Weather Prediction Using Machine Learning

**SCHOOL OF COMPUTER APPLICATIONS AND TECHNOLOGY**

(Deemed to be University)

Submitted in partial fulfillment of the requirements for the award of

Master of computer applications

By

**DIYA KUMARI**

**24SCSE2030284**

**SWEETY KUMARI**

**24SCSE2030462**

**ANKITA RAJ**

**24SCSE2030096**

****

**SCHOOL OF COMPUTER APPLICATIONS AND TECHNOLOGY**

**GALGOTIAS UNIVERSITY, GREATER NOIDA, INDIA**

**SCHOOL OF COMPUTER APPLICATIONS AND TECHNOLOGY**

**GALGOTIAS UNIVERSITY, GREATER NOIDA, INDIA**

****

---------------------------------------------------------------------------------------------------

SCHOOL OF COMPUTER APPLICATIONS AND TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Professional Training Report is the bonafide work of DIYA KUMARI (24SCSE2030284) SWEETY KUMARI

(24SCSE2030462) ANKITA RAJ (24SCSE2030096) who underwent the professional training in “WEATHER PREDICTION” under our supervision from SEPT 2024 to FEB 2025.

**Internal Guide**

Mr.Srinivasan Raju

Assistant professor

**Head of the Department**

--------------------------------------------------------------------------------------------------------

Submitted for Viva voce Examination held on\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Internal Examiner External Examiner

Mr.Srinivasan Raju

**DECLARATION**

We, the undersigned, hereby declare that the Report on “WEATHER

PREDICTION USING MACHINE LEARNING” done by Diya Kumari(24SCSE2030284),Sweety Kumari(24SCSE2030462),Ankita Raj(24SCSE2030096)under the guidance of Mr.Srinivasan Raju(Assistant professor), at Galgotias University is submitted in partial fulfillment of the requirements for the award of Masters of Computer Applications.

DATE: 18/01/25 SIGNATURE OF THE CANDIDATE

PLACE: Greater Noida

**ACKNOWLEDGEMENT**

We, the undersigned, would like to express our heartfelt gratitude to everyone who contributed to the successful completion of our project titled **“WEATHER PREDICTION USING MACHINE LEARNING”.**

First and foremost, we extend our sincere thanks to Mr.Srinivasan Raju, whose invaluable guidance, encouragement, and expertise were instrumental in shaping this project. Their insights and suggestions enriched our understanding and helped us achieve our objectives.

We also wish to thank Galgotias University  for providing the resources and facilities that enabled us to conduct our research and complete this project effectively.

Finally, we acknowledge the teamwork, cooperation, and dedication of each member of our group—Diya Kumari, Sweety Kumari, and Ankita Raj without which this project would not have been possible.

Thank you all for being a part of this endeavor.

**Team Members:**

1. Diya Kumari
2. Sweety Kumari
3. Ankita Raj

Date: 18/1/25  
Place: Greater Noida

**ABSTRACT**

Weather prediction plays a critical role in numerous sectors, including agriculture, transportation, and disaster management. Traditional meteorological models have relied on numerical weather prediction (NWP) methods, which require extensive computational resources and a deep understanding of atmospheric dynamics. Despite the promise of ML in weather prediction, challenges such as data quality, model interpretability, and computational complexity remain. This paper also examines recent advances in ML techniques, such as neural networks and reinforcement learning, and their potential to further revolutionize weather prediction systems in the future.

**LIST OF TABLE**

|  |  |  |
| --- | --- | --- |
| **SNO** | **TOPICS** | **PAGE NO.** |
| **1** | **ABSTRACT** | **5** |
| **2** | **INTRODUCTION** | **7** |
| **3** | **FUTURE SCOPE** | **9** |
| **4** | **PROJECT DESCRIPTION** | **8** |
| **5** | **SYSTEM ANALYSIS & DESIGN** | **9** |
| **6** | **CONCLUSION** | **14** |
| **7** | **CODE** | **15** |
| **8** | **OUTPUT** | **16** |

**INTRODUCTION**

Weather prediction has always been a crucial task, impacting a wide range of sectors, including agriculture, aviation, transportation, disaster management, and daily life. Traditional weather forecasting methods have relied on numerical weather prediction (NWP), which involves complex mathematical models and simulations of atmospheric dynamics. These methods, though highly accurate, require significant computational power and are limited by the quality and resolution of available data. As a result, they often face challenges in accurately predicting short-term weather patterns or sudden changes in weather conditions.

**FUTURE SCOPE**

The future of weather prediction through machine learning (ML) holds tremendous promise, driven by continuous advancements in both data availability and computational power. As ML technologies mature and weather data becomes increasingly detailed and accessible, we can expect several transformative developments in the field. Advanced ML models, such as deep learning networks, are expected to contribute to more granular predictions, enhancing both short-term (e.g., next-hour or next-day) and long-term (e.g., seasonal or annual) forecasting.

**PROJECT DESCRIPTION**

The project titled **"Weather Prediction System Using Machine Learning"** is a Python-based application designed to provide real-time temperature predictions for Indian cities. This project leverages machine learning algorithms and current meteorological data to make accurate and user-friendly weather forecasts. The primary goal of this project is to create an efficient and reliable system where users can input the name of an Indian city and receive its current temperature and weather details. The system integrates the following components:

1. **Data Collection and Processing**:  
   The application uses publicly available weather datasets and APIs to gather real-time weather information. Data preprocessing techniques ensure the quality and accuracy of the information used for predictions.
2. **Machine Learning Model**:  
   A machine learning algorithm is trained on historical weather data to analyze trends and patterns, enabling the system to predict current and near-future temperatures. The model's performance is optimized for accuracy and speed.
3. **Python-Based Implementation**:  
   The entire system is implemented in Python, utilizing libraries such as Pandas for data manipulation, Scikit-learn for machine learning, and Matplotlib for visualization. The user interface is simple, allowing for seamless interaction.
4. **Real-Time Weather Retrieval**:  
   By integrating live data APIs (e.g., OpenWeatherMap), the system fetches up-to-date weather conditions for the requested city. This ensures that the results reflect the most current data available.

**Key Features**:

User inputs any Indian city name to get its current temperature.

Machine learning-based predictions enhance reliability and scalability.

Real-time data integration ensures up-to-date weather information.

Designed specifically to cater to Indian cities, providing localized predictions.

**Objective**:  
The project aims to combine machine learning techniques with real-time data to develop an intuitive and functional weather prediction system for Indian cities. It demonstrates the practical application of AI in meteorology while enhancing user accessibility to weather information.

**Applications**:  
This project can be used for educational purposes, research in weather prediction, or as a foundation for developing more advanced meteorological applications.

**SYSTEM ANALYSIS & DESIGN**

* **HARDWARE REQUIREMENTS**

|  |  |
| --- | --- |
| **NUMBER** | **DESCRIPTION** |
| 1 | PC With 256 GB or more hard disk |
| 2 | Pc with 4GB RAM |
| 3 | Pc with pentinum 1 |

* **SOFTWARE REQUIREMENTS**

|  |  |  |
| --- | --- | --- |
| **NUMBER** | **DESCRIPTION** | **TYPE** |
| 1 | OPERATING SYSTEM | WINDOWS HP |
| 2 | LANGUAGE | PYTHON |
| 3 | IDE | JUPYTER NOTEBOOK |
| 4 | BROWSER | GOOGLE CHROME |

**CONCLUSION**

Machine learning (ML) has emerged as a transformative tool in the field of weather prediction, offering significant improvements over traditional forecasting methods. By leveraging large datasets, such as historical weather records, real-time sensor data, and satellite imagery, ML models have demonstrated superior capabilities in identifying complex patterns, improving the accuracy of forecasts, and predicting extreme weather events with greater precision. Techniques such as deep learning, reinforcement learning, and hybrid models combining ML with traditional numerical weather prediction (NWP) systems have shown promising results in enhancing both short-term and long-term weather forecasting.

**CODE**

import requests

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_absolute\_error

# Your OpenWeatherMap API key

API\_KEY = "db837532278e7cd8b73cdec9918eed93"

# List of Indian cities

cities = ["Delhi", "Mumbai", "Kolkata", "Chennai", "Bangalore"]

# Function to fetch weather data for a city

def fetch\_weather\_data(city):

url = f"http://api.openweathermap.org/data/2.5/weather?q={city}&appid={API\_KEY}&units=metric"

response = requests.get(url)

if response.status\_code == 200:

data = response.json()

return {

"City": city,

"Temperature": data["main"]["temp"],

"Humidity": data["main"]["humidity"],

"WindSpeed": data["wind"]["speed"],

"Pressure": data["main"]["pressure"]

}

else:

print(f"Failed to fetch data for {city}")

return None

# Fetch data for all cities

weather\_data = [fetch\_weather\_data(city) for city in cities]

weather\_data = [data for data in weather\_data if data]

# Convert data to a DataFrame

df = pd.DataFrame(weather\_data)

# Features and target

X = df[["Humidity", "WindSpeed", "Pressure"]]

y = df["Temperature"]

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Train a Random Forest Regressor

model = RandomForestRegressor(n\_estimators=100, random\_state=42)

model.fit(X\_train, y\_train)

# Make predictions

y\_pred = model.predict(X\_test)

# Evaluate the model

mae = mean\_absolute\_error(y\_test, y\_pred)

print(f"Mean Absolute Error: {mae}")

# Predict temperature for a user-provided city

def predict\_temperature\_for\_city(city):

city\_data = fetch\_weather\_data(city)

if city\_data:

new\_data = pd.DataFrame({

"Humidity": [city\_data["Humidity"]],

"WindSpeed": [city\_data["WindSpeed"]],

"Pressure": [city\_data["Pressure"]]

})

predicted\_temp = model.predict(new\_data)[0]

print(f"Predicted Temperature for {city}: {predicted\_temp:.2f}°C")

else:

print(f"Could not fetch data for {city}.")

# Example usage

city\_to\_predict = input("Enter a city name: ")

predict\_temperature\_for\_city(city\_to\_predict)

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Weather Prediction</title>

<link rel="stylesheet" href="/static/styles.css">

</head>

<body>

<div class="container">

<h1>Weather Prediction for Indian Cities</h1>

<form action="/" method="POST">

<label for="city">Enter City Name:</label>

<input type="text" id="city" name="city" placeholder="e.g., Delhi" required>

<button type="submit">Predict Temperature</button>

</form>

{% if predicted\_temp %}

<div class="result">

<h3>Predicted Temperature for {{ city }}:</h3>

<p>{{ predicted\_temp }}°C</p>

</div>

{% elif error %}

<div class="error">

<p>{{ error }}</p>

</div>

{% endif %}

</div>

</body>

</html>

body {

font-family: Arial, sans-serif;

background-color: #f0f8ff;

margin: 0;

padding: 0;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

}

.container {

background: #fff;

padding: 20px;

border-radius: 10px;

box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

text-align: center;

max-width: 400px;

width: 100%;

}

h1 {

margin-bottom: 20px;

font-size: 24px;

}

form {

margin-bottom: 20px;

}

label {

display: block;

margin-bottom: 10px;

}

input {

width: calc(100% - 20px);

padding: 10px;

margin-bottom: 10px;

border: 1px solid #ccc;

border-radius: 5px;

}

button {

background-color: #007BFF;

color: #fff;

border: none;

padding: 10px 20px;

border-radius: 5px;

cursor: pointer;

}

button:hover {

background-color: #0056b3;

}

.result, .error {

margin-top: 20px;

text-align: left;

}

.result p {

font-size: 20px;

font-weight: bold;

}

.error p {

color: red;

font-weight: bold;

}

**OUTPUT**

