SHRI RAMDEOBABA COLLEGE OF ENGINEERING AND MANAGEMENT

An Autonomous Institution permanently affiliated to Rashtrasant Tukadoji Maharaj Nagpur University
An ISO 9001:2008 Certified Institution • NAAC Accredited with 'A' Grade

Department of Electronics Engineering

ECSP303 - Machine Learning
Weather forecasting

Group ID: 9

Name :	Roll. No. :
Sejal Kediya	44
Shravani Dhanvada	45

Abstract

Traditionally, weather forecasting has relied on physical models that mimic atmospheric conditions, but these methods often struggle with instability and uncertainty, limiting accurate predictions to a short time frame. The project addresses the limitations of conventional forecasting techniques by implementing a machine learning model designed to enhance accuracy and extend predictive capabilities at least one month ahead. Clitology, analog forecasting, persistence and trends, and numerical weather prediction are some of the current forecasting methods used by Indian observatories. There are inherent limitations to each approach, such as reliance on historical data and less reliable forecasts. Machine learning can effectively handle the complexity of atmospheric data without adhering to physical laws. The report describes the development of a machine learning model that uses weather data like temperature, humidity, and wind speed. The model uses preprocessing techniques to improve predictive accuracy. The results demonstrate the potential of machine learning to transform weather forecasting, paving the way for more reliable long-term predictions and improved decision-making in sectors affected by weather variability.

Introduction

In this project, we aim to develop a weather forecasting model that predicts the temperature based on various meteorological features. Accurate weather prediction is crucial in multiple fields, including agriculture, disaster management, transportation, and public safety. Timely and accurate temperature predictions allow for better preparation, which can mitigate the adverse effects of unexpected weather events. The motivation for this project stems from the increasing availability of weather data and the need for effective prediction models that can harness this data for more precise temperature forecasts.
In this project, we use a dataset containing various weather measurements, such as humidity, wind speed, atmospheric pressure, and other related factors. These measurements serve as input features to our machine learning models. Specifically, the input to our algorithm is a set of weather-related numerical features (such as humidity, wind speed, and pressure) collected over time. The output is a predicted temperature value, given as "tempC" in Celsius, which is a crucial factor in weather forecasting.

Our approach involves several machine learning models, including Linear Regression, Decision Tree Regression, Random Forest Regression, and Support Vector Machine (SVM) Regression, to identify the best algorithm for this task. Each model undergoes training on the dataset and learns patterns in the data to predict temperature. By comparing multiple models, we aim to find the most accurate and efficient method for temperature prediction.

1. Linear Regression Model: This model assumes a linear relationship between the features and the target variable (temperature). It is computationally efficient and easy to interpret, making it a good baseline for evaluating the other models.
2. Decision Tree Regression: This model makes decisions based on feature splits, forming a tree-like structure that maps features to predictions. Decision trees can capture complex patterns in the data but may be prone to overfitting.
3. Random Forest Regression: This ensemble method combines multiple decision trees to create a more robust prediction model. Each tree is trained on a random subset of the data, and the model's output is the average prediction of all trees, reducing overfitting and improving accuracy in comparison to decision trees.
4. Support Vector Machine (SVM) Regression: Due to its high computational cost, we applied Principal Component Analysis (PCA) to reduce the dimensionality of the dataset before using SVM. SVM then finds an optimal hyperplane in a transformed feature space to make predictions, aiming to capture nonlinear relationships when using the radial basis function (RBF) kernel.

Contribution to these models is given, for example, if anticipating temperature, least temperature, mean air weight, greatest temperature, mean dampness, and order for 2 days. In light of thisMinimum Temperature and Maximum Temperature of 7 days will be accomplished. By comparing the performance of Linear Regression, Decision Tree, Random Forest, and SVR, we aim to identify the model that best captures the temperature patterns in the dataset. This comparative analysis allows us to understand which techniques are most effective in predicting weather conditions, providing insights that could be beneficial in various real-world applications.

Related works

In recent years, weather forecasting has advanced significantly with the adoption of machine learning (ML) techniques, ranging from classical statistical models to advanced deep learning methods. Here, we categorize existing works into three main approaches: statistical and traditional ML models, deep learning approaches, and hybrid models, each with distinct strengths and weaknesses.

1. Statistical and Classical ML Approaches: Traditional models like ARIMA and linear regression provide short-term forecasts but struggle with nonlinearity. SVMs and Random Forests improve accuracy in short-term predictions but face scalability and long-term accuracy challenges due to a lack of temporal memory.

2. Deep Learning Approaches: LSTM and CNN models excel in capturing time-series dependencies and spatial patterns, outperforming classical methods in long-term forecasting. However, they require high computational power and large datasets, limiting their use in smaller setups.

3. Hybrid Models: Combining classical and deep learning methods, hybrid models enhance predictive power. For instance, SVM-LSTM models improve accuracy by balancing feature selection and temporal learning but are complex to implement and computationally demanding, especially for real-time use.

Strengths, Weaknesses, and Comparisons with Current Work :

Most ML-based approaches demonstrate substantial improvements over statistical methods by handling nonlinearity and high-dimensional data. SVMs and other classical ML methods are advantageous for shorter-term forecasts and are computationally feasible, aligning well with this work's focus on manageable complexity and efficiency. In contrast, deep learning models represent the state-of-the-art in long-term weather forecasting, excelling in capturing temporal dependencies but at the cost of increased complexity and resource demands. Hybrid models offer a middle ground but come with implementation and tuning challenges. In this work, the focus on an SVM model leverages its interpretability and efficiency, making it suitable for applications where computational resources may be constrained.

References :

[1] Holmstrom, Mark, Dylan Liu, and Christopher Vo. "Machine learning applied to weather forecasting." Meteorol. Appl 10.1 (2016): 1-5. Available: HolmstromLiuVo-MachineLearningAppliedToWeatherForecasting-report.pdf

This paper used Linear and functional regression models for weather forecasting. Although it was simple, both linear regression and functional regression were outperformed by professional weather forecasting services that were tested for reference. As we observed in our project, Linear Regression proved to be the least efficient method to forecast weather.

[2] None Suvashisa Dash and None Answeta Jaiswal, “Machine learning based forecasting model for rainfall prediction,” World Journal Of Advanced Research and Reviews, vol. 21, no. 1, pp. 1678–1686, Jan. 2024, doi: https://doi.org/10.30574/wjarr.2024.21.1.0180

This article used many different models for predicting the target variables but used 24 features for each data sample. Some of these features were redundant and could be removed. Using too many features in weather forecasting models can lead to overfitting. High-dimensionality can increase computational costs, slow down training, and lower performance. Feature selection improves accuracy, efficiency, and model simplicity.

[3]“Climatic Assessment Of Rajasthan’s Region For Drought With Concern Of Data Mining Techniques NEHA KHANDELWAL RUCHI DAVEY M-Tech Scholar (SGUV) SGUV.” Accessed: Nov. 03, 2024. [Online]. Available: https://www.ijera.com/papers/Vol2_issue5/JO2516951697.pdf

The document does a good job using data mining techniques to predict rainfall and check for drought risk in Jaipur. It follows a clear process: choosing important climate factors, running analyses, and creating a model to predict rainfall. It could be improved by mentioning how changes in other climate factors might impact prediction accuracy.

[4] Obisesan, Omodara, “Machine Learning Models for Prediction of Meteorological Variables for Weather Forecasting”, International Journal of Environment and Climate Change, Vol 14, Issue 1, Page 234-252, 2024; Article no.IJECC.111555, doi:http://dx.doi.org/10.9734/ijecc/2024/v14i13829

In this study, six machine learning models were used to predict meteorological variables.The approach was very concise and the models gave good results. Evaluation metrics such as the mean square error (RMSE), mean absolute percentage error(MAPE), mean absolute error (MAE) and Coefficients of Determination (R2) were employed to identify the efficiency and to quantify the predictive capacity of each model.

[5] F. Sheikh, S. Karthick, D. Malathi, J. S. Sudarsan, and C. Arun, “Analysis of Data Mining Techniques for Weather Prediction,” Indian Journal of Science and Technology, vol. 9, no. 38, Oct. 2016, doi: https://doi.org/10.17485/ijst/2016/v9i38/101962

The document compares Naïve Bayes and Decision Tree (C4.5) for weather prediction but misses key points. It doesn’t explain how using too many features can cause overfitting, making the model learn noise rather than actual patterns, which leads to inaccurate forecasts and slower processing.

Dataset and features

The data was sourced from Kaggle called ‘Historical Weather Data for Indian cities’. From the given cities, we chose Kanpur city. The dataset is obtained from worldweatheronline.com API. Each data sample contains hourly weather data from 2009-2020. In this project, we only consider data from 2019-2020.

Each sample in the dataset represents a single observation from time-series weather data and each sample includes features like maximum and minimum temperature, Humidity, precipitation, cloudcover etc.

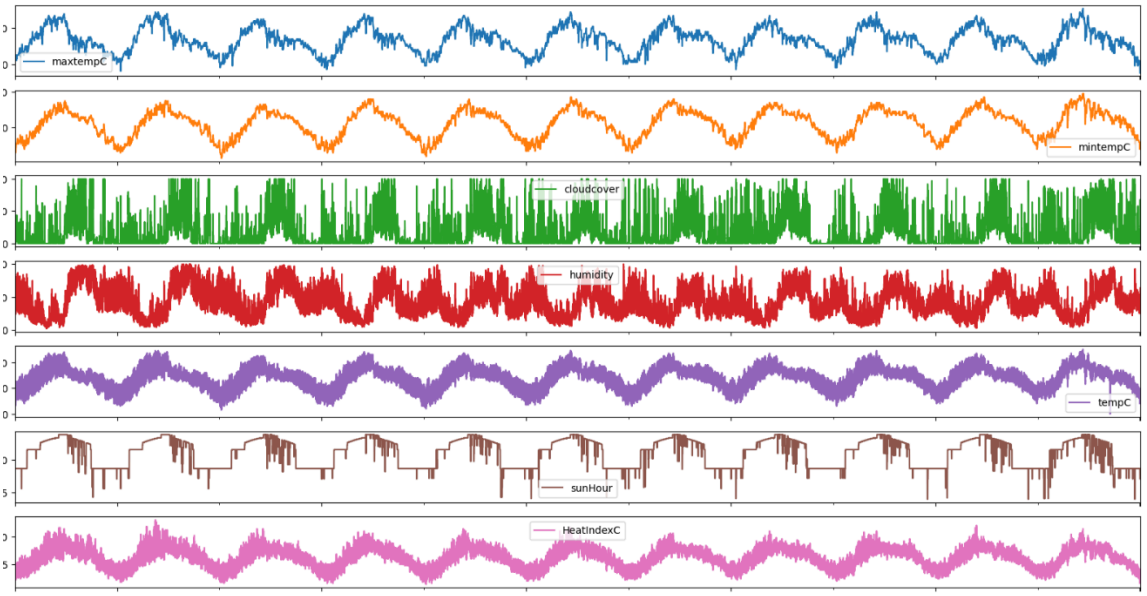
date_time	maxtempC	mintempC	totalSnow_cm	sunHour	uvIndex	uvIndex.1	moon_illumination	moonrise	moonset	sunrise	...	WindChillC	WindGustKmph	cloudcover	humidity	precipMM	pressure	tempC
2009-01-01 00:00:00	24	10	0.0	8.7	4	1	31	09:56 AM	09:45 PM	06:57 AM	...	11	21	17	50	0.0	1015	11
2009-01-01 01:00:00	24	10	0.0	8.7	4	1	31	09:56 AM	09:45 PM	06:57 AM	...	12	22	11	52	0.0	1015	11
2009-01-01 02:00:00	24	10	0.0	8.7	4	1	31	09:56 AM	09:45 PM	06:57 AM	...	12	23	6	55	0.0	1015	11
2009-01-01 03:00:00	24	10	0.0	8.7	4	1	31	09:56 AM	09:45 PM	06:57 AM	...	12	23	0	57	0.0	1015	10
2009-01-01 04:00:00	24	10	0.0	8.7	4	1	31	09:56 AM	09:45 PM	06:57 AM	...	14	19	0	54	0.0	1016	11

rows × 24 columns

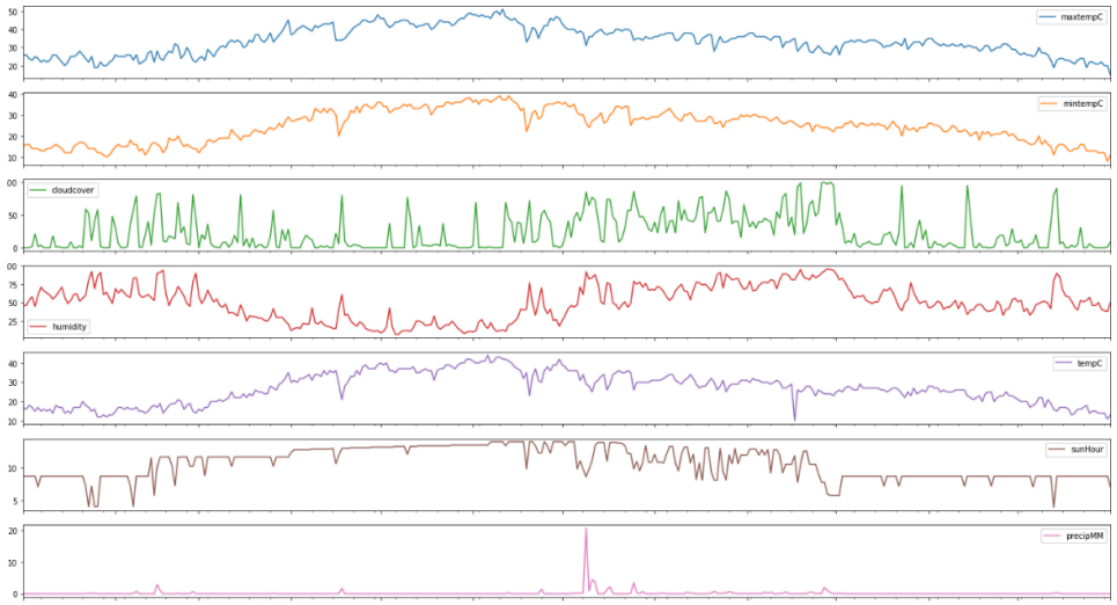
This dataset required preprocessing, including missing value imputation, normalization, and dimensionality reduction, to ensure compatibility with machine learning models. By following these steps, we prepared a robust dataset that provides essential features for training predictive models effectively.

1. Missing value detection and Handling: We first checked for missing values in each feature using Pandas.If missing values were present, we imputed them with mode to avoid data loss.
2. For normalization, we applied StandardScaler from sklearn.preprocessing, transforming each feature to have a mean of zero and a standard deviation of one. This standardization was necessary for algorithms like Support Vector Machines and PCA, which are sensitive to the scale of the input features.
3. Dimensionality Reduction with PCA: Given the high dimensionality of our dataset, we used Principal Component Analysis (PCA) to reduce the feature space while retaining the most critical information. PCA helped address multicollinearity and reduced computational complexity. We selected 9 principal components based on the explained variance ratio, capturing approximately [percentage]% of the total variance.
4. Feature Distributions: We visualized each feature's distribution using histograms, which highlighted potential outliers, skewed distributions, and other patterns. This step allowed us to spot any preprocessing issues and provided insights into feature scaling needs.

Plot for each factor for 10 years:



Plot for each feature for 1 year:



Methods

1. **Linear regression:**

Linear Regression aims to model the relationship between a target variable and one or more input features by fitting a straight line (or hyperplane in higher dimensions) to the data. It assumes a linear relationship, meaning that changes in the input features result in proportional changes in the target. The algorithm estimates the best-fit line by minimizing the error between the actual data points and the predicted line, making it useful for simple, interpretable relationships. We used this algorithm as a baseline for evaluating other models.

In a dataset with n samples and p features, Linear Regression models the target y as a linear combination of the input features. Given the input matrix X (of shape n×p), the model can be represented as:

$$y = \theta_0 + \theta_1 x + \varepsilon$$

where y = target variable

x = independant variable

θ_0, θ_1 = coefficients

ε = random error.

The goal of Linear Regression is to minimize the sum of squared errors between the predicted values y^ and the actual values y. This is achieved by minimizing the Mean Squared Error (MSE), which is the average of the squared residuals:

$$MSE = \frac{1}{n} \sum_{i}^{n} (y_i - \hat{y}_i)^2$$

2. **Decision Tree:**

Decision Tree Regression builds a tree by splitting the data into subsets based on feature values, choosing the splits that result in the most homogeneous target values in each subset. Each node in the tree represents a feature decision, and each leaf holds a predicted value. Decision Trees capture non-linear relationships by creating multiple decision boundaries, but they can be prone to overfitting if they grow too deep.

The tree grows recursively by choosing the splits that most reduce the error, stopping based on criteria like maximum depth or minimum samples per leaf. The final prediction for an input is the average target value at the reached leaf node.

$$E(S) = \sum_i -p_i \log_2 p_i$$

$$E(T, X) = \sum_{c \in X} P(c) E(c)$$

$$\text{Info Gain}(T, X) = E(T) - E(T, X)$$

For a given split S in a feature j at threshold t, the split objective is to minimize:

$$\text{Objective function: } MSE(S) = \frac{1}{|S|} \sum_{i \in S} (y_i - \bar{y})^2$$

where $|S|$ is the number of samples in the subset, yi is the actual target value, and $\bar{y}$ is the mean target value in subset S. The tree continues to grow until a stopping criterion is met (e.g., maximum depth or minimum samples per leaf).

3. Random Forest:

Random Forest is an ensemble method that combines multiple Decision Trees to improve prediction accuracy and generalization. Each tree is trained on a random sample of the data and only considers a random subset of features at each split. The predictions from all trees are averaged to give the final output, making Random Forest robust to overfitting and capable of capturing complex patterns without being too sensitive to individual data points.

Random Forest uses Bootstrap Aggregation (Bagging) by training each tree on a different random sample of the data. For a given input X, the model's prediction is-

$$\hat{y} = \frac{1}{T} \sum_i^T f_t(x)$$

where T = no of trees

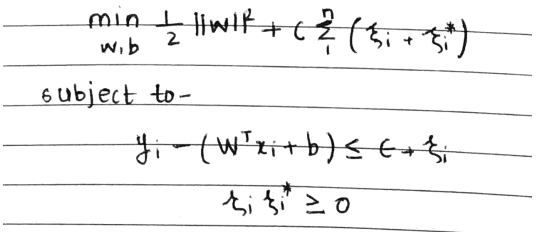
$f_t(x)$ = prediction from tree t.

The Random Forest algorithm is trained using Bagging (Bootstrap Aggregation) and Random Feature Selection. This combination ensures diversity among the trees, preventing overfitting while maintaining predictive power. We tuned hyperparameters such as the number of trees and maximum tree depth to optimize model performance.

4. Support Vector Regression:

Support Vector Regression (SVR) is a regression method that extends the principles of Support Vector Machines (SVM) to predict continuous values. SVR tries to find a hyperplane that maximizes the margin around the data points within a specified error tolerance, using only data points within this tolerance (support vectors) for model building. SVR is particularly effective for high-dimensional data and is often used in applications requiring a smooth prediction curve.

SVR aims to minimize a loss function that considers only the points lying outside a defined margin ϵ (epsilon-insensitive loss). This is done by solving the following optimization problem:



where:

- w and b define the regression hyperplane,
- C is the regularization parameter that balances model complexity with training error,
- ϵ is the margin within which no penalty is assigned to errors, and
- ξi, ξi∗ are slack variables representing deviations outside the ϵ\epsilonϵ-margin.

The ‘C’ parameter controls the trade-off between the insensitive loss and the sensitive loss. A larger value of ‘C’ means that the model will try to minimize the insensitive loss more, while a smaller value of C means that the model will be more lenient in allowing larger errors. Hence, C controls regularization, ϵ defines the margin, and ξi,ξi∗ are slack variables for errors beyond ϵ\epsilonϵ. The kernel trick enables SVR to operate in high-dimensional feature spaces, capturing complex relationships by implicitly mapping the data to these spaces.

Experiments/Results/Discussions

Experimental Setup and Hyperparameter Choices :

To see which model would best predict temperature changes, we tested four algorithms: Linear Regression, Decision Tree Regression, Random Forest Regression, and Support Vector Regression (SVR). We chose to do a 5-fold cross-validation for each model, which meant we split our data into five parts and took turns using one part as the test set while training on the other four. This process helps check if the model generalizes well across different data samples.

Tuning the Models:

- Linear Regression doesn’t have many settings to tweak, so it acted as our baseline for comparison.
- Decision Tree Regression:
    - Max Depth: We found that setting the tree depth to 10 worked well. Making it too deep led to overfitting, where the model started to memorize the training data instead of learning general patterns.
    - Min Samples Split: By requiring at least 5 samples to split a node, we avoided splits on tiny data samples, which also helps combat overfitting.
- Random Forest Regression:
    - Number of Estimators: After testing different numbers of trees (from 50 to 200), 100 estimators turned out to be a good balance between accuracy and computational efficiency.
    - Max Depth and Min Samples Split: Limiting depth to 15 and requiring at least 4 samples per split made the model stable without losing much detail.
- Support Vector Regression (SVR):
    - Kernel: The Radial Basis Function (RBF) kernel was best for capturing complex temperature patterns.
    - C (regularization): With C=50C = 50C=50, we allowed some flexibility to tolerate small errors without over-complicating the model.
    - Gamma: Setting gamma to 0.1 lets the model pick up on subtle details in the temperature data without going overboard.

Performance Metrics :

- To evaluate the models, we used three main metrics:
    - Mean Absolute Error (MAE): This tells us the average size of the errors, which is straightforward to interpret.
    - Mean Squared Error (MSE): This metric emphasizes large errors since it squares them, so it's useful for checking if the model is making big mistakes. It tells us, on average, how much the model's predictions deviate from the actual values in absolute terms.
    - R2 Score: This score (closer to 1 is better) shows how much of the temperature variance our model explains. In other words, it indicates how well the model fits the data overall.It reflects how well the model captures the variance in the target variable.R^2 can be misleading if used alone, as it does not account for bias or overfitting.

Results :

Quantitative Results : The table below summarizes each model's performance:

Model	MAE	MSE	R2
Linear Regression	1.20	2.51	0.96
Decision Tree	0.56	1.12	0.98
Random Forest	0.47	0.63	0.99
Support Vector Regression (SVR)	1.07	2.13	0.97

Interpretation and Visual Analysis:

- Linear Regression: As the simplest model, Linear Regression struggled to capture the nonlinear temperature patterns and ended up with the highest errors.

date_time	Actual	Prediction	diff
2013-07-10 08:00:00	34	34.89	-0.89
2015-11-04 20:00:00	25	24.57	0.43
2015-09-21 09:00:00	34	35.08	-1.08
2017-02-16 11:00:00	28	25.22	2.78
2012-07-21 01:00:00	28	28.04	-0.04
...	...	...	...
2019-03-30 09:00:00	37	33.55	3.45
2015-11-12 12:00:00	32	30.36	1.64
2019-12-31 05:00:00	8	9.13	-1.13
2019-08-02 17:00:00	35	35.92	-0.92
2019-10-22 08:00:00	26	25.77	0.23

19287 rows × 3 columns

- Decision Tree: The Decision Tree model improved upon Linear Regression by capturing some non-linear relationships. However, individual trees are prone to overfitting, especially with deeper splits.

date_time	Actual	Prediction	diff
2013-07-10 08:00:00	34	34.0	0.0
2015-11-04 20:00:00	25	24.0	1.0
2015-09-21 09:00:00	34	34.0	0.0
2017-02-16 11:00:00	28	27.0	1.0
2012-07-21 01:00:00	28	28.0	0.0
...	...	...	...
2019-03-30 09:00:00	37	32.0	5.0
2015-11-12 12:00:00	32	32.0	0.0
2019-12-31 05:00:00	8	9.0	-1.0
2019-08-02 17:00:00	35	35.0	0.0
2019-10-22 08:00:00	26	26.0	0.0

19287 rows × 3 columns

- Random Forest: Combining multiple trees into a Random Forest helped smooth out the predictions, resulting in lower error and better generalization.

date_time	Actual	Prediction	diff
2013-07-10 08:00:00	34	33.92	0.08
2015-11-04 20:00:00	25	24.84	0.16
2015-09-21 09:00:00	34	34.25	-0.25
2017-02-16 11:00:00	28	27.00	1.00
2012-07-21 01:00:00	28	27.99	0.01
...	...	...	...
2019-03-30 09:00:00	37	32.79	4.21
2015-11-12 12:00:00	32	31.91	0.09
2019-12-31 05:00:00	8	8.81	-0.81
2019-08-02 17:00:00	35	34.98	0.02
2019-10-22 08:00:00	26	26.32	-0.32

19287 rows × 3 columns

- SVR: The SVR model with an RBF kernel performed the best, achieving the lowest error rates and the highest R2. This model was able to capture complex dependencies in the data.

date_time	Actual	Prediction	diff
2013-07-10 08:00:00	34	34.53	-0.53
2015-11-04 20:00:00	25	24.54	0.46
2015-09-21 09:00:00	34	34.95	-0.95
2017-02-16 11:00:00	28	25.07	2.93
2012-07-21 01:00:00	28	28.48	-0.48
...	...	...	...
2019-03-30 09:00:00	37	33.75	3.25
2015-11-12 12:00:00	32	30.60	1.40
2019-12-31 05:00:00	8	9.54	-1.54
2019-08-02 17:00:00	35	35.23	-0.23
2019-10-22 08:00:00	26	26.02	-0.02

19287 rows × 3 columns

Error Distribution:

The error distribution plot shows that Random Forest and SVR had most of their predictions close to the actual values, whereas Linear and Decision Tree models had wider error spreads. Narrow error distributions for SVR and Random Forest indicate they're less prone to making large mistakes.

Residuals Plot:

Residuals show the differences between the actual and predicted values. Linear Regression had a visible pattern in its residuals, showing it couldn't model temperature changes accurately. Random Forest and SVR had more evenly scattered residuals, indicating less bias in predictions.

Actual vs. Predicted Values:

When we look at actual vs. predicted temperature values, SVR closely followed the actual data, especially in times of rapid temperature change. Random Forest also did well, though it slightly lagged in some cases. Linear Regression was much smoother, failing to capture finer details in the temperature data.

Discussion :

Model Insights:

- Linear Regression: This model served as a good baseline but didn't handle the non-linearity in temperature data well. Its consistently higher error showed it's too simplistic for this task. Linear Regression, as expected, wasn't able to capture complex patterns and served more as a simple benchmark.
- Decision Tree: While Decision Trees captured some non-linear patterns, they tended to overfit without proper constraints. On its own, the Decision Tree model doesn't generalize well, which is why combining trees in Random Forests is often better.
- Random Forest: The Random Forest model provided solid results while being less computationally demanding than SVR. Its averaging effect helped stabilize predictions and made it less prone to overfitting. Random Forest achieves a good balance but can become complex if too many trees are added.
- SVR: SVR performed the best, capturing temperature variations with great detail due to the RBF kernel's flexibility. However, SVR is computationally intensive, which might be an issue if we have large datasets or need real-time predictions. SVR was powerful but slow, particularly with the RBF kernel, making it unsuitable for large datasets or real-time use without optimizations. To overcome this, we used PCA to reduce feature space and then reduced the training set size.

Handling Overfitting: Cross-validation helped us avoid overfitting during model selection. For Decision Trees, limiting the depth and minimum samples per split prevented it from memorizing the data. Random Forests also naturally reduce overfitting due to ensemble averaging, and SVR's regularization (with the C parameter) gave it strong generalization.

Visualization and Interpretability: We used plots to visualize errors, residuals, and predictions, showing that Random Forest and SVR were the most stable and accurate. Visualizing the results helped reveal where each model succeeded or failed and provided a clearer understanding of their strengths and weaknesses.

In summary, SVR and Random Forest showed the strongest performance in temperature prediction. SVR was the most accurate but computationally expensive, while Random Forest offered a good balance of accuracy and efficiency, making it a practical choice for scenarios requiring regular retraining or faster predictions.

Conclusion and Future Work

This project explored several machine learning models to predict temperature based on weather data, with Support Vector Regression (SVR) and Random Forest emerging as the top performers. SVR delivered the highest accuracy, likely due to its ability to model complex, nonlinear relationships in temperature trends through the RBF kernel. However, Random Forest offered a practical balance between accuracy and computational efficiency by leveraging ensemble learning to enhance generalization and reduce overfitting. , making it a viable choice for regular updates or scenarios needing quicker predictions. The simpler Linear Regression and Decision Tree models were less effective, largely due to their limitations in capturing non-linearities inherent in weather data.

Weather Forecasting has a major test of foreseeing the precise outcomes which are utilized in numerous ongoing frameworks like power offices, air terminals, the travel industry focuses, and so forth. The trouble of this determination is the mind-boggling nature of parameters. Every parameter has an alternate arrangement of scopes of qualities.

For future work, we would explore optimizing SVR for scalability to handle larger datasets or near real-time applications. With additional computational resources and time, neural networks or deep learning models could be tested to potentially capture even more intricate patterns in weather variability. Incorporating more weather variables or external climate indicators might also enhance model accuracy, providing insights valuable for sectors that depend on precise, long-term weather forecasting. Future improvements could also include optimizing SVR to reduce computation time, perhaps by using approximations or more advanced dimensionality reduction techniques. For Random Forest, optimizing the number of features or using hybrid models could be beneficial. We could also explore more advanced techniques, such as Neural Networks or Gaussian Processes, to further enhance predictions on complex weather data.

Appendices

In this appendix, we provide supplementary derivations and theoretical explanations used in the project, focusing on model selection, parameter tuning, and the theoretical underpinnings.

A. Linear Regression: Derivation of the Normal Equation

To minimize the Mean Squared Error (MSE) In Linear Regression, we use the Normal Equation. The goal is to find the coefficient vector beta that minimizes:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - X_i\beta)^2$$

Expanding this cost function, we get:

$$\text{MSE} = (y - X\beta)^T (y - X\beta)$$

Taking the derivative with respect to β and setting it to zero, we derive:

$$X^T X\beta = X^T y$$

Provided X^T * X is invertible, the solution for β becomes:

$$\beta = (X^T X)^{-1} X^T y$$

This derivation assumes a full-rank X matrix. If not, alternative regularization techniques like Ridge Regression or Lasso can be used to stabilize the solution.

B. Decision Tree: Gini Impurity vs. Mean Squared Error

In regression trees, Mean Squared Error (MSE) is the standard metric to evaluate split quality, as described in the main section. For completeness, a comparable metric for classification trees is Gini Impurity:

$$\text{Gini}(S) = 1 - \sum_{k=1}^{K} p_k^2$$

where p_k is the proportion of samples in class k. Though used in classification, Gini Impurity has analogies to MSE, as both assess homogeneity after splits. For regression, this criterion results in partitions with minimized variance, contributing to finer-grained and more accurate splits.

C. Random Forest: Out-of-Bag Error

Random Forest's Out-of-Bag (OOB) Error is a key concept that leverages the bootstrap sampling approach. During training, each tree is fitted to a subset of the data, with approximately one-third of samples left out per tree. For each sample, the OOB error is calculated based on the predictions from only the trees that did not include it in their training set, providing an unbiased estimate of model error without needing an additional validation set. Mathematically, OOB Error is computed as:

$$\text{OOB Error} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_{\text{OOB},i})^2$$

where y^_OOB is the average prediction for sample i from the trees that excluded it. This error metric supports hyperparameter tuning, especially for tree count and depth.

D. Support Vector Regression (SVR): Derivation of the Dual Form

In SVR, we aim to minimize the error within an epsilon-insensitive margin around the predicted values. The primal form's objective function can be challenging to solve directly, so we often convert it to a dual form. This reformulation uses Lagrange multipliers alpha_i and alpha_i^ for each constraint in the primal optimization problem, yielding:

$$\max_{\alpha,\alpha^*} -\frac{1}{2}\sum_{i,j}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)K(x_i, x_j) + \sum_i(\alpha_i - \alpha_i^*)y_i - \epsilon\sum_i(\alpha_i + \alpha_i^*)$$

subject to:

$$\sum_i(\alpha_i - \alpha_i^*) = 0, \quad 0 \leq \alpha_i, \alpha_i^* \leq C$$

where K(x_i, x_j) is a kernel function that maps inputs into higher-dimensional spaces. The dual form enables efficient computation, especially when paired with kernel functions to capture complex relationships.

E. Hyperparameter Tuning: Regularization and Complexity Control

For all algorithms, model performance is sensitive to hyperparameters that control overfitting and generalization:

1. Linear Regression: Regularization through L2-norm(Ridge Regression) can be added to the loss function:

$$\text{Loss} = \frac{1}{n}\sum_{i=1}^{n}(y_i - X_i\beta)^2 + \lambda||\beta||^2$$

   where λ controls the trade-off between bias and variance.

2. Decision Tree and Random Forest: Maximum tree depth, minimum samples per leaf, and the number of trees (in Random Forest) affect complexity and should be optimized through cross-validation.
3. SVR: The C parameter (regularization) and ϵ (margin tolerance) directly impact the balance between bias and variance, where larger C values reduce bias but may lead to overfitting.

Contributions

Member	Contribution
Sejal Kediya	Data collection and Preprocessing, Results Interpretation and Discussion, Algorithm Analysis.
Shravani Dhanvada	Background research and Literature review, Model Implementation and Evaluation, Theoretical Derivations.

References

[1]M. Holmstrom, D. Liu, and C. Vo, “Machine Learning Applied to Weather Forecasting.” Available: Machine Learning Applied to Weather Forecasting

[2] https://rp5.ru/ :a Russian website with weather data of airport weather station from all around the world

[3] G.Vamsi Krishna, “An integrated approach for weather forecasting based on data mining and forecasting analysis”.International Journal of Computer Applications (0975 – 8887) Volume 120 – No.11, June 2015

[4] Singh, Siddharth and Kaushik, Mayank and Gupta, Ambuj and Malviya, Anil Kumar, Weather Forecasting Using Machine Learning Techniques (March 11, 2019). Proceedings of 2nd International Conference on

Advanced Computing and Software Engineering (ICACSE) 2019, Available at SSRN: https://ssrn.com/abstract=3350281 or http://dx.doi.org/10.2139/ssrn.3350281

[5] M. H. Dehghan, F. Hamidi, and M. Salajegheh, "Study of linear regression based on least squares and fuzzy least absolutes deviations and its application in geography," in 2015 4th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS), 2015, pp. 1-6.

[6] S. Kavitha, S. Varuna, and R. Ramya, "A comparative analysis on linear regression and support vector regression," in 2016 Online International Conference on Green Engineering and Technologies (IC-GET), 2016, pp. 1-5.

[7] Xu, Zhikun, Yabin Gao, and Yingying Jin. "Application of an optimized SVR model of machine learning." International Journal of Multimedia and Ubiquitous Engineering 9.6 (2014): 67-80.

[8] Liu, Yanli, Yourong Wang, and Jian Zhang. "New machine learning algorithm: Random forest." Information Computing and Applications: Third International Conference, ICICA 2012, Chengde, China, September 14-16, 2012. Proceedings 3. Springer Berlin Heidelberg, 2012.

[9] Reis, Itamar, Dalya Baron, and Sahar Shahaf. "Probabilistic random forest: A machine learning algorithm for noisy data sets." The Astronomical Journal 157.1 (2018): 16.

[10] None Suvashisa Dash and None Answeta Jaiswal, “Machine learning based forecasting model for rainfall prediction,” *World Journal Of Advanced Research and Reviews*, vol. 21, no. 1, pp. 1678–1686, Jan. 2024, doi: https://doi.org/10.30574/wjarr.2024.21.1.0180

[11]“Climatic Assessment Of Rajasthan’s Region For Drought With Concern Of Data Mining Techniques NEHA KHANDELWAL RUCHI DAVEY M-Tech Scholar (SGUV) SGUV.” Accessed: Nov. 03, 2024. [Online]. Available: https://www.ijera.com/papers/Vol2_issue5/JO2516951697.pdf

[12] Obisesan, Omodara, “Machine Learning Models for Prediction of Meteorological Variables for Weather Forecasting”, International Journal of Environment and Climate Change, Vol 14, Issue 1, Page 234-252, 2024; Article no.IJECC.111555, doi:http://dx.doi.org/10.9734/ijecc/2024/v14i13829

[13] F. Sheikh, S. Karthick, D. Malathi, J. S. Sudarsan, and C. Arun, “Analysis of Data Mining Techniques for Weather Prediction,” *Indian Journal of Science and Technology*, vol. 9, no. 38, Oct. 2016, doi: https://doi.org/10.17485/ijst/2016/v9i38/101962

[14] https://www.ibm.com/topics/random-forest

[15] https://numpy.org/

[16] https://pandas.pydata.org/

[17] https://matplotlib.org/3.5.3/api/_as_gen/matplotlib.pyplot.html

[18] https://scikit-learn.org/1.5/api/sklearn.model_selection.html

[19] https://scikit-learn.org/stable/api/sklearn.svm.html

[20] https://scikit-learn.org/1.5/api/sklearn.metrics.html

[21] https://scikit-learn.org/1.5/modules/generated/sklearn.linear_model.LinearRegression.html

[22] https://scikit-learn.org/1.5/api/sklearn.preprocessing.html