

Faculty of Informatics and Control Systems

Machine Learning

Course work/project topic : **Forcasting The Future**

A Machine Learning Approach To Weather Prediction

III Course, group --108258---

**Student's name and surname:** Sandro Makhatadze, Rati Nabijashvili

**Lecturer's name and surname:** Teona Gogidze

Tbilisi, 0160, Georgia

2025

CONTENT

[Introduction 2](#_Toc126340285)

[1. From Ancient Sky Watchers To Machine Learning Models 3](#_Toc126340287),4

[1.2.Vision and Mission Statement](#_Toc126340288) 4

[1.3. Data Source 4](#_Toc126340289)

[1.4. Data Preprocessing 5](#_Toc126340290)

[1.5. Choosing The Right Algorithm 6](#_Toc126340290)

[1.6. Training And Testing The Model 6](#_Toc126340290)

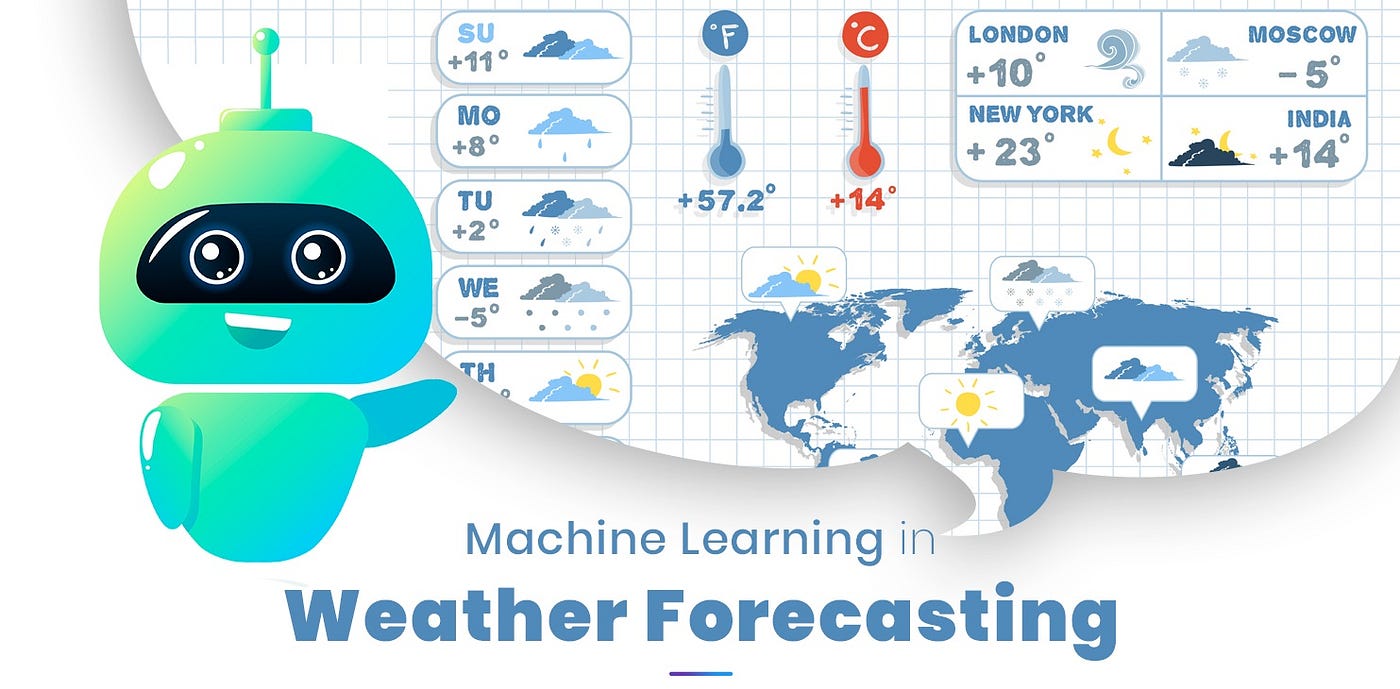
[1.7. Results 7](#_Toc126340290)

[Conclusion 7](#_Toc126340291)

[Literature 8](#_Toc126340292)

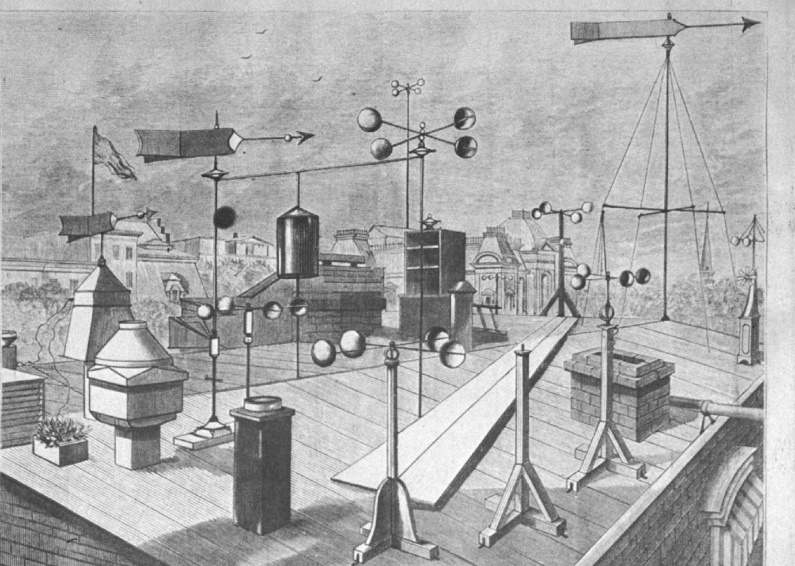
**Introduction**

Weather prediction is the process of using data and models to guess what the weather will be like in the near future. It plays a very important role in our daily lives — helping people decide what to wear, when to travel, or how to protect crops,homes, and health.

**Pic. 1**

**From Ancient Sky Watchers To Machine Learning Models**

It has always been important in human life. Hundreds of years ago, people used to predict the weather by looking at the sky,feeling the wind, or watching the behavior of animals. Farmers would look at cloud shapes or sunrise colors to guess if rain was coming. These methods were based on experience and tradition, but they were not always correct.Later, scientists started to use tools like barometers and thermometers to measure weather conditions.With the invention of computers, weather forecasting became more accurate, using complex equations to simulate the atmosphere.



**Pic. 2**

Today, we are in a new era of forecasting. Thanks to machine learning and big data, we can train models on huge amounts of weather information to find patterns and make better predictions. Machine learning gives us a faster and sometimes smarter way to understand what the weather might do next. In this project, I explore how machine learning can be used to predict weather conditions, using real historical data to train and test a model. This work shows how modern technology is helping improve something as old — and as important — as the weather forecast.

**Pic. 3**

**Vision and Mission Statement**

The main goal of this project is to build a machine learning model that can predict certain weather conditions, like temperature or rainfall,based on past data.

do this, we use historical weather data to train the model and help it learn patterns that can be used to forecast future weather.

The bigger goal is to show how machine learning can be applied to real-world problems, like weather prediction,and how it can lead to better, faster, and smarter forecasts than traditional methods.

**Data Source**

For this project, we used a historical weather dataset collected from OpenWeatherMap. The dataset includes several key features that are important for weather forecasting — such as temperature, humidity, air pressure, wind speed, and weather conditions.

The data is organized by date and includes daily or hourly observations, depending on the source. This kind of structured, time-based data is very useful for training machine learning models, especially for time series prediction.

Before using it, we also checked for missing values and made sure the data was clean enough to use in a model.

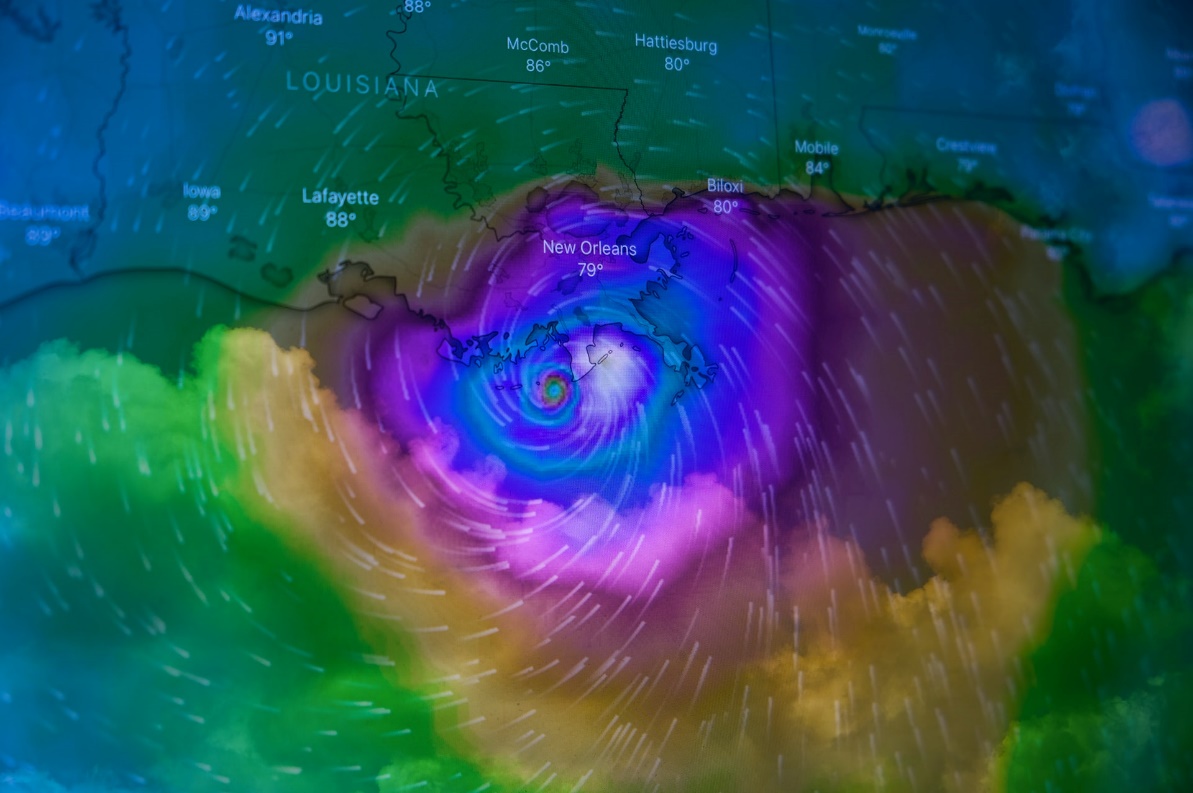
**Data Preprocessing**

The first step in preprocessing was handling missing values. We either removed rows with missing data or filled them using averages, depending on the case.

Next, we extracted new features from the date, such as the day of the month, the month itself, and the season. These can help the model better understand patterns.

Then we encoded any categorical features, such as “weather condition,” turning them into numeric values so the model could process them.

Finally, we normalized the numerical data — like temperature and humidity — to keep all values on a similar scale. This helps the model train more efficiently and prevents bias toward larger numbers.

**Pic. 4**

**Choosing The Right Algorithm**

In this project, we used the Random Forest algorithm for both parts of my prediction task. For classification, we used Random Forest Classifier to predict whether it will rain tomorrow based on features like temperature, humidity, pressure, and wind direction.

For regression, we used Random Forest Regressor to forecast future temperature and humidity values. This model learns from past values and predicts the next few hours ahead.

We chose Random Forest because it performs well with real-world, slightly messy data, and it works for both classification and regression problems. It combines the power of many decision trees, which helps reduce overfitting and improves prediction accuracy.

**Training And Testing The Model**

After preparing the data, we split it into two parts: 80% for training and 20% for testing. We trained the models on historical weather data, so they could learn patterns based on temperature, humidity, wind speed, and more.

Then we tested the models on unseen data to see how well they could predict — both for rain classification and for temperature/humidity regression.

To evaluate how good the predictions were, we used Mean Squared Error (MSE), which helps measure the difference between the predicted and actual values. The lower the error, the better the model performs.

**Pic. 5**

**Results**

The model’s Mean Squared Error for rain prediction was 0.16, which shows it performs well on test data.

For a real example, we tested the model with current weather data from Tbilisi, and it predicted "Yes" for rain. The live weather description was “broken clouds”, which often appears before rain, so the model prediction was reasonable.

The temperature forecast for the next 5 hours showed a natural decrease — from 20.2°C to 15.4°C — as expected during nighttime. The humidity levels are predicted to rise gradually, from 39.7% up to 58.1%, which often happens at night when the air cools and holds more moisture.

Overall, the model gave a realistic and useful short-term forecast for Tbilisi, showing that machine learning can be applied effectively in local weather prediction.

**Conclusion**

In this project, we built a weather prediction system using machine learning and real-world weather data.

We used two types of models: a Random Forest Classifier to predict rain, and Random Forest Regressors to forecast temperature and humidity. Both models performed well and gave accurate, realistic results when tested on Tbilisi’s current data.

This project helped me better understand how machine learning works — from cleaning data and choosing the right features to training, testing, and interpreting model results.

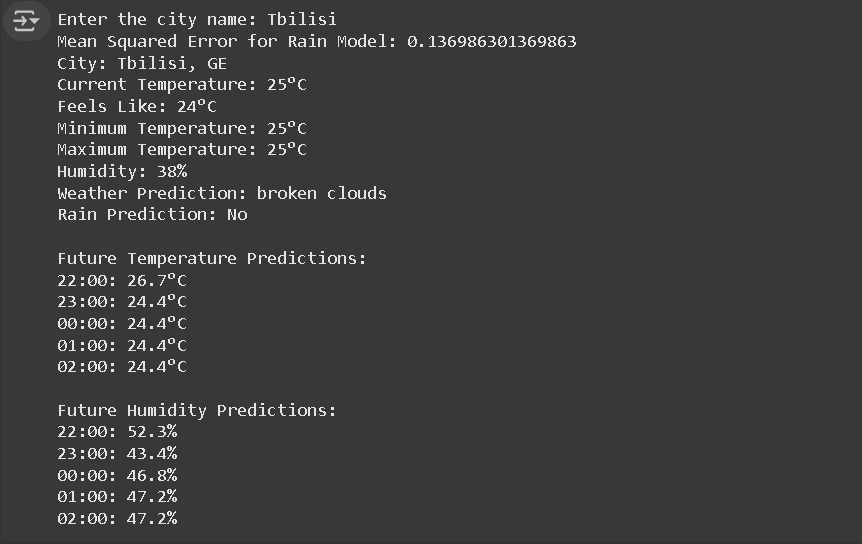
It also showed how machine learning can help solve real-world problems and improve everyday tools like weather forecasts, which affect millions of people.

**Pic. 6**

**Task**

*Weather Prediction With Machine Learning*

**Visualization of the results**

****

**Literature**

1. ABC Radio National – How the history of weather forecasting went from reading clouds to supercomputers.
2. Francois Chollet, Deep learning with Python, Manning Publications, 2020
3. P.Deitel, H.Deitel, intro to Python for Computer Science and Data Science, 2020
4. Richardson’s Fantastic Forecast Factory – By Peter Lynch, School of Mathematics, University College Dublin.
5. Rain Viewer – The History of Weather Forecasting.

**Code Link:** <https://colab.research.google.com/drive/1GUhgZ811tEV4lsZWaByJzj38QoggKlDq?usp=sharing>

**Code**

import requests

import pandas as pd

import numpy as np

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

from sklearn.preprocessing import LabelEncoder

from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor

from sklearn.metrics import mean\_squared\_error

from datetime import datetime, timedelta

import pytz

API\_KEY = 'b0b4d35affcee15a65066f7cf19868c8'

BASE\_URL = 'https://api.openweathermap.org/data/2.5/'

def get\_current\_weather(city):

    url = f"{BASE\_URL}weather?q={city}&appid={API\_KEY}&units=metric"

    response = requests.get(url)

    if response.status\_code != 200:

        raise Exception(f"Error fetching data: {response.status\_code} - {response.text}")

    data = response.json()

    return {

        'city': data['name'],

        'current\_temp': round(data['main']['temp']),

        'feels\_like': round(data['main']['feels\_like']),

        'temp\_min': round(data['main']['temp\_min']),

        'temp\_max': round(data['main']['temp\_max']),

        'humidity': round(data['main']['humidity']),

        'description': data['weather'][0]['description'],

        'country': data['sys']['country'],

        'wind\_gust\_dir': data['wind']['deg'],

        'pressure': data['main']['pressure'],

        'Wind\_Gust\_Speed': data['wind']['speed']

    }

def read\_historical\_data(filename):

    df = pd.read\_csv(filename)

    df = df.dropna()

    df = df.drop\_duplicates()

    return df

def prepare\_data(data):

    le = LabelEncoder()

    data["WindGustDir"] = le.fit\_transform(data['WindGustDir'])

    data["RainTomorrow"] = le.fit\_transform(data['RainTomorrow'])

    X = data[['MinTemp', 'MaxTemp', 'WindGustDir', 'WindGustSpeed', 'Humidity', 'Pressure', 'Temp']]

    y = data['RainTomorrow']

    return X, y, le

def train\_rain\_model(X, y):

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

    model = RandomForestClassifier()

    model.fit(X\_train, y\_train)

    y\_pred = model.predict(X\_test)

    print("Mean Squared Error for Rain Model:", mean\_squared\_error(y\_test, y\_pred))

    return model

def prepare\_regression\_data(data, feature):

    X, y = [], []

    for i in range(len(data) - 1):

        X.append(data[feature].iloc[i])

        y.append(data[feature].iloc[i + 1])

    X = np.array(X).reshape(-1, 1)

    y = np.array(y)

    return X, y

def train\_regression\_model(X, y):

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

    model.fit(X, y)

    return model

def predict\_future(model, current\_value):

    predictions = [current\_value]

    for i in range(5):

        next\_value = model.predict(np.array([[predictions[-1]]]))

        predictions.append(next\_value[0])

    return predictions[1:]

def weather\_view():

    city = input("Enter the city name: ")

    current\_weather = get\_current\_weather(city)

    # Load Historical Data

    historical\_data = read\_historical\_data("/content/weather.csv")

    # Prepare and Train The Rain Prediction model

    X, y, le = prepare\_data(historical\_data)

    rain\_model = train\_rain\_model(X, y)

    # Map Wind Direction to Compass Point

    wind\_deg = current\_weather['wind\_gust\_dir'] % 360

    compass\_points = [

        ("N", 0, 11.25), ("NNE", 11.25, 33.75), ("NE", 33.75, 56.25),

        ("ENE", 56.25, 78.75), ("E", 78.75, 101.25), ("ESE", 101.25, 123.75),

        ("SE", 123.75, 146.25), ("SSE", 146.25, 168.75), ("S", 168.75, 191.25),

        ("SSW", 191.25, 213.75), ("SW", 213.75, 236.25), ("WSW", 236.25, 258.75),

        ("W", 258.75, 281.25), ("WNW", 281.25, 303.75), ("NW", 303.75, 326.25),

        ("NNW", 326.25, 348.75)

    ]

    compass\_direction = next(point for point, start, end in compass\_points if start <= wind\_deg < end)

    compass\_direction\_encoded = le.transform([compass\_direction])[0] if compass\_direction in le.classes\_ else -1

    current\_data = {

        'MinTemp': current\_weather['temp\_min'],

        'MaxTemp': current\_weather['temp\_max'],

        'WindGustDir': compass\_direction\_encoded,

        'WindGustSpeed': current\_weather['Wind\_Gust\_Speed'],

        'Humidity': current\_weather['humidity'],

        'Pressure': current\_weather['pressure'],

        'Temp': current\_weather['current\_temp'],

    }

    current\_df = pd.DataFrame([current\_data])

    # Rain Prediction

    rain\_prediction = rain\_model.predict(current\_df)[0]

    # Prepare Regression Model for Temperature and Humidity

    X\_temp, y\_temp = prepare\_regression\_data(historical\_data, 'Temp')

    X\_hum, y\_hum = prepare\_regression\_data(historical\_data, 'Humidity')

    temp\_model = train\_regression\_model(X\_temp, y\_temp)

    hum\_model = train\_regression\_model(X\_hum, y\_hum)

    # Predict Future Temperature and Humidity

    future\_temp = predict\_future(temp\_model, current\_weather['temp\_min'])

    future\_humidity = predict\_future(hum\_model, current\_weather['humidity'])

    # Prepare Time for Future Predictions

    timezone = pytz.timezone('Asia/Karachi')

    now = datetime.now(timezone)

    next\_hour = now + timedelta(hours=1)

    next\_hour = next\_hour.replace(minute=0, second=0, microsecond=0)

    future\_times = [(next\_hour + timedelta(hours=i)).strftime("%H:00") for i in range(5)]

    # Display Results

    print(f"City: {city}, {current\_weather['country']}")

    print(f"Current Temperature: {current\_weather['current\_temp']}°C")

    print(f"Feels Like: {current\_weather['feels\_like']}°C")

    print(f"Minimum Temperature: {current\_weather['temp\_min']}°C")

    print(f"Maximum Temperature: {current\_weather['temp\_max']}°C")

    print(f"Humidity: {current\_weather['humidity']}%")

    print(f"Weather Prediction: {current\_weather['description']}")

    print(f"Rain Prediction: {'Yes' if rain\_prediction else 'No'}")

    print("\nFuture Temperature Predictions:")

    for time, temp in zip(future\_times, future\_temp):

        print(f"{time}: {round(temp, 1)}°C")

    print("\nFuture Humidity Predictions:")

    for time, humidity in zip(future\_times, future\_humidity):

        print(f"{time}: {round(humidity, 1)}%")

weather\_view()