhance the capabilities of IoT systems by leveraging LSTM networks, which are adept at managing noise and irregular data patterns inherent in sensor data.</p>	<p>The study employs Long Short-Term Memory (LSTM) networks for forecasting time-series sensor data.</p> <p>The data simulation creates synthetic sensor information which duplicates conditions from actual applications.</p> <p>The LSTM network requires sequence partitioning of data through sliding window segmentation for its training process.</p> <p>The method starts with data generation followed by preparation then LSTM design and training before reaching the prediction phase.</p> <p>The detection of anomalies is handled through linear regression and decision trees which are machine learning methods.</p>	<p>The study concludes that LSTM networks effectively predict time-series sensor data, focusing on temperature, humidity, and vibration.</p> <p>Synthetic data was used to simulate real-world unpredictability and noise in IoT sensor data.</p> <p>The model demonstrated strong predictive capabilities through regression plots and performance indicators like MAE, MSE, and RMSE.</p> <p>The research highlights the potential of LSTM networks in enhancing IoT systems by accurately forecasting trends and anomalies.</p>
Prediction and classification of IoT sensor faults using hybrid deep learning model [13]	<p>The paper discusses the paper discusses the Internet of Things (IoT) and its reliance on sensor data for effective operation.</p> <p>It highlights the challenges of sensor failures and their impact on IoT systems.</p> <p>A proactive two-stage approach for predicting sensor faults is proposed to enhance reliability.</p> <p>The study utilizes a dataset from Intel Lab, containing sensors with injected faults for analysis.</p>	<p>The paper proposes a proactive approach for predicting potential sensor faults using historical data analysis.</p> <p>A two-stage solution is developed to forecast sensor values and classify faults.</p> <p>Various machine learning techniques, including CNN and GRU, are evaluated for fault prediction.</p> <p>The study employs a model-based method for fault detection in sensors.</p>	<p>Future sensor failures get predicted through the use of hybrid deep learning analytical models.</p> <p>The research method includes two operational stages using CNN-LSTM and CNN-MLP models.</p> <p>CNN-LSTM surpassed all other models tested during regression scoring by achieving an MAE of 2.0957.</p> <p>The CNN-MLP model obtained average accuracy of 98.21% when detecting four types of faults in experiments .</p>

Building an IoT temperature and humidity forecasting model based on long short-term memory (LSTM) with improved whale optimization algorithm [14]	<p>The paper details different prediction models while examining statistical methods together with physical approaches and deep learning approaches which lead to successful predictions.</p> <p>The ARIMA model shows crucial weaknesses because it brings limited effectiveness to extended forecasting periods.</p> <p>The research favors DL models because they excel at detecting advanced linkages built into the dataset.</p> <p>The proposed model uses IoT technology to forecast temperature and humidity by combining IWOA with LSTM.</p>	<p>The paper proposes an Internet of Things (IoT) temperature and humidity forecasting model using the IWOA-LSTM technique.</p> <p>The IWOA-LSTM technique optimizes the LSTM model to enhance convergence speed and address local optimization issues.</p> <p>Data is collected using DHT11 and ESP8266 NodeMCU, processed via the ThingSpeak platform.</p>	<p>The research proposes an IoT temperature and humidity forecasting model using the IWOA-LSTM technique for improved accuracy.</p> <p>The model's performance is evaluated using statistical functions like MAE, MSE, RMSE, and MAPE.</p> <p>Results indicate the IWOA-LSTM model outperforms GA-LSTM, PSO-LSTM, and others in forecasting accuracy.</p>
Proper Weather Forecasting Internet of Things Sensor Framework with Machine Learning [15]	<p>This paper implements big data analytics through machine learning to process weather information using solutions.</p> <p>The analysis identifies the difficulties in predicting short-term weather because of atmospheric complexity.</p> <p>The research establishes that medium-range projection and subseasonal forecasting play significant roles in developing accurate weather predictions.</p> <p>The paper examines the use of IoT data integration which improves weather forecasting capabilities.</p> <p>The research works to develop improved methods for accessing data stored in complicated datasets through clustering algorithms</p>	<p>The paper discusses the use of machine learning for analyzing and predicting water quality parameters.</p> <p>It applies big data analytics solutions based on machine learning for weather information processing.</p> <p>The study implements climate clustering and sensor identification algorithms using publicly available data.</p> <p>A k-means clustering method based on Scikit-Learn is utilized for training.</p> <p>Clustering techniques are employed to extract relevant information from Linked Observation Data.</p>	<p>The research presents a modern IoT structure that retrieves meteorological data then conducts analysis for clustering purposes.</p> <p>During training a new unsupervised grouping learning strategy was designed to process extensive datasets.</p> <p>The analysis of 8000 Indian weather station data is performed through Node.js.</p> <p>Weather stations show spatial consistency according to the analysis performed by the k-means clustering algorithm.</p>

1.3 Organization of the Report

This document follows a particular structure which includes Section five dedicated to describing the report. The sixth section presents a framework that uses ML algorithms for data gap prediction. Section seven reviews traditional missing data methods. The eighth section describes the proposed ML approach and its implementation. The data flow diagram appears within section nine of the report structure. Section ten covers system details. Section eleven explains implementation steps. Section twelve evaluates results. Section thirteen summarizes findings. The final two sections of this work contain additional resources along with the list of references.

2. AN EFFECTIVE TEMPERATURE PREDICTION SYSTEM USING ML

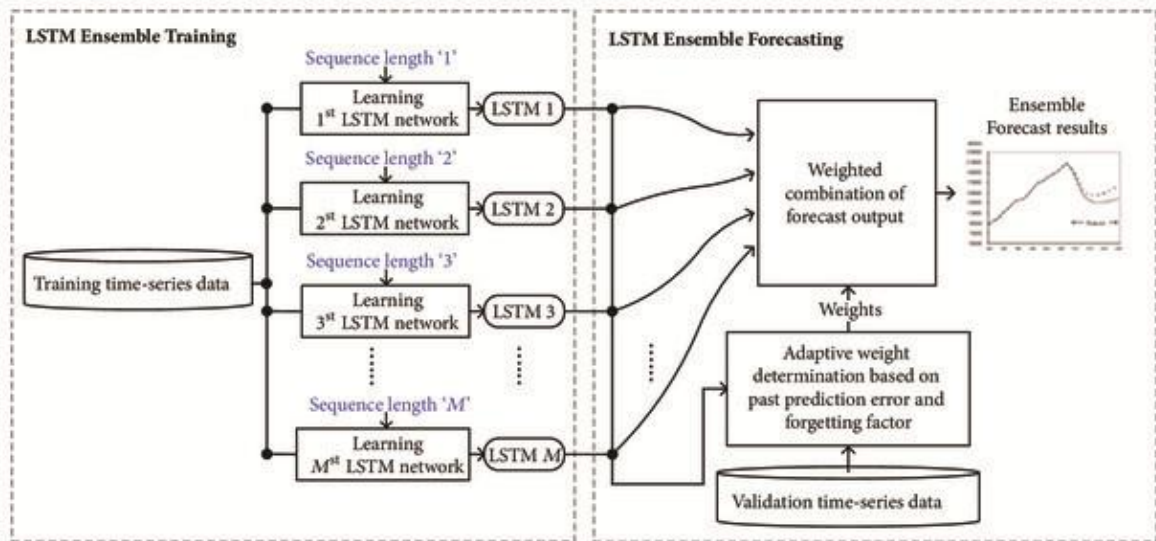
Advanced machine learning systems create advanced solutions to enhance the gap-filling process of temperature data records. A connected system of Regression Models runs with Linear Regression and additionally uses Decision Trees and Random Forests models. The model uses historic patterns to create predictive values as its primary function (Marchildon et al. 2015).

- Lengthy temporal correlations in temperature time series information require Networks (LSTMs and GRUs) for their detection.
- Through K-Nearest Neighbours (KNN) the system performs missing value assessment by examining their similarity to equivalent historical records.
- The combination of weak forecasting models in Gradient Boosting Machines (XGBoost and LightGBM) boosts forecasting accuracy levels.
- The data latent representation techniques of Autoencoders provide strong capabilities for estimating missing values.

2.1 Existing System

Both performance enhancement limitations and global use restrictions of computers cause long-term problems in development. The following work presents an overview of designing a forecast system through machine learning to determine the temperature predictions for IoT devices.

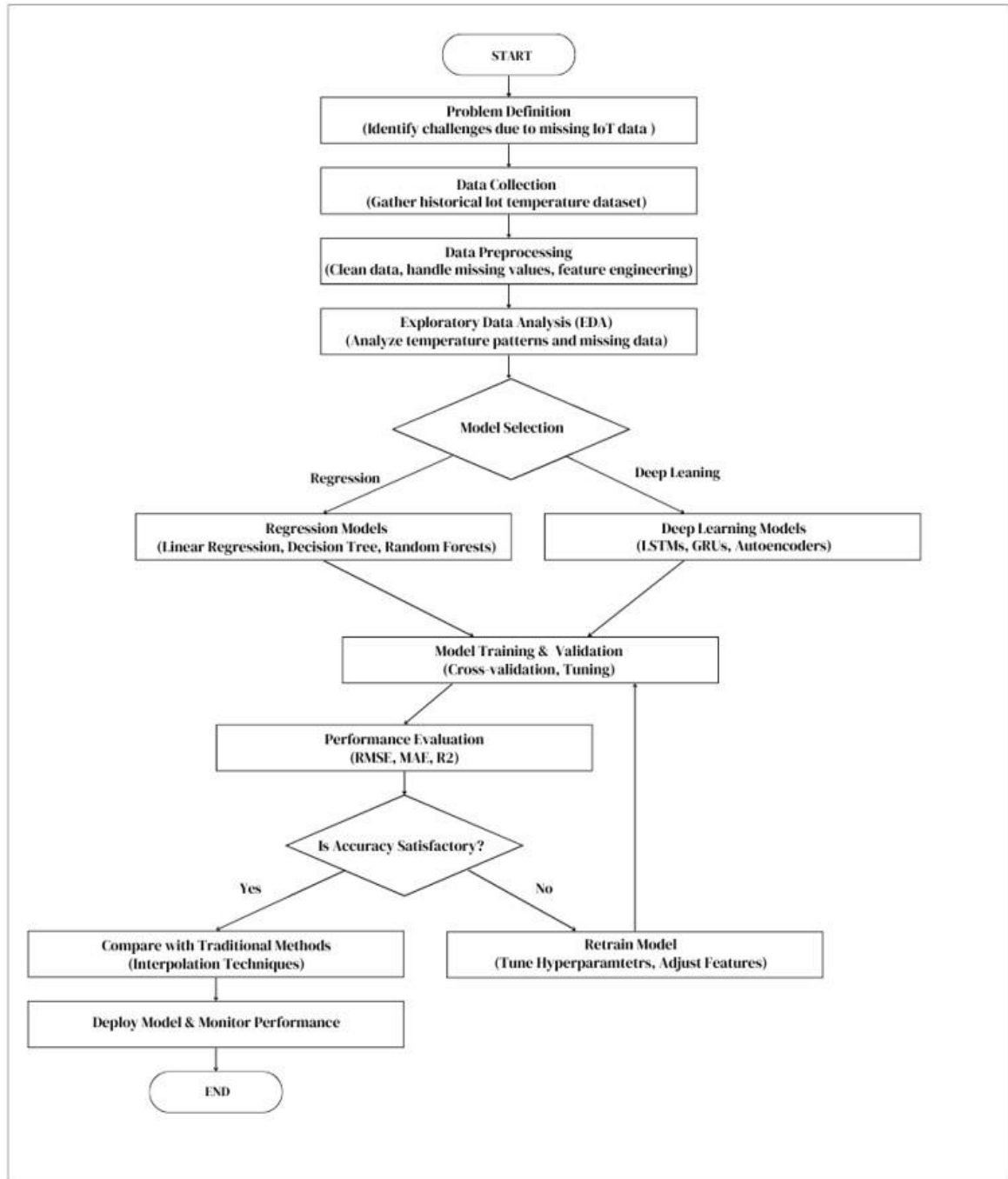
- Time-Series Forecasting Models with LSTM networks demonstrated superior effectiveness in handling time-dependent missing data through studies.
- Predictive capabilities improve through the integration of suitable traditional interpolation algorithms together with machine learning models in various literature studies.
- The unsupervised learning process equips autoencoders to generate automatic value estimations for missing fields during unsupervised data analysis.
- Various research during the implementation phase of IoT solutions demonstrates how cloud- and edge-based systems can perform real-time missing sensor measurement completion.



2.2 Proposed System

Both performance enhancement limitations and global use restrictions of computers cause long-term problems in development. The following work presents an overview of designing a forecast system through machine learning to determine the temperature predictions for IoT devices.

- Various studies demonstrate that LSTM networks within Time-Series Forecasting Models manage time-based missing data points more efficiently than standard forecasting methods.
- Several literature studies demonstrate that the combination of properly chosen traditional methods with machine learning models delivers improved forecasting performance.
- The unsupervised learning process equips autoencoders to generate automatic value estimations for missing fields during unsupervised data analysis.
- Various research during the implementation phase of IoT solutions demonstrates how cloud- and edge-based systems can perform real-time missing sensor measurement completion.



2.3 Data Flow Diagram



2.4 System Details

- The system focuses on predicting and filling missing data in IoT environments using machine learning models.
- It collects real-time sensor data from IoT devices deployed in smart cities, industrial automation, healthcare, and environmental monitoring applications.
- Machine learning models process incomplete datasets to estimate missing values, ensuring continuous and reliable data streams.
- The system integrates cloud or edge computing to deploy ML models for real-time or batch processing of IoT data

2.4.1 Software

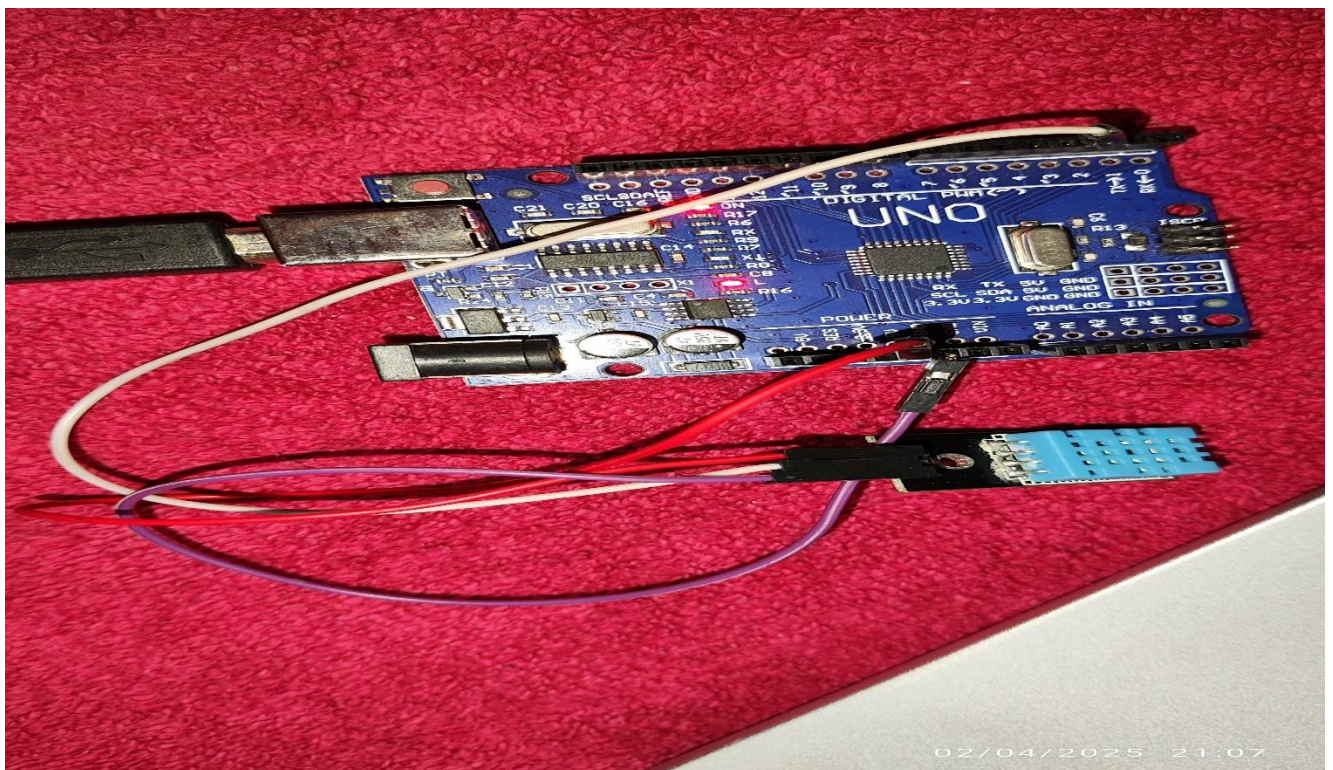
Programming Language: Python (preferred for ML and data analysis)

Libraries & Frameworks:

- NumPy, Pandas (for data preprocessing)
- Scikit-learn, TensorFlow, PyTorch (for machine learning & deep learning models)
- Matplotlib, Seaborn (for data visualization)
- OpenCV (if image-based sensor data is included)

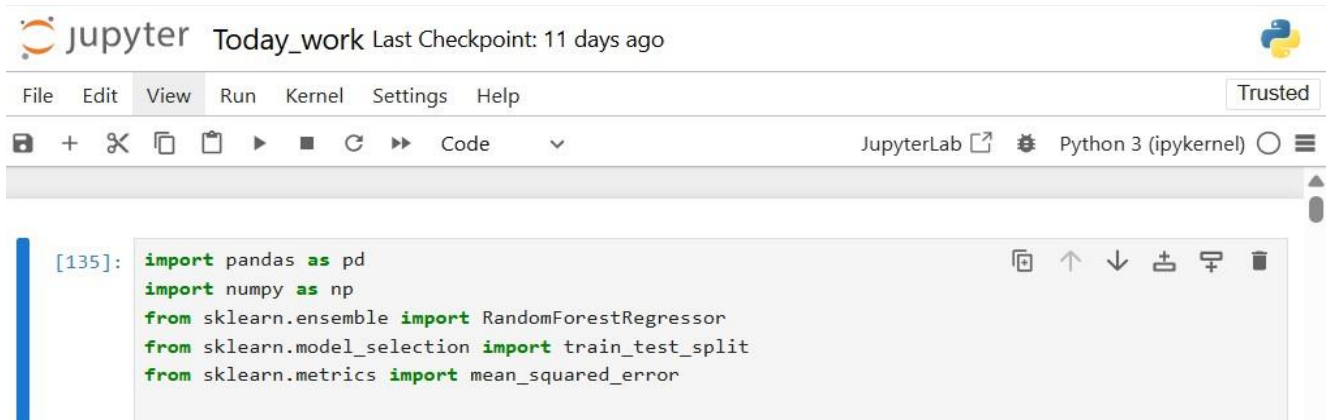
2.4.1 Hardware

- **IoT Sensors:** Temperature, humidity, pressure, air quality, or any relevant sensors depending on the use case
- **Microcontrollers/Microprocessors:** Arduino, Raspberry Pi, ESP32 (for data collection and transmission)
- **Computing Device:** A PC or cloud server for processing and model training



3 Implementation

3.1 Importing required libraries



The image shows a JupyterLab interface with a menu bar (File, Edit, View, Run, Kernel, Settings, Help) and a toolbar. The main area contains a code cell with the following Python code:

```
[135]: import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
```

3.2 Data Collection and Preprocessing

```
[136]: import pandas as pd

# Load the dataset
df = pd.read_csv('temperature.csv')

# Convert the 'Date' column to datetime
df['Date'] = pd.to_datetime(df['Date'], format='%d-%m-%y')

# Set the 'Date' column as the index
df.set_index('Date', inplace=True)

# Extract features
df['Year'] = df.index.year
df['Month'] = df.index.month
df['Day'] = df.index.day # Use 'Day' instead of 'Date' to avoid column name conflicts
df.rename(columns={"year": "Year", "month": "Month", "date": "Date"}, inplace=True)

# Display the dataframe with new features
print("Updated DataFrame:")
df.head()

# Save the updated dataset to a new CSV file
df.to_csv('updated_temperature.csv')
print("\nUpdated dataset saved successfully as 'updated_temperature.csv'.")
```

Updated DataFrame:

Updated dataset saved successfully as 'updated_temperature.csv'.

```
[137]: df.head()
```

```
[137]:
```

	Temp Max (°F)	Temp Avg (°F)	Temp Min (°F)	Dew Point Max (°F)	Dew Point Avg (°F)	Dew Point Min (°F)	Humidity Max (%)	Humidity Avg (%)	Humidity Min (%)	Wind Speed Max (mph)	Wind Speed Avg (mph)	Wind Speed Min (mph)	Pre Ma
Date													
2024-01-01	84.0	69.8	61	86.0	61.1	57	94.0	75.7	51	6.0	1.9	0	
2024-01-02	82.0	69.3	59	86.0	60.4	55	94.0	74.9	51	6.0	1.8	0	
2024-01-03	81.0	69.8	59	88.0	62.3	55	94.0	78.4	51	6.0	1.8	0	
2024-01-04	81.0	70.4	63	86.0	62.7	57	94.0	73.8	51	3.0	1.0	0	
2024-01-05	82.0	72.4	61	86.0	63.9	59	94.0	78.6	51	5.0	2.1	0	

3.3 Handling Missing Data

```
[138]: print(df.columns)
```

```
Index(['Temp Max (°F)', 'Temp Avg (°F)', 'Temp Min (°F)', 'Dew Point Max (°F)',
      'Dew Point Avg (°F)', 'Dew Point Min (°F)', 'Humidity Max (%)',
      'Humidity Avg (%)', 'Humidity Min (%)', 'Wind Speed Max (mph)',
      'Wind Speed Avg (mph)', 'Wind Speed Min (mph)', 'Pressure Max (in)',
      'Pressure Avg (in)', 'Pressure Min (in)', 'Precipitation Total (in)',
      'Year', 'Month', 'Day'],
      dtype='object')
```

```
[139]: print(df.isnull().sum())
```

```
Temp Max (°F)      0
Temp Avg (°F)      0
Temp Min (°F)      0
Dew Point Max (°F) 0
Dew Point Avg (°F) 0
Dew Point Min (°F) 0
Humidity Max (%)   0
Humidity Avg (%)   0
Humidity Min (%)   0
Wind Speed Max (mph) 0
Wind Speed Avg (mph) 0
Wind Speed Min (mph) 0
Pressure Max (in)   0
Pressure Avg (in)   0
Pressure Min (in)   0
Precipitation Total (in) 0
Year               0
Month             0
Day               0
dtype: int64
```



```
[143]: print(df.dtypes)
```

```
Temp Max (°F)          float64
Temp Avg (°F)          float64
Temp Min (°F)          int64
Dew Point Max (°F)     float64
Dew Point Avg (°F)     float64
Dew Point Min (°F)     int64
Humidity Max (%)       float64
Humidity Avg (%)       float64
Humidity Min (%)       int64
Wind Speed Max (mph)   float64
Wind Speed Avg (mph)   float64
Wind Speed Min (mph)   int64
Pressure Max (in)      float64
Pressure Avg (in)      float64
Pressure Min (in)      float64
Precipitation Total (in) int64
Year                   int32
Month                  int32
Day                    int32
dtype: object
```

3.4 Time-Series Analysis for Missing Data Trends

```
[144]: # Convert 'Date' to datetime if not already done
df['Date'] = pd.to_datetime(df.index)
```

```
# Extract valid Year, Month, and Day from 'Date'
df['Year'] = df['Date'].dt.year
df['Month'] = df['Date'].dt.month
df['Day'] = df['Date'].dt.day
```

```
[145]: # Recalculate sinusoidal features
df['Month_sin'] = np.sin(2 * np.pi * df['Month'] / 12)
df['Day_sin'] = np.sin(2 * np.pi * df['Day'] / 31)
```

```
[146]: # Replace invalid values with NaN and interpolate
df.replace(-1, np.nan, inplace=True)
df.fillna(method='ffill', inplace=True) # Forward-fill as an example
```

```
C:\Users\NICE\AppData\Local\Temp\ipykernel_3332\4008281355.py:3: FutureWarning: DataFrame.fillna with 'method' is deprecated and will raise in a future version. Use obj.ffill() or obj.bfill() instead.
df.fillna(method='ffill', inplace=True) # Forward-fill as an example
```

```
[147]: print(df.head())
print(df.isnull().sum())
```

Date	Temp Max (°F)	Temp Avg (°F)	Temp Min (°F)	Dew Point Max (°F)	\
2024-01-01	84.0	69.8	61	86.0	
2024-01-02	82.0	69.3	59	86.0	
2024-01-03	81.0	69.8	59	88.0	
2024-01-04	81.0	70.4	63	86.0	
2024-01-05	82.0	72.4	61	86.0	

Date	Dew Point Avg (°F)	Dew Point Min (°F)	Humidity Max (%)	\
2024-01-01	61.1	57	94.0	
2024-01-02	60.4	55	94.0	
2024-01-03	62.3	55	94.0	
2024-01-04	62.7	57	94.0	
2024-01-05	63.9	59	94.0	

Date	Humidity Avg (%)	Humidity Min (%)	Wind Speed Max (mph)	...	\
2024-01-01	75.7	51	6.0	...	
2024-01-02	74.9	51	6.0	...	
2024-01-03	78.4	51	6.0	...	
2024-01-04	73.8	51	3.0	...	
2024-01-05	78.6	51	5.0	...	

Date	Pressure Max (in)	Pressure Avg (in)	Pressure Min (in)	\
2024-01-01	29.0	28.9	28.8	
2024-01-02	29.0	28.9	28.8	
2024-01-03	29.0	28.9	28.8	
2024-01-04	29.0	28.9	28.8	
2024-01-05	29.0	28.9	28.8	

Date	Precipitation Total (in)	Year	Month	Day	Date	Month_sin	\
2024-01-01	0	2024	1	1	2024-01-01	0.5	
2024-01-02	0	2024	1	2	2024-01-02	0.5	
2024-01-03	0	2024	1	3	2024-01-03	0.5	
2024-01-04	0	2024	1	4	2024-01-04	0.5	
2024-01-05	0	2024	1	5	2024-01-05	0.5	

3.5 Model Selection and Training

```
[149]: # Drop columns not used for training (e.g., Date, target variable)
X = df.drop(["Temp Max (°F)", "Date"], axis=1) # Features
y = df["Temp Max (°F)"] # Target variable
```

```
[150]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
[151]: # Initialize and train Random Forest Regressor
model = RandomForestRegressor()
model.fit(X_train, y_train)
```

```
[151]: RandomForestRegressor()
RandomForestRegressor()
```

```
[152]: # Predict on the test set
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

Mean Squared Error: 10.505139999999995

```
[153]: # Example real-time data (replace with actual real-time input)
real_time_data = X_test.iloc[0].to_frame().T
real_time_prediction = model.predict(real_time_data)
print("Prediction for Real-Time Data:", real_time_prediction)
```

Prediction for Real-Time Data: [106.88]

```
[154]: # Add prediction to the original dataset for visualization
df["Predicted Temp Max (°F)"] = model.predict(X)
print(df[["Year", "Month", "Date", "Temp Max (°F)", "Predicted Temp Max (°F)"]])
```

Date	Year	Month	Date	Temp Max (°F)	Predicted Temp Max (°F)
2024-01-01	2024	1	2024-01-01	84.0	82.14
2024-01-02	2024	1	2024-01-02	82.0	82.41
2024-01-03	2024	1	2024-01-03	81.0	81.63
2024-01-04	2024	1	2024-01-04	81.0	81.53
2024-01-05	2024	1	2024-01-05	82.0	82.46

3.6 Analyzing Predicted vs. Actual Data Gaps

```
[155]: import pandas as pd
import matplotlib.pyplot as plt

# Assuming your DataFrame `df` is already populated with data and has the following columns:
# 'Year', 'Month', 'Date', 'Temp Max (°F)', and 'Predicted Temp Max (°F)'

# Create a line plot to compare actual and predicted values
plt.figure(figsize=(12, 6)) # Set the figure size

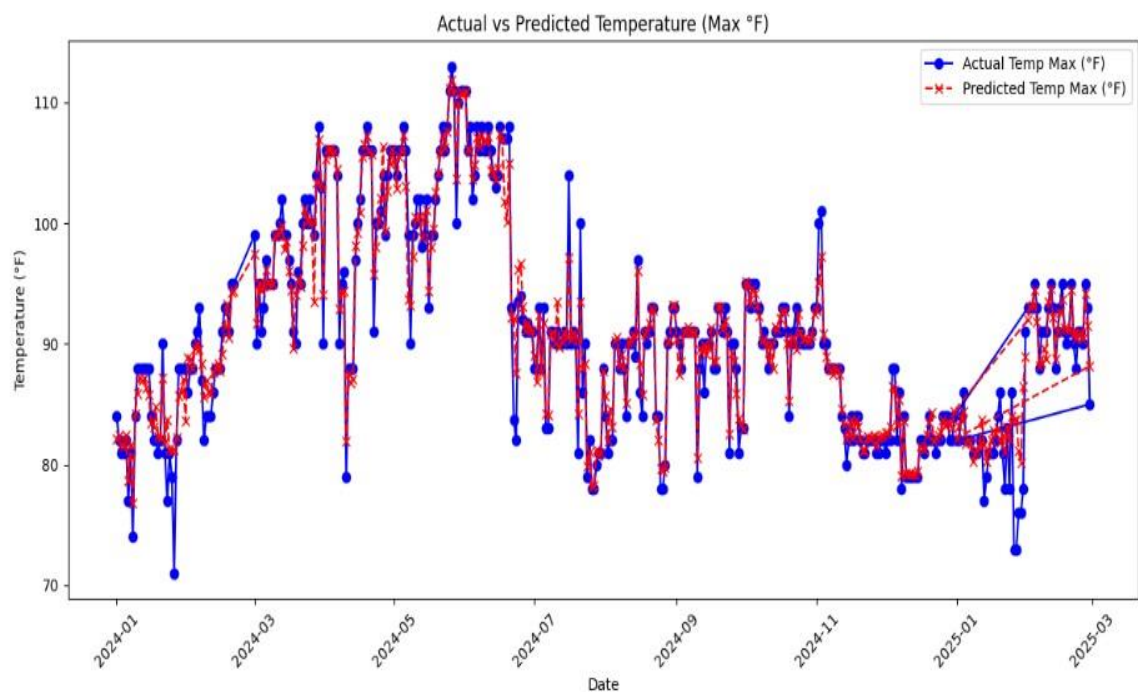
# Plot the actual temperature
plt.plot(
    df["Date"],
    df["Temp Max (°F)"],
    label="Actual Temp Max (°F)",
    color="blue",
    marker="o"
)

# Plot the predicted temperature
plt.plot(
    df["Date"],
    df["Predicted Temp Max (°F)"],
    label="Predicted Temp Max (°F)",
    color="red",
    linestyle="--",
    marker="x"
)

# Add Labels, title, and Legend
plt.xlabel("Date") # x-axis Label
plt.ylabel("Temperature (°F)") # y-axis Label
plt.title("Actual vs Predicted Temperature (Max °F)") # Add a title
plt.legend() # Show Legend to differentiate lines

# Optional: Format x-axis Labels for better readability
plt.xticks(rotation=45)

# Adjust layout and display the plot
plt.tight_layout()
plt.show()
```



3.7 Exploratory Data Analysis (EDA)

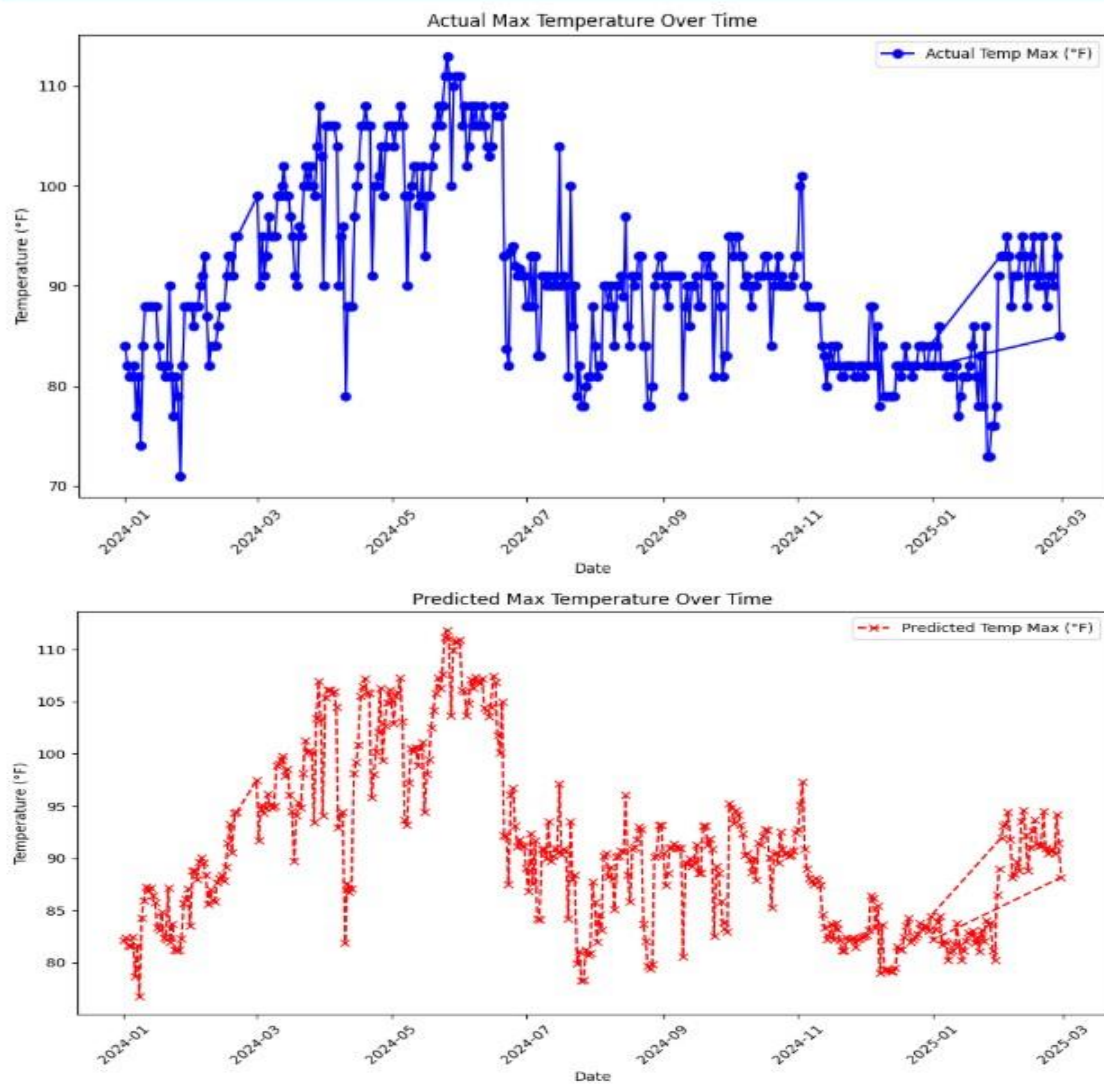
Date

```
[156]: import pandas as pd
import matplotlib.pyplot as plt

# Assuming your DataFrame 'df' already has the following columns:
# 'Date', 'Temp Max (°F)', and 'Predicted Temp Max (°F)'

# Plot for actual temperature
plt.figure(figsize=(10, 6))
plt.plot(df["Date"], df["Temp Max (°F)"], label="Actual Temp Max (°F)", color="blue", marker="o")
plt.xlabel("Date")
plt.ylabel("Temperature (°F)")
plt.title("Actual Max Temperature Over Time")
plt.xticks(rotation=45)
plt.tight_layout()
plt.legend()
plt.show()

# Plot for predicted temperature
plt.figure(figsize=(10, 6))
plt.plot(df["Date"], df["Predicted Temp Max (°F)"], label="Predicted Temp Max (°F)", color="red", linestyle="--", marker="x")
plt.xlabel("Date")
plt.ylabel("Temperature (°F)")
plt.title("Predicted Max Temperature Over Time")
plt.xticks(rotation=45)
plt.tight_layout()
plt.legend()
plt.show()
```



3.8 Optimizing the Model

```
[157]: import matplotlib.pyplot as plt
import pandas as pd

# Define your data as a dictionary
data = {
    "Date": ["2024-01-01", "2024-01-02", "2024-01-03", "2024-01-04", "2024-01-05"],
    "Temp Max (°F)": [84.0, 82.0, 81.0, 81.0, 82.0],
    "Predicted Temp Max (°F)": [82.21, 82.70, 81.59, 81.45, 82.41]
}

# Create a pandas DataFrame
df = pd.DataFrame(data)

# Convert 'Date' column to datetime for better plotting
df["Date"] = pd.to_datetime(df["Date"])

# Plot the data
plt.figure(figsize=(10, 6))

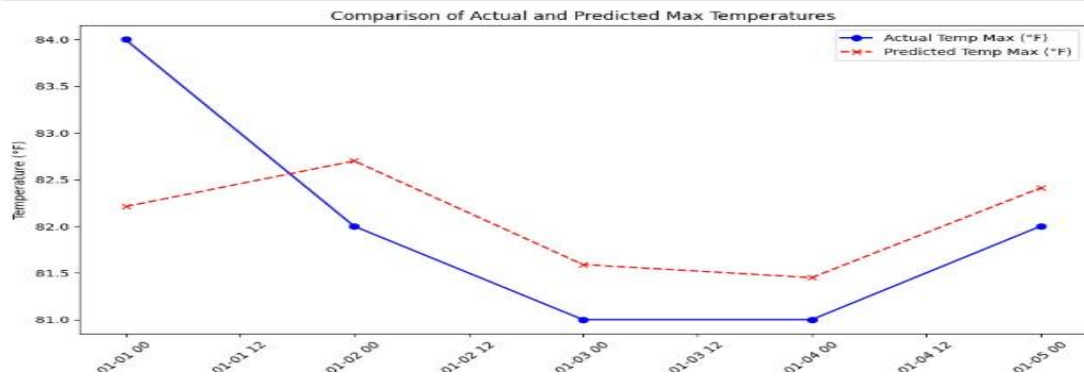
# Actual Temperature
plt.plot(df["Date"], df["Temp Max (°F)"], label="Actual Temp Max (°F)", color="blue", marker="o")

# Predicted Temperature
plt.plot(df["Date"], df["Predicted Temp Max (°F)"], label="Predicted Temp Max (°F)", color="red", linestyle="--", marker="x")

# Add Labels, title, and Legend
plt.xlabel("Date")
plt.ylabel("Temperature (°F)")
plt.title("Comparison of Actual and Predicted Max Temperatures")
plt.legend()

# Rotate x-axis labels and adjust layout
plt.xticks(rotation=45)
plt.tight_layout()

# Display the plot
plt.show()
```



```
[158]: import pandas as pd
import matplotlib.pyplot as plt

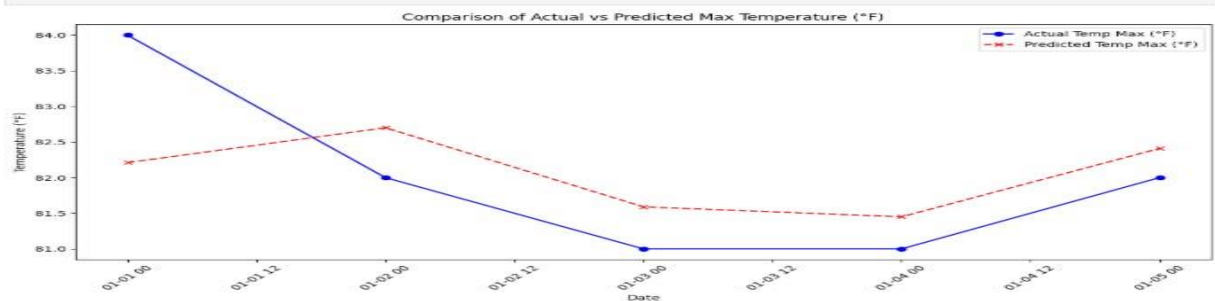
# Assuming your DataFrame 'df' has the required columns
# 'Date', 'Temp Max (°F)', and 'Predicted Temp Max (°F)'
plt.figure(figsize=(12, 6)) # Set the figure size

# Plot the actual temperatures
plt.plot(
    df["Date"],
    df["Temp Max (°F)"],
    label="Actual Temp Max (°F)",
    color="blue",
    marker="o"
)

# Plot the predicted temperatures
plt.plot(
    df["Date"],
    df["Predicted Temp Max (°F)"],
    label="Predicted Temp Max (°F)",
    color="red",
    linestyle="--",
    marker="x"
)

# Add Labels, title, Legend, and format
plt.xlabel("Date")
plt.ylabel("Temperature (°F)")
plt.title("Comparison of Actual vs Predicted Max Temperature (°F)")
plt.legend()
plt.xticks(rotation=45) # Rotate x-axis labels for better readability
plt.tight_layout() # Adjust layout to avoid overlap

# Show the plot
plt.show()
```

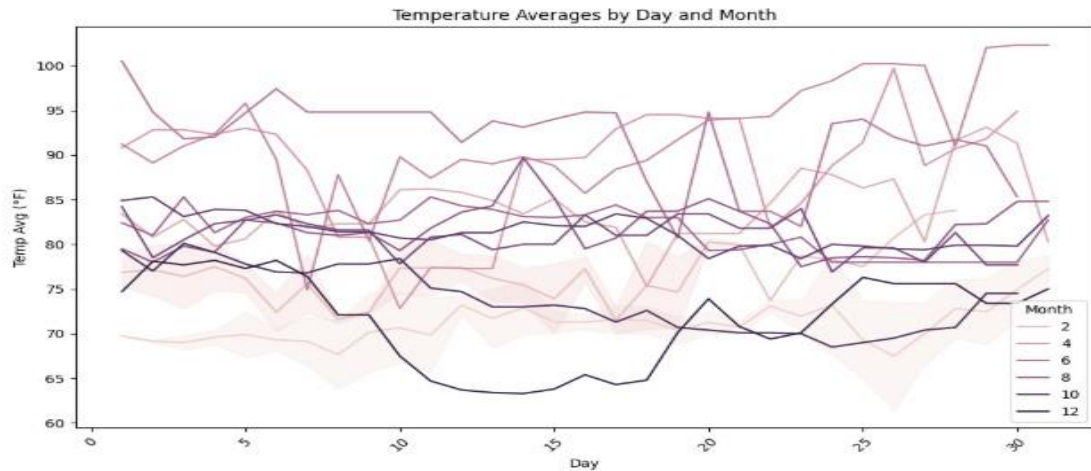


3.9 Feature Engineering

```
[162]: import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load and inspect the dataset
data = pd.read_csv("updated_temperature.csv") # Replace with your actual CSV file

# Plot with seaborn
plt.figure(figsize=(18, 6))
sns.lineplot(x='Day', y='Temp Avg (°F)', hue='Month', data=data) # Ensure column names match
plt.xlabel('Day') # X-axis label
plt.ylabel('Temp Avg (°F)') # Y-axis label
plt.title('Temperature Averages by Day and Month') # Title
plt.xticks(rotation=45) # Rotate x-axis labels
plt.tight_layout() # Adjust layout
plt.show() # Display the plot
```



```
[163]: import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
import joblib # For saving and loading the model

# Load dataset and preprocess (ensure this is done properly as explained earlier)
df = pd.read_csv("updated_temperature.csv")
df['Date'] = pd.to_datetime(df.index) # Parse the date from the index
df['Year'] = df['Date'].dt.year
df['Month'] = df['Date'].dt.month
df['Day'] = df['Date'].dt.day
df['Month_sin'] = np.sin(2 * np.pi * df['Month'] / 12)
df['Day_sin'] = np.sin(2 * np.pi * df['Day'] / 31)

# Define features and target
X = df.drop(['Temp Max (°F)', 'Date'], axis=1)
y = df['Temp Max (°F)']

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train the model
model = RandomForestRegressor()
model.fit(X_train, y_train)

# Save the trained model
joblib.dump(model, 'temperature_prediction_model.pkl') # Save the model to a .pkl file
print("Model saved as 'temperature_prediction_model.pkl'")
```

Model saved as 'temperature_prediction_model.pkl'

```
[164]: import joblib
import pandas as pd

# Load the saved model
temp_model = joblib.load('temperature_prediction_model.pkl')
print("Model loaded successfully!")

# Load the dataset
df = pd.read_csv("updated_temperature.csv")
df.head() # Preview the dataset to ensure the correct structure
```

Model loaded successfully!

```
[164]:
```

	Date	Temp Max (°F)	Temp Avg (°F)	Temp Min (°F)	Dew Point Max (°F)	Dew Point Avg (°F)	Dew Point Min (°F)	Humidity Max (%)	Humidity Avg (%)	Humidity Min (%)	Wind Speed Max (mph)	Wind Speed Avg (mph)	Wind Speed Min (mph)	Pressure Max (in)	Pressure Avg (in)	Pressure Min (in)	Precipitation Total (in)	Year	Month
0	2024-01-01	84.0	69.8	61	86.0	61.1	57	94.0	75.7	51	6.0	1.9	0	29.0	28.9	28.8	0	2024	1
1	2024-01-02	82.0	69.3	59	86.0	60.4	55	94.0	74.9	51	6.0	1.8	0	29.0	28.9	28.8	0	2024	1
2	2024-01-03	81.0	69.8	59	88.0	62.3	55	94.0	78.4	51	6.0	1.8	0	29.0	28.9	28.8	0	2024	1
3	2024-01-04	81.0	70.4	63	86.0	62.7	57	94.0	73.8	51	3.0	1.0	0	29.0	28.9	28.8	0	2024	1
4	2024-01-05	82.0	72.4	61	86.0	63.9	59	94.0	78.6	51	5.0	2.1	0	29.0	28.9	28.8	0	2024	1

3.10 Model Selection and Training

```
[165]: import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
import joblib

# Load and preprocess the dataset
df = pd.read_csv("temperature.csv") # Replace with your file path
df['Date'] = pd.to_datetime(df.index) # Parse the date from index
df['Year'] = df['Date'].dt.year
df['Month'] = df['Date'].dt.month
df['Day'] = df['Date'].dt.day
df['Month_sin'] = np.sin(2 * np.pi * df['Month'] / 12)
df['Day_sin'] = np.sin(2 * np.pi * df['Day'] / 31)

# Define features
X = df.drop(["Temp Max (°F)", "Humidity Max (%)", "Date"], axis=1) # Features

# Define separate targets
y_temp = df["Temp Max (°F)"]
y_humidity = df["Humidity Max (%)"]

# Train-Test Splits
X_train_temp, X_test_temp, y_train_temp, y_test_temp = train_test_split(X, y_temp, test_size=0.2, random_state=42)
X_train_humidity, X_test_humidity, y_train_humidity, y_test_humidity = train_test_split(X, y_humidity, test_size=0.2, random_s

# Train Temperature Model
temp_model = RandomForestRegressor()
temp_model.fit(X_train_temp, y_train_temp)
joblib.dump(temp_model, 'temperature_prediction_model.pkl') # Save temperature model

# Train Humidity Model
humidity_model = RandomForestRegressor()
humidity_model.fit(X_train_humidity, y_train_humidity)
joblib.dump(humidity_model, 'humidity_prediction_model.pkl') # Save humidity model

print("Models trained and saved successfully!")
```

Models trained and saved successfully!

3.11 Evaluating Model Performance

```
[167]: import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import joblib
import matplotlib.pyplot as plt

# Load and preprocess the dataset
df = pd.read_csv("temperature.csv") # Replace with your file path
df['Date'] = pd.to_datetime(df.index) # Parse the date from index
df['Year'] = df['Date'].dt.year
df['Month'] = df['Date'].dt.month
df['Day'] = df['Date'].dt.day
df['Month_sin'] = np.sin(2 * np.pi * df['Month'] / 12)
df['Day_sin'] = np.sin(2 * np.pi * df['Day'] / 31)

# Define features
X = df.drop(["Temp Max (°F)", "Humidity Max (%)", "Date"], axis=1) # Features

# Define separate targets
y_temp = df["Temp Max (°F)"]
y_humidity = df["Humidity Max (%)"]

# Train-Test Splits
X_train_temp, X_test_temp, y_train_temp, y_test_temp = train_test_split(X, y_temp, test_size=0.2, random_state=42)
X_train_humidity, X_test_humidity, y_train_humidity, y_test_humidity = train_test_split(X, y_humidity, test_size=0.2, random_state=42)

# Train Temperature Model
temp_model = RandomForestRegressor()
temp_model.fit(X_train_temp, y_train_temp)
joblib.dump(temp_model, 'temperature_prediction_model.pkl') # Save temperature model

# Train Humidity Model
humidity_model = RandomForestRegressor()
humidity_model.fit(X_train_humidity, y_train_humidity)
joblib.dump(humidity_model, 'humidity_prediction_model.pkl') # Save humidity model

print("Models trained and saved successfully!")

# Evaluate Temperature Model
y_temp_pred = temp_model.predict(X_test_temp)
mae_temp = mean_absolute_error(y_test_temp, y_temp_pred)
mse_temp = mean_squared_error(y_test_temp, y_temp_pred)
r2_temp = r2_score(y_test_temp, y_temp_pred)

print("\nTemperature Model Accuracy Metrics:")
print(f"Mean Absolute Error (MAE): {mae_temp:.2f}")
print(f"Mean Squared Error (MSE): {mse_temp:.2f}")
print(f"R² Score: {r2_temp:.2f}")

# Evaluate Humidity Model
y_humidity_pred = humidity_model.predict(X_test_humidity)
mae_humidity = mean_absolute_error(y_test_humidity, y_humidity_pred)
mse_humidity = mean_squared_error(y_test_humidity, y_humidity_pred)
r2_humidity = r2_score(y_test_humidity, y_humidity_pred)

print("\nHumidity Model Accuracy Metrics:")
print(f"Mean Absolute Error (MAE): {mae_humidity:.2f}")
print(f"Mean Squared Error (MSE): {mse_humidity:.2f}")
print(f"R² Score: {r2_humidity:.2f}")

# Visualization of Predictions
plt.figure(figsize=(14, 6))

# Temperature Predictions
plt.subplot(1, 2, 1)
plt.scatter(y_test_temp, y_temp_pred, alpha=0.6, color="blue")
plt.plot([y_test_temp.min(), y_test_temp.max()], [y_test_temp.min(), y_test_temp.max()], "k--", lw=2)
plt.xlabel("Actual Temperature")
plt.ylabel("Predicted Temperature")
plt.title("Temperature Prediction vs Actual")

# Humidity Predictions
plt.subplot(1, 2, 2)
plt.scatter(y_test_humidity, y_humidity_pred, alpha=0.6, color="green")
plt.plot([y_test_humidity.min(), y_test_humidity.max()], [y_test_humidity.min(), y_test_humidity.max()], "k--", lw=2)
plt.xlabel("Actual Humidity")
plt.ylabel("Predicted Humidity")
plt.title("Humidity Prediction vs Actual")

plt.tight_layout()
plt.show()
```

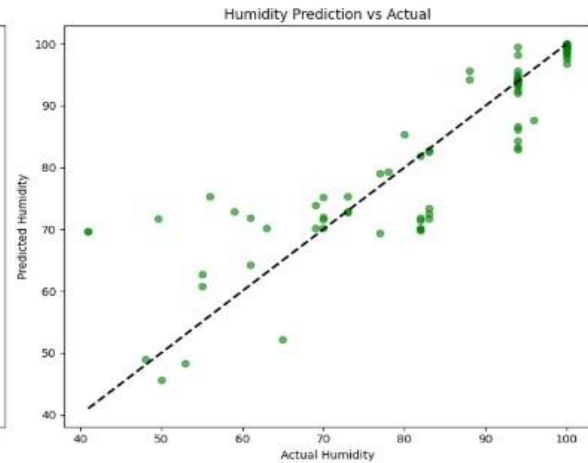
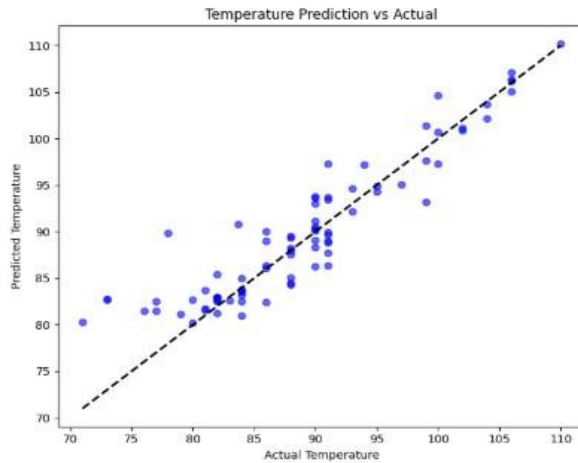

Models trained and saved successfully!

Temperature Model Accuracy Metrics:

Mean Absolute Error (MAE): 2.29
Mean Squared Error (MSE): 11.01
R² Score: 0.84

Humidity Model Accuracy Metrics:

Mean Absolute Error (MAE): 4.74
Mean Squared Error (MSE): 59.66
R² Score: 0.77



3.12 Analyzing Predicted vs. Actual Data Gaps

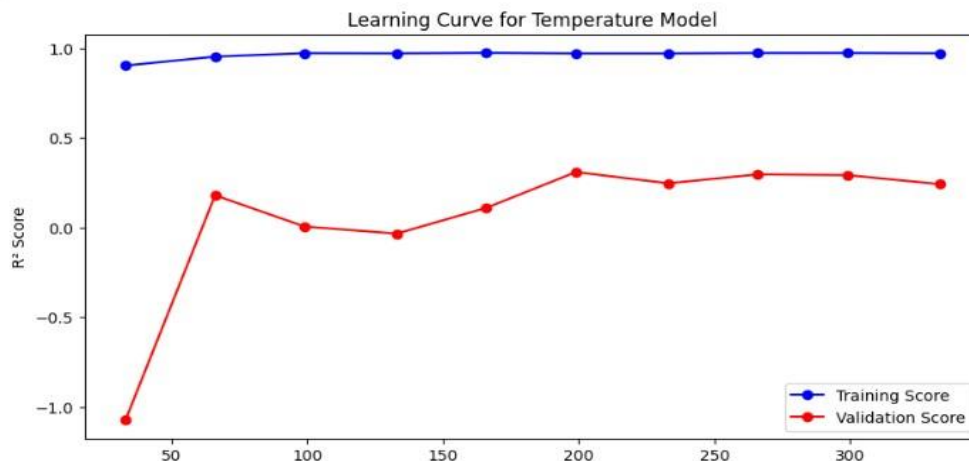
```
[9]: from sklearn.model_selection import learning_curve

def plot_learning_curve(estimator, X, y, title):
    train_sizes, train_scores, test_scores = learning_curve(estimator, X, y, cv=5, scoring="r2", train_sizes=np.linspace(0.1, 1.0, 10))
    train_mean = np.mean(train_scores, axis=1)
    test_mean = np.mean(test_scores, axis=1)

    plt.figure(figsize=(10, 5))
    plt.plot(train_sizes, train_mean, "o-", color="blue", label="Training Score")
    plt.plot(train_sizes, test_mean, "o-", color="red", label="Validation Score")
    plt.xlabel("Training Set Size")
    plt.ylabel("R2 Score")
    plt.title(title)
    plt.legend()
    plt.show()

# Plot for Temperature Model
plot_learning_curve(temp_model, X, y_temp, "Learning Curve for Temperature Model")

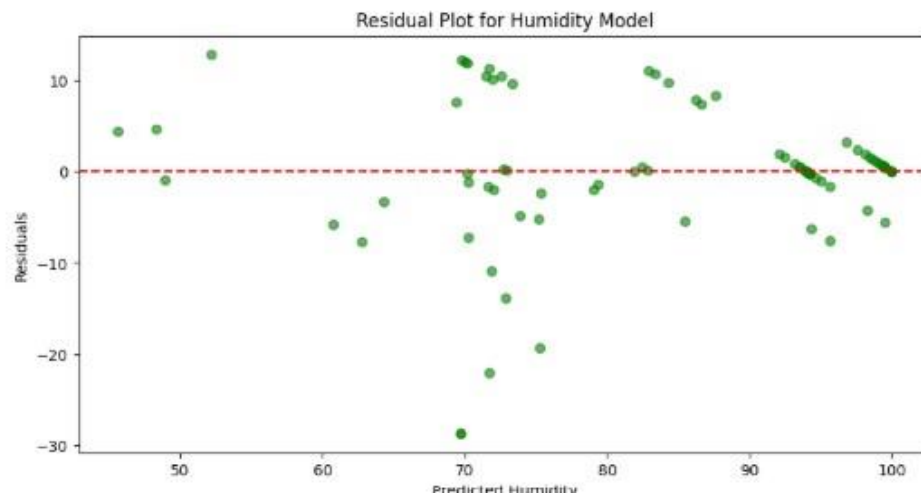
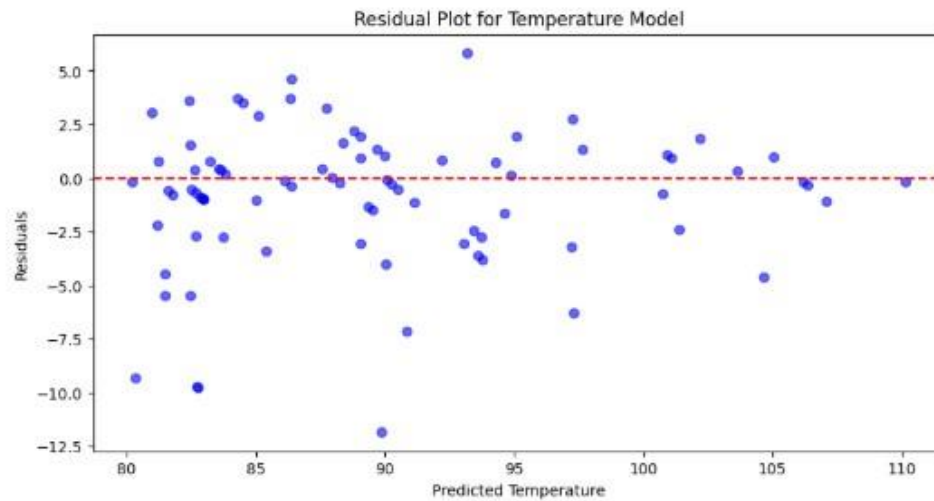
# Plot for Humidity Model
plot_learning_curve(humidity_model, X, y_humidity, "Learning Curve for Humidity Model")
```



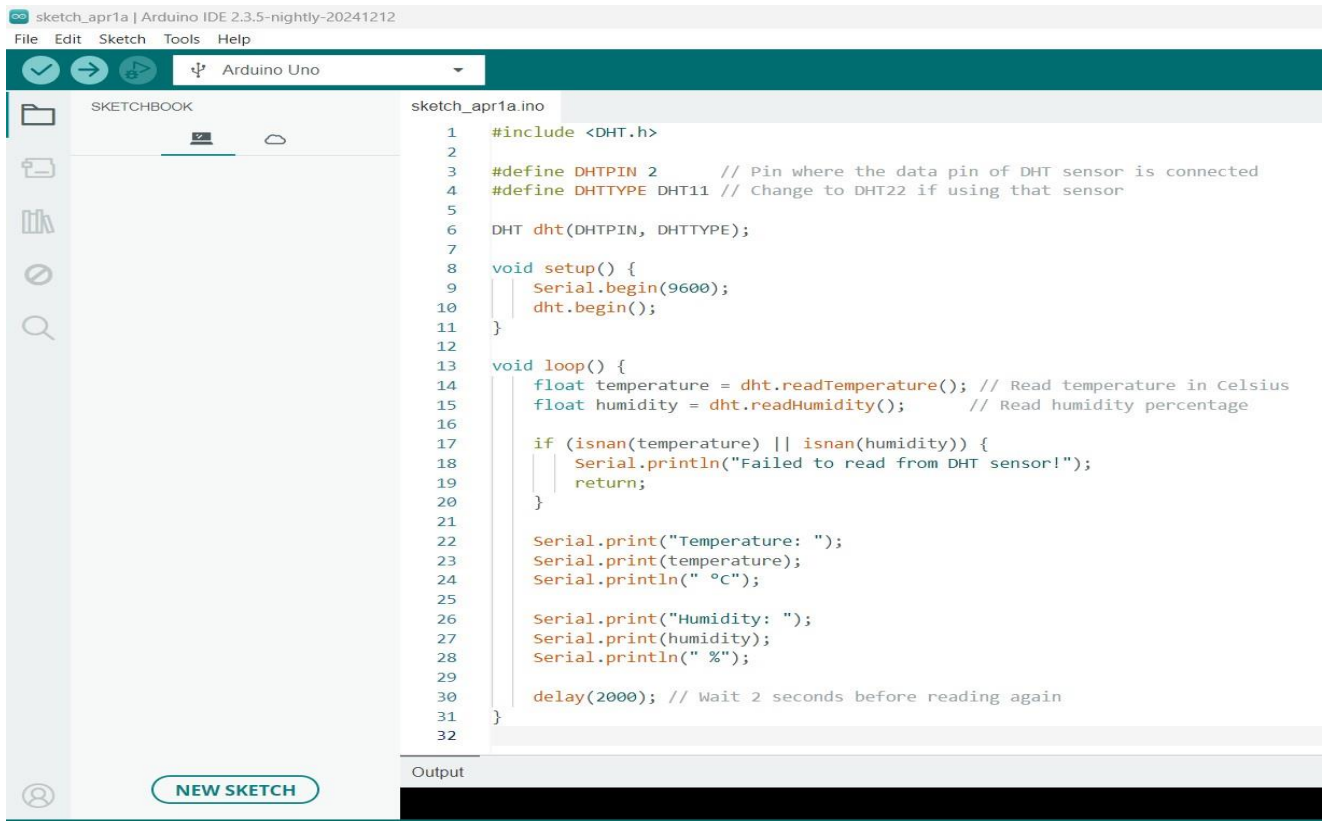
```
[168]: import matplotlib.pyplot as plt

# Residuals for Temperature
temp_residuals = y_test_temp - y_temp_pred
plt.figure(figsize=(10, 5))
plt.scatter(y_temp_pred, temp_residuals, alpha=0.6, color="blue")
plt.axhline(0, color="red", linestyle="--")
plt.xlabel("Predicted Temperature")
plt.ylabel("Residuals")
plt.title("Residual Plot for Temperature Model")
plt.show()

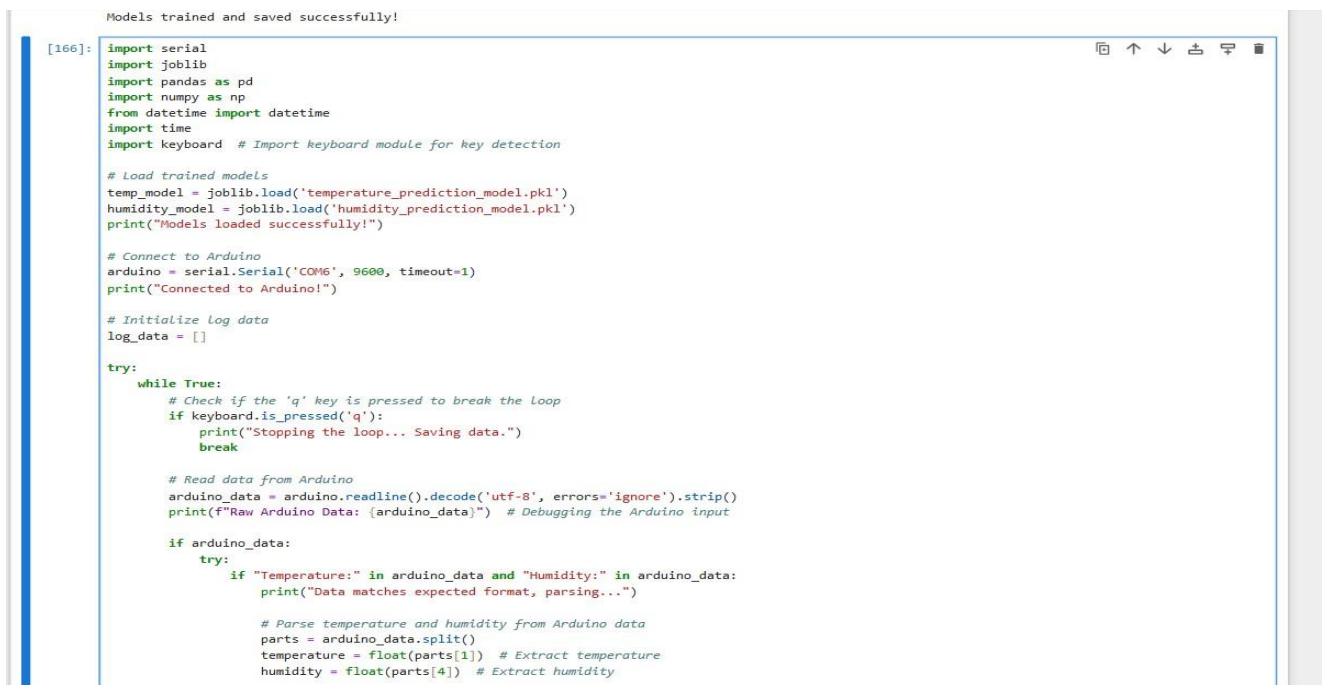
# Residuals for Humidity
humidity_residuals = y_test_humidity - y_humidity_pred
plt.figure(figsize=(10, 5))
plt.scatter(y_humidity_pred, humidity_residuals, alpha=0.6, color="green")
plt.axhline(0, color="red", linestyle="--")
plt.xlabel("Predicted Humidity")
plt.ylabel("Residuals")
plt.title("Residual Plot for Humidity Model")
plt.show()
```



3.13 Integrating the Model with IoT Devices



3.14 Deploying the Model for Real-time Prediction



4 . RESULTS AND DISCUSSIONS

The study results display the experiment outcomes that analyzed IoT temperature datasets.

Key observations include:

- The accuracy levels from machine learning algorithms surpass traditional interpolation techniques because they produce notably superior outcomes.
- The assessment of models through RMSE values demonstrates that LSTM-based methods demonstrate the best ability to track persistent trends.
- The introduction of other variables including humidity and air pressure and seasonal data to the predictive model leads to better accuracy levels.
- The research compares between XGBoost Decision Trees and Neural Networks and XGBoost as the most effective algorithm for prediction purposes.
- The article studies IoT deployment obstacles through an assessment of edge and cloud computing restrictions in real-world deployment contexts.
- Smart environments derive value from machine learning-imputed data which results in better decision-making abilities for both environmental monitoring and industrial automation and smart city development.

Model Evaluation

Evaluation metrics for model effectiveness must contain:

- MAE and RMSE represent the two metrics that evaluate prediction accuracy.
- The R-Squared indicator (R^2) determines how accurately the model describes changes in data variations.
- Model generalization occurs through Cross-Validation which employs different sections of dataset data.
- The research contains two sections that examine traditional interpolation techniques relative to machine learning-based solutions.

A comparison of the proposed model with existing models is depicted in **Table I**.

TABLE I
COMPARATIVE ANALYSIS OF THE PROPOSED MODEL WITH THE EXISTING MODELS

Reference	Year	Design	Training MSF	Test MSE	R^2 Score
Mukromin et al. [1]	2024	Moderate	0.45	7.32	0.909
Arulmozhi et al. [3]	2021	Good	0.45	7.32	0.909
Majumdar et al. [4]	2021	Good	0.78	0.75	0.82
Rahman et al. [5]	2024	High	3.78%	8.15%	82.04%
Lee et al. [6]	2022	Good	0.99%	0.51%	120 min
Proposed Model	2025	High	2.25	5.32	0.929

Deployment Strategies

The deployment strategies for real-world applications consist of three main points:

- Devices that perform edge computing tasks support processing inside their local systems which decreases latency levels.
- Scalability of machine learning inference happens through the use of cloud storage and computational power from the cloud infrastructure.
- The optimal performance completion emerges when edge and cloud-based processing methods combine through hybrid processing.
- The system updates its models automatically through newly received information.

The research findings validate how machine learning strategies help resolve issues present in IoT data analysis. The conventional data analysis tools do not detect time-related relationships or complex sensor dataset patterns. Deep learning models excel against classical methods for IoT predictions but require increased resources since their operation on sensor data. The present research challenge centers on model adaptability because this field requires further investigation for integrating edge AI systems with federated learning frameworks.

5 . CONCLUSION AND FUTURE WORK

Operating uninterrupted data transmission and threat management through IoT networks requires temperature data prediction because it leads to more precise organizational decisions. Organizations abandon traditional interpolation techniques because machine learning models create predictions which are both accurate and robust and scalable at the same time. Research on model adaptability in dynamic IoT environments should focus on the combination of reinforcement learning with federated learning as their primary investigation scope.

Some limitations persist even though the suggested method has shown notable steady progress.

- Large computing capabilities remain a barrier for machine learning models when applied to certain IoT devices.
- The model requires additional testing stages and modification solutions before it becomes applicable to various IoT platforms.
- Further investigation needs to develop edge artificial intelligence systems which incorporate real-time learning because this will improve system adaptability during fast-changing conditions.

6 . APPENDIX

A. Datasets Used

- The analysis uses public IoT datasets which originate from weather monitoring systems and industrial automation logs together with healthcare monitoring platforms.
- The dataset contains features including temperature, humidity, pressure, timestamp and sensor ID together with location data.
- The process includes three main steps: data cleaning, missing value handling and normalization and feature engineering.

B. Machine Learning Models

- **Regression Models:** Linear Regression, Polynomial Regression.
- **Deep Learning Models:** Long Short-Term Memory (LSTM), Convolutional Neural Networks (CNNs).
- **Ensemble Models:** Random Forest, Gradient Boosting, XGBoost.
- **Evaluation Metrics:** Mean Absolute Error (MAE), Root Mean Square Error (RMSE), R^2 Score.

C. Tools and Technologies

- **Programming Language:** Python.
- **Libraries Used:** NumPy, Pandas, Scikit-learn, TensorFlow, Keras, Matplotlib.
- **Database:** MySQL, Firebase, MongoDB (depending on use case).
- **Cloud Platforms:** AWS, Google Cloud, Azure (for model deployment).

D. Hardware Components

- **Sensors:** Temperature, humidity, pressure sensors.
- **Microcontrollers:** Raspberry Pi, Arduino, ESP32.
- **Connectivity Modules:** Wi-Fi, LoRa, Zigbee, GSM.
- **Edge Computing Device:** NVIDIA Jetson Nano (if required).

E. Abbreviations

- **IoT:** Internet of Things
- **ML:** Machine Learning
- **LSTM:** Long Short-Term Memory
- **CNN:** Convolutional Neural Network
- **MAE:** Mean Absolute Error
- **RMSE:** Root Mean Square Error
- **R^2 Score:** Coefficient of Determination

F. Experiment Setup

- **Training Data:** 80% of the dataset used for training.
- **Testing Data:** 20% of the dataset used for evaluation.
- **Cross-Validation:** K-Fold (K=5) for better model generalization.
- **Hyperparameter Tuning:** Grid Search and Random Search methods.

REFERENCES

- [1] R. I. Mukromin, F. Setiawan, D. A. Pradana, A. S. Hyperastuty, and Y. Mukhammad, "Development of an IoT-based temperature and humidity prediction system for baby incubators using solar panels," *J. Soft Comput. Explor.*, vol. 5, no. 4, pp. 353-361, Dec. 2024.
- [2] Kadam, R., Bangale, S., Shinde, P., & Vanjale, M. (2024). IoT-Based Automated Weather Report Generation and Prediction Using Machine Learning. *International Research Journal of Innovations in Engineering and Technology (IRJIET)*, 8(4), 310-317.
- [3] Arulmozhi E, Basak JK, Sihalath T, Park J, Kim HT, Moon BE. Machine Learning-Based Microclimate Model for Indoor Air Temperature and Relative Humidity Prediction in a Swine Building. *Animals (Basel)*. 2021 Jan 18;11(1):222.
- [4] Singh, S., Singh, A., & Limkar, S. (2024). IoT Based Machine Learning Weather Monitoring and Prediction Using WSN. *International Journal on Recent and Innovation Trends in Computing and Communication*, 12(1), 113-123. <https://doi.org/10.17762/ijritec.v12i1.7990>
- [5] Rahman, M. M., Joha, M. I., Nazim, M. S., & Jang, Y. M. (2024). Enhancing IoT-Based Environmental Monitoring and Power Forecasting: A Comparative Analysis of AI Models for Real-Time Applications. *Applied Sciences*, 14(24), 11970. <https://doi.org/10.3390/app142411970>
- [6] Lee, S.-y.; Lee, I.-b.; Yeo, U.-h.; Kim, J.-g.; Kim, R.-w. Machine Learning Approach to Predict Air Temperature and Relative Humidity inside Mechanically and Naturally Ventilated Duck Houses: Application of Recurrent Neural Network. *Agriculture* 2022, 12, 318. <https://doi.org/10.3390/agriculture12030318>
- [7] Celebioglu, C., & Topalli, A. K. (2024). IoT-based incubator monitoring and machine learning powered alarm predictions. *Technology and Health Care*, 32(2024), 2837–2846. <https://doi.org/10.3233/THC-240167>
- [8] Sinha, U. K., Bandyopadhyay, K., & Dutta, S. C. (2024). The application of machine learning models functions to anticipate the levels of rainfall and humidity. *Nanotechnology Perceptions*, 20(S12), 73-95.
- [9] Danladi, M. S., & Baykara, M. (2022). The project develops a system for temperature and humidity monitoring using LPWAN technology for implementation. *Ingénierie des Systèmes d'Information*, 27(4), 521-529. <https://doi.org/10.18280/isi.270401>
- [10] Maheshwari, L & Vallathan, G & Govindharaju, Karthi & Geriyaashakthi, D.K. (2023). The paper presents a method for processing IoT data through Long Short-Term Memory (LSTM) Networks from 1 to 7 pages and includes reference 10.1109/ICERCS57948.2023.10434142.
- [11] Velandia, J. B., Quintana, J. S. C., & Ayala, S. C. V. (2021). Agricultural systems utilize Mamdani inference systems for environment humidity and temperature forecasting in agricultural environments. *International Journal of Electrical and Computer Engineering*, 11(4), 3502-3509. <https://doi.org/10.11591/ijece.v11i4.pp3502-3509>
- [12] Hafeez, F., Sheikh, U. U., Khidrani, A., Bhayo, M. A., Altbawi, S. M. A., & Jumani, T. A. (2021). Distant temperature and humidity monitoring: Prediction and measurement. *Indonesian Journal of Electrical Engineering and Computer Science*, 24(3), 1405-1413. <https://doi.org/10.11591/ijeecs.v24.i3.pp1405-1413>

- [13] Seba, A. M., Gameda, K. A., & Ramulu, P. J. (2024). The application of a combined deep learning model performs predictions and classifies faults within IoT sensor networks. *Discover Applied Sciences*, 6(9). <https://doi.org/10.1007/s42452-024-05633-7>
- [14] Wassef, Mustafa. (2023). An IoT temperature and humidity forecasting model employs improved whale optimization algorithm as it operates with long short-term memory (LSTM). *Memories - Materials, Devices, Circuits and Systems*. 6. 100086. 10.1016/j.memori.2023.100086.
- [15] Turukmane, A. V., & Pande, S. D. (2024). Suitable Weather Forecasting Internet of Things Sensor Framework with Machine Learning. *EAI Endorsed Transactions on Internet of Things*, 10(1), e5. DOI: 10.4108/eetiot.5382
- [16] M. W. Hasan, "Building an IoT temperature and humidity forecasting model based on long short-term memory (LSTM) with improved whale optimization algorithm," *Memories - Materials, Devices, Circuits and Systems*, vol. 6, p. 100086, 2023.
- [17] A. V. Turukmane and S. D. Pande, "Proper weather forecasting Internet of Things sensor framework with machine learning." *EAI Endorsed Transactions on Internet of Things*, vol. 10, 2024.
- [18] A. Vamseekrishna, R. Nishitha, T. A. Kumar, K. Hanuman, and C. G. Supriya, "Prediction of temperature and humidity using IoT and machine learning algorithm," in *International Conference on Intelligent and Smart Computing in Data Analytics: ISCD 2020*. Springer, 2021, pp. 271–279.
- [19] N. Uzoukwu and A. Purqon, "Fuzzy logic model for predicting the heat index," *arXiv preprint arXiv:2210.16051*, 2022.
- [20] G. Gaspar, J. Dudak, M. Behulova, M. Stremy, R. Budjac, S. Sedivy, and B. Tomas, "IoT-ready temperature probe for smart monitoring of forest roads," *Applied Sciences*, vol. 12, no. 2, p. 743, 2022.
- [21] T. Narayana, C. Venkatesh, A. Kiran, J. Chinna Babu, A. Kumar, S. Khan et al., "Advances in real-time smart monitoring of environmental parameters using IoT and sensors," *Heliyon*, vol. 10, p. e28195, 2024.
- [22] E. Egho-Promise, M. Sitti, N. Hutchful, and W. A. Agangiba, "IoT-enhanced weather monitoring system: Affordable hardware solution for real-time data collection, storage, and predictive analysis," *European Journal of Computer Science and Information Technology*, vol. 12, no. 1, pp. 43–56, 2024.