**Weather Prediction and Forecasting using SARIMA**

**Submitted by:**

**Dhawal Duggar, 21BEC1844**

**Jagjeet Singh Sehgal, 21BEC1229**

# Preface

1.Objectives…………………………………………………………………………………………01

2.Abstract………………………………………………………………………………………… 02

1. Background Study……………………………………………………………………………02
2. Code - Implementation and Explanation ……………………………………………………………03 5.Conclusion…………………………………………………………………………………….31

# Objectives

We are going to build a Weather Prediction Model using the machine learning algorithm to do prediction of weather on the basis of live data gathered by the use of sensor. Hence creating a system that is accurate enough for particularly smaller cities.

# Abstract

# Seasonal Autoregressive Integrated Moving Average (SARIMA) model and its application in a ThingSpeak system.

1. **SARIMA Model Overview**:
   * **SARIMA** (Seasonal ARIMA) is an extension of the ARIMA (AutoRegressive Integrated Moving Average) model.
   * It is particularly useful for time series data with **seasonal patterns**.
   * The components of SARIMA include:
     + **Seasonal Autoregressive (SAR) terms**: Captures the relationship between the current value and past values at seasonal intervals.
     + **Seasonal Moving Average (SMA) terms**: Accounts for the influence of past forecast errors at seasonal intervals.
     + **Integration (I) term**: Represents the number of differences needed to make the data stationary.
   * SARIMA models are widely used for various applications, including weather forecasting.
2. **Why SARIMA for Weather Forecasting?**:
   * Weather data often exhibits **seasonal patterns** (e.g., temperature variations across months).
   * SARIMA can handle non-stationary data (where the mean and variance change over time) and incorporate seasonality.
   * For instance, this month’s temperature may depend on the previous month’s temperature, making SARIMA suitable.
   * Additionally, SARIMA considers white noise error terms, which aligns with real-world weather fluctuations.
3. **Python Implementation Example**:
   * To demonstrate SARIMA, let’s consider Istanbul’s temperature data.
   * We’ll convert daily data into monthly data due to computational limitations (daily data with SARIMA can be resource-intensive).
4. **ThingSpeak Integration**:
   * ThingSpeak is a platform for collecting, analyzing, and visualizing IoT data.
   * To apply SARIMA in a ThingSpeak system:
     + Utilize temperature and humidity sensors (e.g., DHT11).
     + Upload sensed data to ThingSpeak using devices like NodeMCU and ESP8266-01.
     + Display real-time weather values on a customized HTML webpage for monitoring.

# Background Study

Certainly! Let’s delve into the background study for your research on weather prediction using the **SARIMA model** in conjunction with **ThingSpeak**, incorporating live data from the **DHT11 sensor** and **NodeMCU**.

### 1. Introduction

Weather prediction plays a crucial role in various domains, from agriculture to sports events. The ability to forecast weather conditions accurately enables better decision-making and planning. In this study, we propose a real-time weather prediction system that leverages IoT (Internet of Things) and machine learning techniques.

### 2. Components of the System

* **NodeMCU**: NodeMCU, based on the ESP8266 Wi-Fi SoC, serves as our IoT device. It includes necessary sensors, actuators, and a communication interface.
* **DHT11 Sensor**: The DHT11 sensor measures temperature and humidity. It provides essential data for weather prediction.
* **ThingSpeak**: We use ThingSpeak as our cloud service provider. It allows us to upload sensor data and monitor it remotely.
* **Machine Learning Model**: We employ a logistic regression model for weather prediction. This model is trained using pre-recorded sensor data.

### 3. System Workflow

1. **Data Sensing**:
   * The DHT11 sensor continuously measures temperature and humidity.
   * NodeMCU collects data from the sensor.
2. **Data Upload to ThingSpeak**:
   * NodeMCU uploads the sensed data to the ThingSpeak cloud server.
   * ThingSpeak stores the data for monitoring and analytics.
3. **Real-Time Monitoring**:
   * A customized HTML webpage displays real-time temperature and humidity values.
   * Users can access this webpage for live weather information.
4. **Machine Learning Model**:
   * We train a logistic regression model using historical sensor data.
   * The model learns the relationship between temperature, humidity, and weather conditions.
5. **Prediction and Actuation**:
   * NodeMCU records real-time data.
   * The model predicts weather conditions based on the input data.
   * For demonstration, we blink an LED connected to NodeMCU to indicate predicted weather (e.g., rain, sunny).

### 4. Significance and Applications

* **Smart Homes**: Residents can receive real-time weather updates for better planning (e.g., adjusting thermostat settings).
* **Agriculture**: Farmers can make informed decisions regarding irrigation, crop protection, and planting.
* **Stadiums and Events**: Organizers can decide whether to proceed with outdoor events based on weather predictions.

### 5. Conclusion

The proposed real-time weather prediction system demonstrates the synergy between IoT, machine learning, and low-cost hardware. By integrating DHT11 sensor data, ThingSpeak, and NodeMCU, we create a practical solution for accurate weather forecasting.

## Hardware Associated

## 

The Hardware involves:  
1. NodeMCU  
2. DHT11 sensor  
3. Breadboard   
4. Jumper Wires

Connections:  
The DHT pins GND and VCC are connected to GND and Vin respectively, and the Data Pin is connected to D4 port using the jumper wires and the Breadboard.

# CODE and Its Implementation

SARIMA Model for the prediction of weather data based on the given weather data.

import pandas as pd

from statsmodels.tsa.statespace.sarimax import SARIMAX

from statsmodels.tsa.stattools import adfuller

import warnings

# Load dataset

try:

    data = pd.read\_csv('weather\_data\_test1.csv')  # Replace with your CSV file path

except FileNotFoundError:

    print("Error: File 'weather\_data\_test1.csv' not found. Please provide the correct file path.")

    exit(1)

# Check for stationarity

def test\_stationarity(timeseries):

    dftest = adfuller(timeseries, autolag='AIC')

    return dftest[1]

# Assuming 'Temperature' is the column with time series data

try:

    p\_value = test\_stationarity(data['Temperature'])

except KeyError:

    print("Error: Column 'Temperature' not found in the dataset. Please check the column name.")

    exit(1)

if p\_value > 0.05:

    print("Data is non-stationary. Differencing needed.")

else:

    print("Data is stationary. No differencing needed.")

# Fit SARIMA model (p,d,q)(P,D,Q,s) values to be determined as per your dataset

try:

    model = SARIMAX(data['Temperature'], order=(1, 1, 1), seasonal\_order=(1, 1, 1, 12))

    results\_SARIMA = model.fit()

    # Predict future values

    predictions = results\_SARIMA.get\_forecast(steps=7)  # Predict next 7 values (1 week)

    # Extract predicted mean and confidence intervals

    predicted\_mean = predictions.predicted\_mean

    confidence\_intervals = predictions.conf\_int()

    # Create a DataFrame with predictions and confidence intervals

    predictions\_df = pd.DataFrame({

        'Predicted\_Avg\_Temperature': predicted\_mean,

        'Minimum Temperature': confidence\_intervals.iloc[:, 0],

        'Maximum Temperature': confidence\_intervals.iloc[:, 1]

    })

    # Save predictions to CSV

    predictions\_df.to\_csv('sarima\_predictions.csv', index=False)

    print("Predictions saved to sarima\_predictions.csv")

except Exception as e:

    print(f"Error: {e}")

    exit(1)

Algorithm to convert SARIMA output CSV to ThingSpeak importable csv file.

import pandas as pd

from datetime import datetime, timedelta

# Load the CSV file into a DataFrame

df = pd.read\_csv('sarima\_predictions.csv')

# Create a 'created\_at' column with timestamps starting from now and incrementing by 1 minute for each entry

start\_time = datetime.now()

df['created\_at'] = [start\_time + timedelta(minutes=i) for i in range(len(df))]

# Rename the 'Predicted\_Avg\_Temperature' column to 'field1' as required by ThingSpeak

df.rename(columns={'Predicted\_Avg\_Temperature': 'field1'}, inplace=True)

# Check if 'Minimum Temperature' and 'Maximum Temperature' columns exist in the DataFrame

if 'Minimum Temperature' in df.columns and 'Maximum Temperature' in df.columns:

    # Rename the 'Minimum Temperature' and 'Maximum Temperature' columns to 'field2' and 'field3' respectively

    df.rename(columns={'Minimum Temperature': 'field2', 'Maximum Temperature': 'field3'}, inplace=True)

# Reorder the DataFrame columns to match the ThingSpeak import requirements

df = df[['created\_at', 'field1', 'field2', 'field3']]

# Convert the 'created\_at' column to the ISO 8601 format with a UTC offset

df['created\_at'] = df['created\_at'].dt.strftime('%Y-%m-%dT%H:%M:%S%z')

# Save the modified DataFrame to a new CSV file

df.to\_csv('thing\_speak\_importable.csv', index=False, encoding='utf-8')

# Print a success message

print("The file has been converted to ThingSpeak importable form and saved as 'thing\_speak\_importable.csv'")

**Arduino IDE code:**  
  
#include <DHT.h>

#include <Wire.h>

#define DHTPIN 4 // Digital pin connected to the DHT sensor

#define DHTTYPE DHT11 // DHT 11

DHT dht(DHTPIN, DHTTYPE);

void setup() {

Serial.begin(9600);

dht.begin();

}

void loop() {

delay(2000);

float temperature = dht.readTemperature();

float humidity = dht.readHumidity();

if (isnan(temperature) || isnan(humidity)) {

Serial.println("Failed to read from DHT sensor!");

return;

}

// Print data to serial monitor

Serial.print("Temperature: ");

Serial.print(temperature);

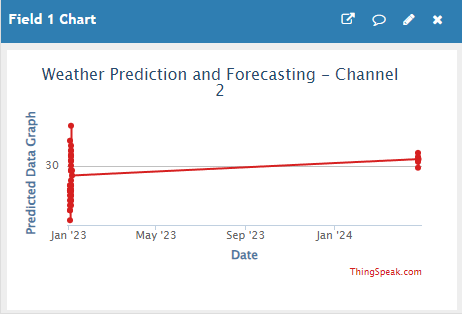
Serial.print("°C | Humidity: ");

Serial.print(humidity);

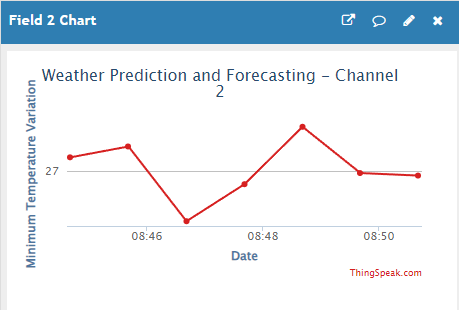
Serial.println("%");

}

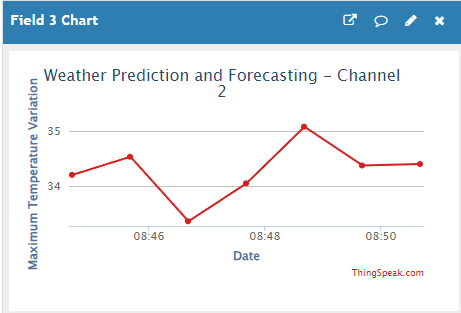
# Outputs



This graph shows the expected temperature values of present conditions based on the data from past



This graph shows the Maximum temperature for the day



This graph shows the Minimum temperature for the day

# Conclusion

* Through this machine learning SARIMA model, we were capable of predicting temperature values for an upcoming day.
* Through the data gathered using the DHT 11 Temperature sensor, we were able to predict weather for an upcoming day.