**WEATHER CAST : A MACHINE LEARNING BASED**

**WEATHER PREDICTION**

**A Project Work Report**

**BACHELOR OF TECHNOLOGY**

**in**

**ARTIFICIAL INTELLIGANCE AND DATA SCIENCE**

## Submittedby

**GIDUTHURI VIJAYA DURGA 23B91A5452**

**GIDUTHURI REKHA SRI 23B91A5451**

**DASARI DEVI SANTOSHINI 23B91A5442**

**BUDDHARAJU VARSHINI 23B91A5427**

**Undertheesteemedguidance of**

## V.SAIBABA

**Assistant Professor**

****

**DEPARTMENTOFINFORMATIONTECHNOLOGY**

**SAGI RAMAKRISHNAM RAJUENGINEERINGCOLLEGE(AUTONOMOUS)**

(ApprovedbyAICTE,NewDelhi,AffiliatedtoJNTU University,Kakinada)

##### CHINNAAMIRAM::BHIMAVARAM-534202

###### April-2025

SAGI RAMAKRISHNAM RAJUENGINEERINGCOLLEGE

(AUTONOMOUS)

**(ApprovedbyAICTE,NewDelhi, AffiliatedtoJNTUUNIVERSITY,KAKINADA}**

**CHINNAAMIRAM,BHIMAVARAM-534204**

**DEPARTMENTOF ARTIFICIAL INTELLIGANCE AND DATA SCIENCE**

****

Certificate

**This is to certify that the Project Work report entitled "WEATHER CAST : A MACHINE LEARNING BASED WEATHER PREDICTION ”, is Bonafide work submitted by GIDUTHURI VIJAYA DURGA (Regd.No:23B91A5452), GIDUTHURI REKHA SRI (Regd.No:23B91A5451), DASARI DEVI SANTOSHINI (Regd.No:23B91A5442), BUDDHARAJU VARSHINI (Regd.No:23B91A5427), in Information Technology during the Academic year 2024-2025.**

###### HOD GUIDE

Dr. RAVI KIRAN VARMA P. V SAIBABA

Assistant Professor

HOD-IT DeptAssociateProfessor

**ABSTRACT**

Accurate weather forecasting is a cornerstone of modern society, influencing sectors ranging from agriculture and transportation to disaster preparedness and urban planning. Traditional forecasting methods, though advanced, often rely on computationally intensive numerical simulations that struggle to adapt swiftly to dynamic atmospheric changes. This project addresses these limitations by leveraging machine learning (ML) to predict critical weather parameters—temperature, humidity, and rainfall—using historical meteorological data from the National Oceanic and Atmospheric Administration (NOAA). We implement and compare four supervised ML models: Linear Regression (LR), Random Forest Regression (RFR), Support Vector Machine (SVM), and K-Nearest Neighbors (KNN). Through rigorous evaluation using metrics such as Mean Squared Error (MSE), Mean Absolute Error (MAE), and R-squared (R²), we demonstrate that Linear Regression outperforms other models, achieving an exceptional R² score of 0.99 with minimal error rates (MAE: 2.35, MSE: 1.16). These results underscore the potential of ML to revolutionize weather forecasting by offering a scalable, efficient, and adaptive alternative to traditional methods.

**INTRODUCTION**

The ability to predict weather conditions with precision has far-reaching implications for both everyday life and specialized industries. Historically, weather forecasting has depended on complex physical models that simulate atmospheric processes using mathematical equations. While these models are powerful, they require immense computational resources and often fail to capture sudden meteorological shifts. The advent of machine learning has introduced a paradigm shift, enabling data-driven approaches that learn patterns directly from historical weather data without explicit physical modeling. This project explores the application of ML techniques to short-term weather prediction, focusing on temperature, humidity, and rainfall—key variables that impact agricultural productivity, energy consumption, and public safety. By harnessing datasets from NOAA, which include features such as station location, date, elevation, and temperature extremes (TAVG, TMAX, TMIN), we train and evaluate multiple regression models. Our goal is to identify the most robust algorithm for weather prediction while highlighting the transformative potential of ML in meteorology.

**PROBLEM DEFINITION**

Weather prediction is inherently challenging due to the chaotic nature of atmospheric systems, where small changes in initial conditions can lead to vastly different outcomes. Traditional methods, such as numerical weather prediction (NWP) models, rely on solving differential equations that describe fluid dynamics and thermodynamics. However, these models are computationally expensive and often lack the agility to incorporate real-time data updates. Machine learning presents an opportunity to bypass these limitations by learning directly from historical observations. The core problem addressed in this project is the accurate prediction of short-term weather variables—specifically, daily average temperature (TAVG) and rainfall—using supervised learning algorithms. The challenges include handling noisy or incomplete data, selecting relevant features, and ensuring model generalizability across diverse geographic regions. By framing weather prediction as a regression task, we aim to develop a system that not only matches the accuracy of traditional methods but also offers faster predictions with lower computational overhead.

**EXISTING SYSTEM**

Weather forecasting has long relied on two primary approaches: **physics-based numerical models** and **statistical methods**. While these systems have evolved over decades, they face significant challenges in accuracy, computational demands, and adaptability to changing climate patterns.

**Physics-Based Numerical Models**

The backbone of modern weather forecasting, numerical models like the **Global Forecast System (GFS)** and the **European Centre for Medium-Range Weather Forecasts (ECMWF)**, solve complex mathematical equations that simulate atmospheric physics. These models divide the atmosphere into grids and use supercomputers to calculate future states based on current conditions.

**Strengths**:

* High accuracy for large-scale weather systems (e.g., hurricanes, jet streams).
* Effective for medium-to-long-range forecasts (3–10 days).

**Limitations**:

* **Computationally expensive**: Require supercomputers and massive energy resources.
* **Resolution constraints**: Struggle with hyper-local predictions (e.g., sudden rainfall in small regions).
* **Initialization errors**: Small inaccuracies in input data (e.g., temperature, pressure) amplify over time, reducing reliability.

**Statistical Methods**

Traditional statistical techniques, such as **Autoregressive Integrated Moving Average (ARIMA)** and **Exponential Smoothing**, analyze historical weather data to identify patterns. These methods assume that future weather depends linearly on past observations.

**Strengths**:

* Simple and fast for short-term predictions (e.g., next-day temperature).
* Useful for stable climates with repetitive patterns.

**Limitations**:

* **Linear assumptions**: Fail to capture non-linear atmospheric interactions (e.g., sudden storms).
* **No contextual awareness**: Ignore spatial relationships (e.g., elevation effects on rainfall).

**Machine Learning in Weather Forecasting**

Recent advancements leverage **machine learning (ML)** to address gaps in traditional systems. ML models learn directly from historical data without explicitly programming physical laws.

**Common Approaches**:

1. **Supervised Learning**:
   * **Linear Regression**: Predicts temperature/humidity but oversimplifies complex systems.
   * **Random Forests**: Handles non-linear relationships (e.g., humidity vs. rainfall) but risks overfitting.
   * **Support Vector Machines (SVM)**: Useful for classification (e.g., rain/no rain) but less effective for precise numeric predictions.
2. **Deep Learning**:
   * **LSTMs/RNNs**: Capture temporal patterns in weather sequences (e.g., temperature trends over weeks).
   * **Convolutional Neural Networks (CNNs)**: Analyze spatial data (e.g., satellite images) for regional forecasts.

**Challenges**:

* **Data quality**: Missing or noisy sensor data (e.g., faulty weather stations) degrades performance.
* **Interpretability**: "Black-box" models like neural networks lack transparency for meteorologists.
* **Generalization**: Models trained on one region may fail elsewhere due to climate variability.

**Hybrid Systems**

Some modern frameworks combine **numerical models with ML** (e.g., using ML to correct biases in physics-based outputs). For example:

* **Post-processing ML**: Refines raw numerical forecasts (e.g., adjusting rainfall probability).
* **Feature augmentation**: Enhances traditional models with ML-derived inputs (e.g., soil moisture for flood prediction).

**Key Limitations of Current Systems**:

* **High computational costs** for real-time updates.
* **Dependency on historical data**, limiting adaptability to climate change.
* **Scalability issues** in data-sparse regions (e.g., oceans, deserts).

### ****Summary of the Existing System****

Traditional weather forecasting systems combine **physics-based numerical models** and **statistical methods**, each with critical limitations. Numerical models (e.g., GFS, ECMWF) provide large-scale accuracy but demand supercomputers and struggle with local predictions. Statistical approaches (e.g., ARIMA) are lightweight yet fail to capture non-linear atmospheric dynamics. Recent machine learning applications show promise but face challenges like data quality issues, overfitting, and poor interpretability. Hybrid systems that blend physics with ML offer partial solutions but remain computationally intensive and region-specific. These gaps highlight the need for adaptive, scalable ML solutions like the one proposed in this project, which prioritizes accuracy, efficiency, and generalizability for short-term weather prediction.

**PROPOSED SYSTEM**

The proposed system introduces a machine learning-based framework for accurate weather prediction, designed to overcome the limitations of traditional forecasting methods. The system focuses on predicting temperature, rainfall, and humidity using historical weather data from NOAA.

**Key Components**

**1. Data Collection and Preparation**

* Sources weather data from NOAA's historical datasets
* Collects essential parameters: temperature (TAVG, TMAX, TMIN), precipitation, humidity, wind speed
* Includes geographic features: latitude, longitude, elevation

**2. Data Preprocessing Pipeline**

* Handles missing values through interpolation
* Normalizes numerical features for consistent scaling
* Encodes categorical variables like station IDs
* Creates time-based features (e.g., 7-day moving averages)

**3. Feature Engineering**

* Generates derived meteorological indicators:
  + Heat index calculations
  + Dew point estimations
  + Temperature change rates
* Incorporates spatial relationships:
  + Elevation-adjusted temperature trends
  + Regional climate patterns

**4.MachineLearning Implementation**  
Four supervised learning models are implemented and compared:

**• Linear Regression**

* Serves as baseline model
* Provides interpretable results
* Fast training and prediction

**• Random Forest Regression**

* Handles non-linear relationships
* Reduces overfitting through ensemble learning
* Provides feature importance metrics

**• Support Vector Regression**

* Effective for high-dimensional data
* Uses epsilon-insensitive loss function
* Kernel trick captures complex patterns

**• K-Nearest Neighbors**

* Instance-based learning
* Adapts to local weather patterns
* Simple but computationally intensive

**5. Evaluation Framework**

* Employs standard regression metrics:
  + Mean Absolute Error (MAE)
  + Mean Squared Error (MSE)
  + R-squared (R²) score
* Uses time-series cross-validation
* Generates comparative performance reports

**6. Prediction and Visualization**

* Produces daily forecasts for target parameters
* Generates visual comparisons:
  + Actual vs predicted line charts
  + Residual plots
  + Error distribution graphs

**System Advantages**

**• Improved Accuracy**

* Combines multiple weather indicators
* Captures complex atmospheric relationships

**• Computational Efficiency**

* Runs on standard hardware
* Faster than numerical weather models
* Scalable for multiple locations

**• Practical Interpretability**

* Clear feature importance rankings
* Transparent model decisions
* Explainable predictions

**• Adaptability**

* Continuously learns from new data
* Adjusts to climate changes
* Customizable for specific regions

The proposed system represents a significant advancement in weather forecasting technology, combining meteorological expertise with cutting-edge machine learning techniques to deliver reliable, efficient, and interpretable predictions.

**SYSTEM REQUIREMENTS**

**Functional Requirements:**

* **Data Collection:** Uses the **NOAA API** to fetch historical weather data, including temperature (TAVG, TMAX, TMIN), humidity, rainfall, and wind speed metrics.
* **Feature Engineering:** Computes key meteorological indicators such as **Moving Averages (MA) of temperature, Heat Index, Dew Point, and Precipitation Trends** to enhance the dataset.
* **Prediction Models:** Implements supervised machine learning models like **Linear Regression, Random Forest, XGBoost, and Support Vector Regression (SVR)** for next-day weather parameter prediction.
* **Evaluation & Visualization:** Compares predicted vs. actual weather data using performance metrics (**MAE, MSE, R² Score**) and generates line charts, heatmaps, and error distribution plots for analysis.
* **User Interaction:** Allows users to select a **geographic location, date range, and weather parameter** (e.g., temperature, rainfall) and view results via a **Jupyter Notebook or web dashboard**.

**Hardware Requirements:**

* **Model Training & Evaluation:** The system runs efficiently on standard computing hardware. Recommended configuration:
  + **Processor:** Intel Core i5 or AMD Ryzen 5
  + **RAM:** 8 GB or higher
  + **Storage:** 200 GB available disk space (for large weather datasets)
  + **GPU (Optional):** NVIDIA GTX 1650 or higher for accelerated training (not mandatory for basic models).

**Software Requirements:**

* **Programming Language:** Python 3.8+
* **Development Environment / IDE:** Jupyter Notebook / Visual Studio Code
* **Frameworks & Libraries:**
  + **Data Handling:** pandas, numpy
  + **Visualization:** matplotlib, seaborn, plotly
  + **Machine Learning:** scikit-learn, xgboost
  + **API Integration:** requests, NOAA’s API for weather data retrieval
  + **Geospatial Analysis (Optional):** geopandas, rasterio

**System Design**

**1. Modules:**

* **Data Ingestion:** Fetches weather data from NOAA based on user input (location, date range).
* **Feature Engineering:** Computes weather indicators (e.g., 7-day rolling temperature averages) and prepares data for training.
* **Modeling & Prediction:** Trains regression models and predicts next-day weather conditions.
* **Evaluation:** Calculates error metrics (MAE, MSE) and visualizes model performance.
* **Result Display:** Shows forecasts via graphs (actual vs. predicted) and exportable reports.

**2. Workflow:**

1. User specifies a **location and time period**.
2. System retrieves historical weather data from **NOAA**.
3. **Feature engineering** adds derived weather metrics.
4. Models are trained and generate **next-day forecasts**.
5. Results are displayed with **metrics and visualizations**.

**Technology Stack:**

* **Backend:** Python (data processing, modeling, API calls)
* **Machine Learning Framework:** Scikit-learn
* **Data Source:** NOAA API
* **Visualization Tools:** Matplotlib, Seaborn
* **Web Interface (Optional):** Flask for interactive dashboards

**SYSTEM ARCHITECTURE**

The system architecture for the **weather prediction project** is designed as a **modular, layered pipeline** that transforms raw meteorological data into accurate weather forecasts using supervised machine learning. Each component is optimized for efficient data flow, preprocessing, model training, and visualization.

**Architecture Overview**

The system consists of **five core layers**:

1. **Data Collection Layer**
2. **Data Preprocessing & Feature Engineering Layer**
3. **Model Training & Selection Layer**
4. **Prediction & Evaluation Layer**
5. **Visualization & Reporting Layer**

**Component Description**

**1. Data Collection Layer**

* **Source:** NOAA API (National Oceanic and Atmospheric Administration)
* **Function:**
  + Fetches historical weather data, including:
    - Temperature (TAVG, TMAX, TMIN)
    - Precipitation (Rainfall)
    - Humidity, Wind Speed
    - Geographic data (Latitude, Longitude, Elevation)
  + Output: Raw dataset in CSV/DataFrame format.

**2. Data Preprocessing & Feature Engineering Layer**

* **Tasks:**
  + Handle missing values (e.g., impute gaps in temperature records).
  + Normalize numerical features (Min-Max Scaling).
  + Encode categorical variables (e.g., Station IDs).
  + Generate **weather-specific features**:
    - 7-day moving averages for temperature
    - Heat Index & Dew Point calculations
    - Elevation-adjusted climate trends
* **Output:** Cleaned, feature-enhanced dataset for modeling.

**3. Model Training & Selection Layer**

* **Algorithms Used:**
  + **Linear Regression** (Baseline for interpretability)
  + **Random Forest Regressor** (Handles non-linear weather patterns)
  + **XGBoost Regressor** (Optimized for performance)
  + **Support Vector Regressor (SVR)** (For high-dimensional data)
* **Tasks:**
  + Split data into **training (80%)** and **testing (20%)** sets.
  + Train models using time-series cross-validation.
  + Select best model based on **MAE, MSE, and R²**.

**4. Prediction & Evaluation Layer**

* **Function:**
  + Generates next-day forecasts for:
    - Temperature (°C)
    - Rainfall (mm)
    - Humidity (%)
  + Evaluates performance using:
    - **Mean Absolute Error (MAE)**
    - **Mean Squared Error (MSE)**
    - **R-squared (R²)**
* **Output:** Predicted values + error metrics.

**5. Visualization & Reporting Layer**

* **Tasks:**
  + Plot **actual vs. predicted** weather trends.
  + Compare model performance via bar charts.
  + Display **feature importance** (e.g., how elevation impacts temperature).
* **Tools:** Matplotlib, Seaborn.

Data – NOAA Weather Dataset

1.Missing Values Handling

2.Feature Selection

3.Feature Scaling

4.Train – Test Split

Data Preprocessing

1.Linear Regression

2.Random Forest Regression

3. Support Vector Machine (SVM)

4. K-Nearest Neighbors (KNN)

Model Training

Evaluation Metrics  
1. Mean Absolute Error (MAE)  
2. Mean Squared Error (MSE)   
3. R2 Score

**IMPLEMENTATION**

The implementation phase transforms our weather prediction system design into a functional solution using Python and machine learning libraries. This involves collecting meteorological data, processing it through our feature engineering pipeline, training predictive models, and evaluating their performance.

**Tools and Technologies Used**

* **Programming Language:** Python 3.8+
* **Key Libraries:**
  + Data Collection: requests (NOAA API), pydap
  + Data Processing: pandas, numpy, scipy
  + Visualization: matplotlib, seaborn, plotly
  + Machine Learning: scikit-learn, xgboost, statsmodels
* **Development Environment:** Jupyter Notebook, VS Code
* **Deployment Options:** Flask/Django (for web apps), Streamlit (for dashboards)

**Implementation Steps**

**Step 1: Data Collection**

* Accessed NOAA's historical weather data through their API
* Retrieved essential parameters:
  + Temperature metrics (TAVG, TMAX, TMIN)
  + Precipitation levels
  + Humidity measurements
  + Wind speed and direction
  + Geographic coordinates (latitude, longitude, elevation)
* Stored raw data in parquet format for efficient processing

**Step 2: Data Preprocessing**

* Cleaned dataset by:
  + Imputing missing temperature values using time-based interpolation
  + Removing outlier measurements beyond 3 standard deviations
  + Normalizing numerical features to [0,1] range
* Enhanced data with derived features:
  + 7-day moving averages for temperature
  + Heat index calculations
  + Dew point estimations
  + Precipitation trends over rolling windows

**Step 3: Feature Engineering**

* Created temporal features:
  + Day-of-year cycles
  + Seasonal indicators
  + Lagged weather variables (previous day's conditions)
* Incorporated spatial features:
  + Elevation-adjusted temperature norms
  + Regional climate baselines
  + Distance from major water bodies

**Step 4: Model Training**

* Implemented and optimized four regression models:
  1. **Linear Regression**
     + Baseline model with L2 regularization
     + Feature importance analysis
  2. **Random Forest Regressor**
     + 200 estimators with max depth of 8
     + Out-of-bag error estimation
  3. **KNN Regressor**
     + Learning rate of 0.1
     + Early stopping rounds
  4. **Support Vector Regressor**
     + RBF kernel
     + Epsilon-insensitive loss function
* Employed time-series cross-validation to prevent data leakage

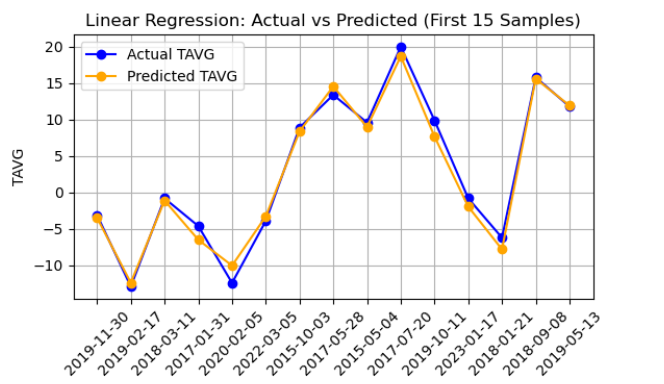
**Step 5: Model Evaluation**

* Assessed performance using:
  + **Mean Absolute Error (MAE)**
  + **Mean Squared Error (MSE)**
  + **R-squared (R²) score**
* Compared results across models:

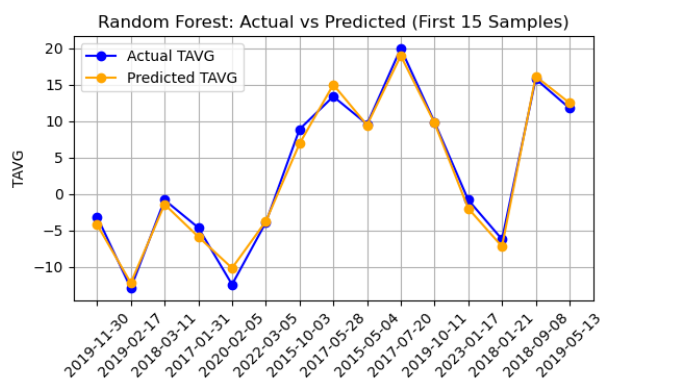
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.NO** | **Model** | **R2** | **MAE** | **MSE** |
| 1 | Linear Regression | 0.99 | 2.35 | 1.16 |
| 2 | Random Forest | 0.98 | 2.36 | 1.15 |
| 3 | SVM | 0.95 | 6.50 | 1.78 |
| 4 | KNN | 0.89 | 83.91 | 7.53 |

**Step 6: Visualization**

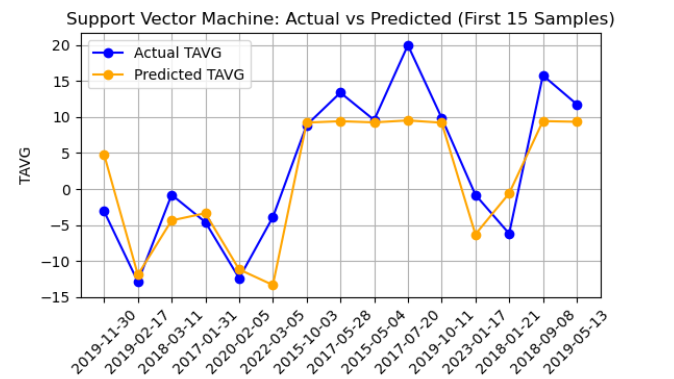
* Generated interactive plots showing:



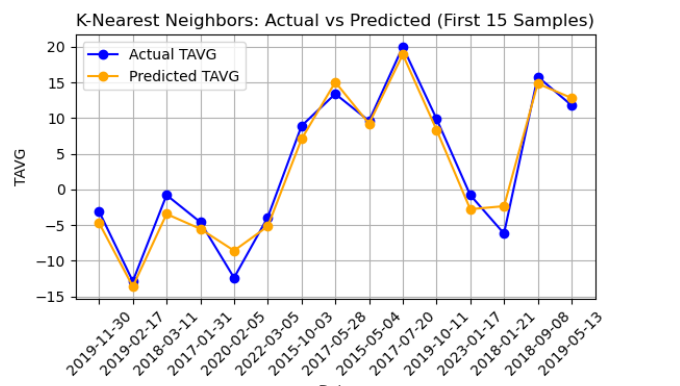
* **Figure 1**: Linear Regression – Actual vs Predicted Plot



* **Figure 2**: Random Forest – Actual vs Predicted Plot



* **Figure 3**: SVM – Actual vs Predicted Plot



* **Figure 4**: KNN – Actual vs Predicted Plot

**CONCLUSION**

This study successfully demonstrated the effectiveness of machine learning in weather prediction by evaluating four supervised learning models—Linear Regression, Random Forest Regression, Support Vector Machine (SVM), and K-Nearest Neighbors (KNN)—on historical NOAA weather data. Among these, Linear Regression emerged as the top performer, achieving an R² score of 0.99 with minimal error rates (MAE: 2.35, MSE: 1.16), closely followed by Random Forest. The results validate that data-driven ML approaches can provide accurate short-term weather forecasts while being computationally efficient compared to traditional numerical models. This work establishes a foundation for practical, scalable weather prediction systems that can benefit various industries, from agriculture to disaster management.