Zubair Ahmed
704971

IoT Assignment 2 Report

1. Machine Learning Algorithm Choice

Algorithm Used:

Support Vector Regression (SVR) using PHP-ML's integration with libsvm.

Justification:

Support Vector Regression is a strong technique that works well with continuous data, particularly when there is a non-linear connection between input and output. It works well with datasets that are small to medium in size, which fits it nicely with our project's scope. Our objective is to anticipate temperature and humidity levels for a specific time of day using our Internet of Things-based weather forecasting algorithm. These factors, which are impacted by daily weather cycles, naturally fluctuate in a smooth pattern throughout the day.

Because SVR can generalize effectively from sparse and noisy datasets, it is a good fit for this task. Because of its kernel-based methodology, it can handle both linear and nonlinear patterns in the data, guaranteeing precise predictions even in cases when the data is faulty or impacted by outside noise.

Training Data Requirements

Per Location:

Input (X): Time of day, represented as minutes since midnight. For example: 00:00 = 0, 00:30 = 30

Output (y): Corresponding measurements of humidity (in %RH) and temperature (in °C) made at that same moment.

Based on past trends, this mapping enables the model to understand how temperature and humidity normally fluctuate during the day.

Data Sampling Steps

We employed a meticulous data pretreatment strategy to get the training dataset ready:

CSV Parsing: We processed the given CSV files, which each included historical weather data that had been collected over a number of years at different intervals.

Date Matching: Records were only chosen if they were from the same day and month as the target forecast date. For instance, only data from prior years on January 1st were considered if the prediction was for that day.

Year-wise Selection: Rows matching to the chosen date were retrieved for every year that was available. This increases forecast reliability and guarantees consistency in seasonal circumstances.

Data Cleaning: In order to avoid distortions during training, missing values were eliminated. Statistical criteria were used to remove outliers and corrupted records (for example, deleting temperature values below -50 or over 60°C). Where required, data was smoothed to account for minute noise or discrepancies between sensors.

2. Accuracy and Enhancement Conversation

Accuracy Findings

Even though the model is rather simple, the outcomes are very useful, especially for short-term weather forecasting. The overall trend of temperature and humidity over the course of a day may be successfully identified using SVR. It makes daily minimum and maximum value predictions by using recurrent seasonal trends seen in previous years. This makes it helpful in situations where high-level environmental trends are enough, such intelligent Internet of Things systems that keep an eye on circumstances and react by providing simple functional or visual feedback.

The model's ability to accurately capture seasonal variations typical of weather data makes its results typically dependable on days with a wealth of previous data. Another factor that supports the model's steady forecasting approach is the lack of abrupt shifts, such as extreme weather occurrences.

Restrictions and Limitations

However, a number of restrictions limit this approach's precision and scalability:

Lack of Contextual Features

Significant external environmental factors, including cloud cover, precipitation, air pressure, and solar radiation, are not taken into account by the model. Both temperature and humidity are frequently significantly influenced by these elements.

Limited or Incomplete Training Information

In certain instances, low historical samples may result in less reliable projections, especially for uncommon dates or recently established locales. Accurately learning seasonal patterns is more difficult for SVR when the sample is smaller.

Fixed Kernel Selection

SVR may have trouble fitting non-linear connections found in meteorological data if it is implemented using a linear kernel, which is PHP-ML's default. The model's versatility is limited since temperature and humidity don't always follow straightforward patterns, particularly when there are abrupt changes.

Absence of a Real-Time Feedback Loop

As fresh data becomes available, the model does not modify predictions based on real-time sensor readings, which might improve accuracy.

Methods for Improving Accuracy

Several tactics may be used to enhance this weather forecasting model's functionality and resilience:

Different Variables Incorporated:

Add more meteorological components to the training dataset, such as:

- The humidity and temperature from the day before
- Pressure in the atmosphere
- Wind direction and speed
- Cloud cover and dew point are two characteristics that can help the model better represent intricate weather dynamics.

Normalization of Input

SVR's learning efficiency can be improved, particularly when using non-linear kernels, by scaling input variables (such as time, temperature, and humidity) to a constant range.

Kernel Tuning or Switching

The model could be able to better suit the non-linear correlations in weather patterns if it were to switch to more expressive kernels, such polynomial kernels or RBF (Radial Basis Function).

Migration to Python or ML Backend

By switching from PHP-ML to a more complex machine learning environment (such as TensorFlow or Python with Scikit-learn), one may access complex models such as: Random Forest, Gradient Boosting Machines, LSTM (Long Short-Term Memory).

Expanding Historical Data

By expanding the historical data set (more years, more reliable samples), the model has a larger training set, which enhances generalization and enables it to identify uncommon patterns.

Semi-Real-Time Updates

Predictions can be constantly adjusted or improved by using real-time data streams from sensors. IoT devices responding to abrupt changes in the environment would particularly benefit from this.

Cross-Validation and Error Logging

Model performance may be verified using k-fold cross-validation, and logging methods that monitor prediction errors over time can be put in place. This encourages ongoing development and assists in determining the circumstances that result in the most errors.