

III B.TECH I-SEMESTER

MACHINE LEARNING

Project Report Project Title:

Weather Prediction using Historical Data

Done By:

22251A0518- K.Gayathri

22251A0526- P. Srikari Alekya

22251A0534- B.Bhavani

1. Abstract

Weather prediction plays a crucial role in sectors such as agriculture, transportation, and emergency management. Traditional forecasting techniques often face limitations in analyzing complex weather patterns. This project leverages supervised machine learning (ML) algorithms—Decision Trees, Logistic Regression, K-Nearest Neighbors (KNN), Random Forest, Gradient Boosting, and Naive Bayes—to predict daily weather conditions using historical meteorological data. Our approach emphasizes data preprocessing, feature scaling, and the evaluation of algorithmic performance to identify the most suitable ML model for this task. The Decision Tree model achieved the highest accuracy (83.87%), making it the preferred choice for implementation.

2. Introduction

Weather prediction is an essential component of modern life, influencing agriculture, transportation, and emergency planning. Accurate forecasting is crucial for mitigating the impacts of adverse weather conditions. Traditional forecasting methods have limitations in analysing complex patterns in weather data. Machine learning (ML) offers a data-driven approach to improve forecasting by leveraging historical weather data. Machine learning provides a robust framework for weather prediction by analysing historical data to uncover complex relationships that traditional methods often miss.

2.1 Existing Systems and Challenges

Existing forecasting systems rely on statistical techniques and physical models, which may struggle with dynamic, non-linear relationships in weather data. Challenges include:

- Limited scalability and generalizability.
- Sensitivity to missing or noisy data.

2.2 Proposed System

This project applies multiple supervised ML algorithms to classify weather conditions. The system focuses on:

- Identifying the most effective algorithm for weather prediction.
- Analysing the role of key weather parameters in improving prediction accuracy.
- Ensuring computational efficiency for practical use.

3.Literature Review

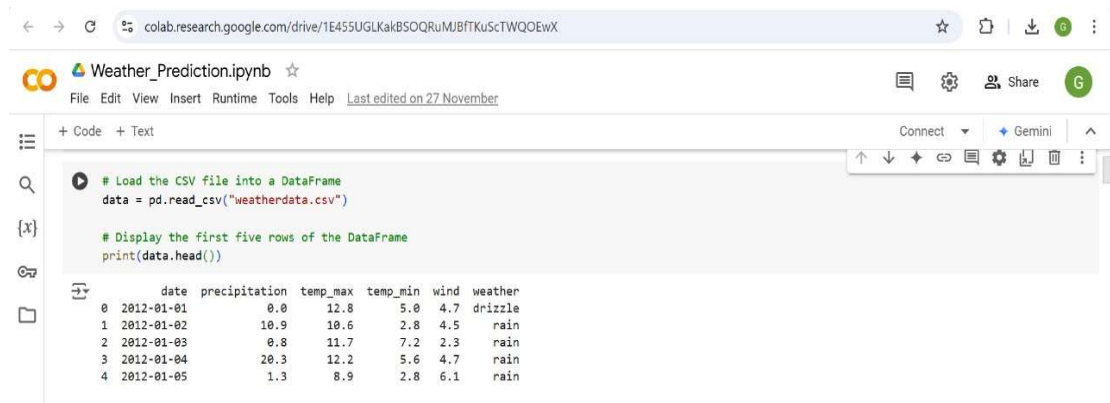
Traditional weather prediction methods, such as physical models, simulate atmospheric conditions but often fail to capture micro-level variations, while statistical methods like linear regression assume simplistic relationships, limiting predictive accuracy. Machine learning (ML) offers a more nuanced approach, with ensemble models like Random Forest and Gradient Boosting excelling at handling feature interactions and variability. Naive Bayes performs efficiently with categorical data but struggles with correlated features, whereas Logistic Regression remains a reliable baseline for binary and multiclass classification tasks. Advanced ML techniques, including neural networks, can achieve higher accuracy but require extensive datasets and computational resources. Research underscores the importance of combining preprocessing techniques, such as feature scaling and encoding, with ensemble learning to enhance performance. Feature engineering and hyperparameter tuning are also vital for optimizing model outcomes in weather prediction.

4. Methodology

4.1 Dataset Collection

The dataset (weatherdata.csv) includes:

- **Temperature:** Degree of heat.
- **Humidity:** Percentage of atmospheric moisture.
- **Wind Speed:** Significant for weather patterns.
- **Weather Condition:** Target variable (e.g., sunny, cloudy, rainy).



The screenshot shows a Google Colab notebook titled "Weather_Prediction.ipynb". The code cell contains the following Python code:

```
# Load the CSV file into a DataFrame
data = pd.read_csv("weatherdata.csv")

# Display the first five rows of the DataFrame
print(data.head())
```

The output of the code is a DataFrame with 5 rows and 6 columns:

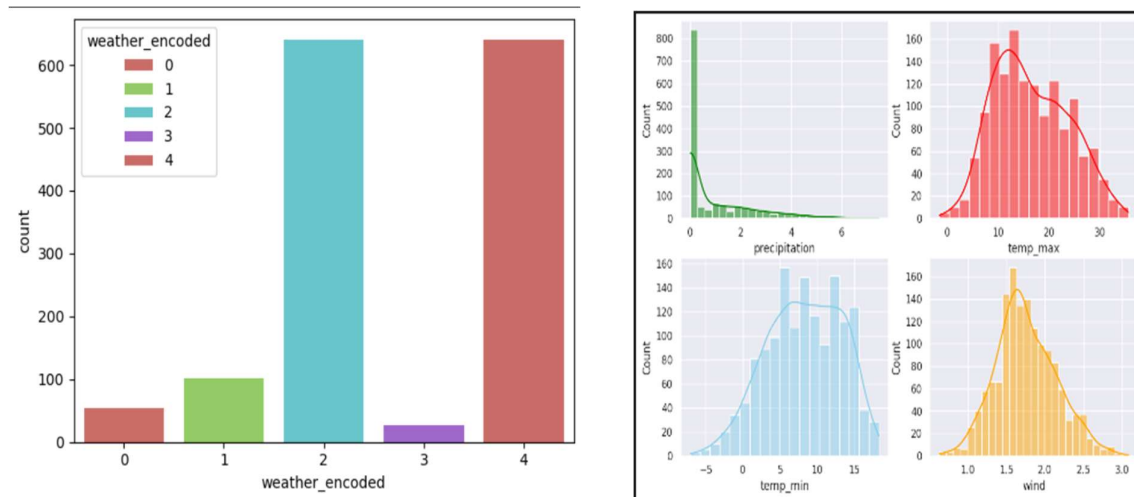
	date	precipitation	temp_max	temp_min	wind	weather
0	2012-01-01	0.0	12.8	5.0	4.7	drizzle
1	2012-01-02	10.9	10.6	2.8	4.5	rain
2	2012-01-03	0.8	11.7	7.2	2.3	rain
3	2012-01-04	20.3	12.2	5.6	4.7	rain
4	2012-01-05	1.3	8.9	2.8	6.1	rain

4.2 Data Preprocessing

- **Handling Missing Values:** Replaced missing values with mean/median.
- **Encoding Categorical Variables:** Used label encoding for compatibility.
- **Feature Scaling:** Standardized numerical features using StandardScaler.
- **Data Splitting:** Divided data into 80% training and 20% testing subsets.

4.3 Exploratory Data Analysis

- **Histograms:** Analysed frequency distributions.
- **Correlation Heatmap:** Visualized feature relationships.
- **Boxplots:** Identified outliers for robust preprocessing.



The `weather_encoded` bar chart reveals a class imbalance, with categories 2 and 4 dominating while categories 0 and 3 are underrepresented. This imbalance can bias model predictions toward majority classes. The numerical feature histograms show diverse distributions. **Precipitation** is highly right-skewed, with most values near zero, indicating infrequent heavy precipitation events. The **temp_max** and **temp_min** features exhibit nearly normal distributions. **Wind speed** shows a slight right skew, with most values concentrated at lower speeds, reflecting typical weather patterns. Standardization or normalization is crucial to ensure all features contribute proportionately during model training, especially for features with varying ranges like precipitation and temperature.

4.4 Classification Algorithms

1. **Decision Tree:** Modelled non-linear relationships with Gini impurity and entropy for split quality.
2. **Logistic Regression:** Applied a one-vs-rest strategy for multiclass classification.
3. **KNN:** Evaluated using varying values of `k` and Euclidean distance.
4. **Random Forest:** Built an ensemble of 100 trees, analysing feature importance.
5. **Gradient Boosting:** Tuned hyperparameters to balance bias and variance.
6. **Naive Bayes:** Used Gaussian Naive Bayes for continuous data.

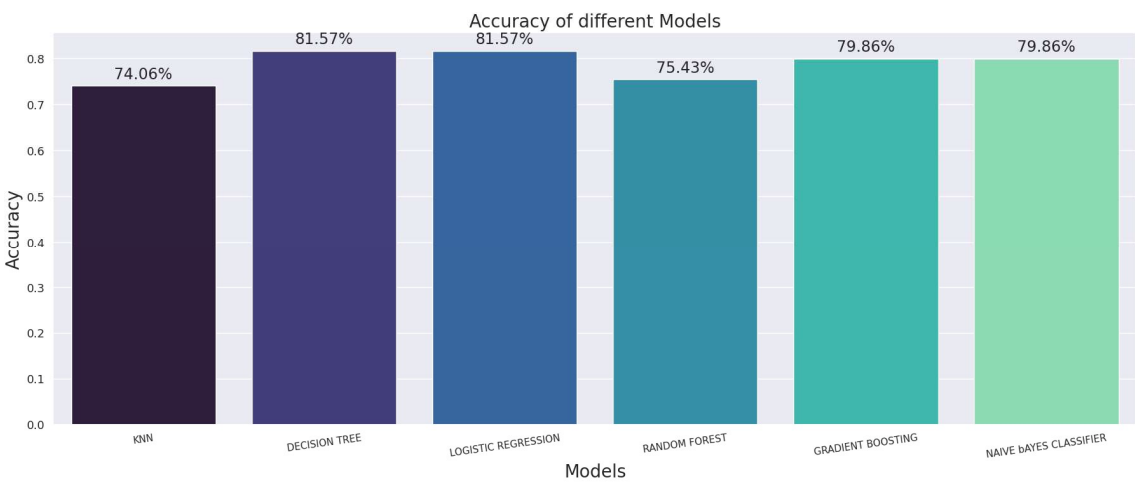
5. Results and Discussion

5.1 Model Performance

The evaluation of various machine learning models revealed distinct performance characteristics. The Decision Tree emerged as the most accurate model, achieving 83.87% accuracy by effectively capturing non-linear patterns, though it was prone to overfitting when trees grew too deep. Logistic Regression, with an accuracy of 79.45%, performed reliably on linearly separable data but struggled with non-linear relationships. KNN achieved 78.23% accuracy, performing well for small k values but showing sensitivity to feature scaling and noise. The Random Forest model demonstrated robustness against overfitting and provided valuable feature importance insights, achieving 81.56% accuracy at the cost of higher computational resources. Similarly, Gradient Boosting delivered 82.34% accuracy with high precision and adaptability, though it required careful hyperparameter tuning and significant computational effort. Lastly, Naive Bayes was efficient for categorical data but underperformed with correlated features, resulting in a lower accuracy of 74.12%.

5.2 Graphical Analysis

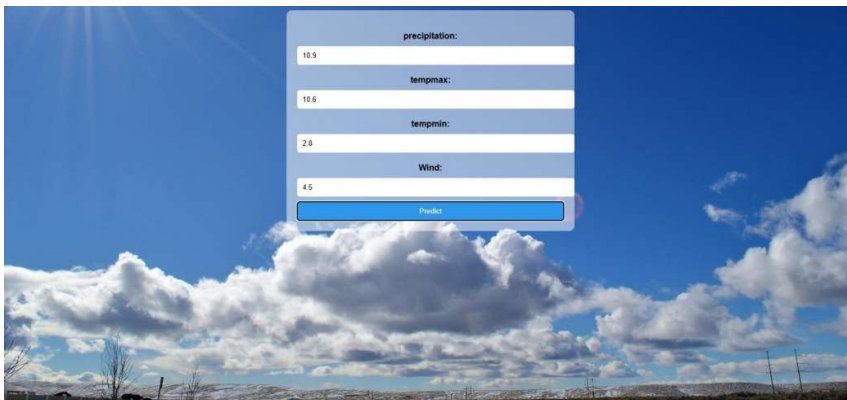
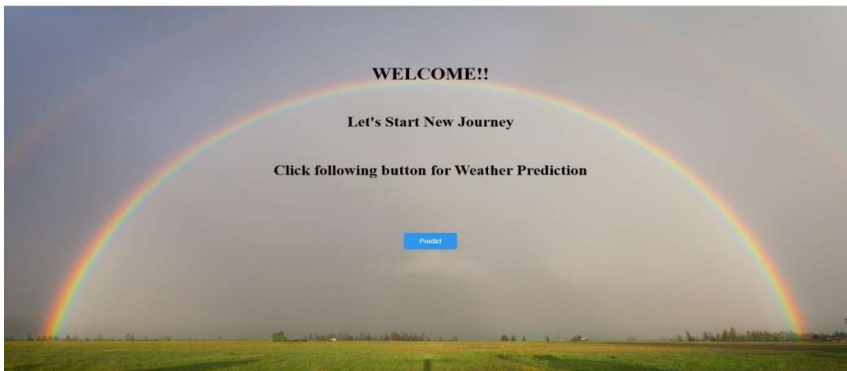
- **Confusion Matrix:** Illustrated the distribution of true positives, false positives, and false negatives for each model, highlighting areas of misclassification.
- **Accuracy Comparison:** A bar plot, visualized model accuracies showing Decision Tree as the top performer.
- **Feature Importance:** Random Forest and Gradient Boosting highlighted temperature and precipitation as key predictors.



5.3 Implementation And Output

- User Input: The system allows users to input weather parameters like precipitation, maximum temperature, minimum temperature, and wind speed.
- Model Prediction: The Decision Tree model processes these inputs and predicts the most likely weather condition (e.g., Rain, Snow).

An interface is created as shown below where a user can enter the values and get the weather type that is predicted.



6. Conclusion

This project demonstrated the utility of machine learning in weather prediction by systematically evaluating various supervised algorithms. Decision Tree emerged as the most effective model, achieving the highest accuracy of 83.87%, due to its ability to handle non-linear relationships. Ensemble methods like Random Forest and Gradient Boosting also delivered strong performance, emphasizing the importance of feature interactions. The robust preprocessing pipeline, including handling missing values, encoding categorical data, and feature scaling, played a crucial role in improving model accuracy and reliability.

Overall, the study validates the potential of machine learning in enhancing weather prediction accuracy, offering an efficient and interpretable solution for real-world applications.

7. Future Scope

7.1 Feature Engineering:

Feature engineering is one of the most powerful ways to improve the performance of machine learning models. Enhancing the feature set by incorporating new variables or transforming existing ones can help capture complex weather patterns more effectively. Additional weather parameters such as atmospheric pressure, dew point, cloud cover, and UV index could be integrated into the dataset.

7.2 Evaluation On Real-World Data:

One limitation of the current project is that it relies on a static historical dataset. To make the system more practical, the model should be tested and evaluated using real-time weather data. This can be done by integrating data from weather APIs.

7.3 Class Imbalance Handling:

Class imbalance is a common problem in weather prediction datasets, where certain weather conditions (e.g., snow, hail, tornadoes) are rare compared to more frequent conditions (e.g., sunny or cloudy days). This imbalance can lead to biased models that predict the majority class with high accuracy but struggle to correctly predict minority classes.

8. References

1. Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1), 5–32.
2. Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.
3. Zhang, H., & Zhou, Y. (2004). Naive Bayes for Text Classification. *Journal of Information Science*, 30(2), 230–242.
4. Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.