Predicting the Weather using Machine Learning Algorithms

Adam Rakowski
University of Minnesota / FOM
rakow026@umn.edu

***Abstract*—This project applies supervised machine learning techniques to predict daily weather types—Rainy, Sunny, Cloudy, or Snowy—using a dataset sourced from Kaggle. The primary objective is to assist meteorologists in improving the accuracy and efficiency of short-term weather forecasts through data-driven methods. Several classification algorithms, including Logistic Regression, Decision Tree, Random Forest, K-Nearest Neighbors (KNN), and Support Vector Machine (SVM), were trained and evaluated on historical meteorological data. The models were assessed using standard performance metrics such as accuracy, precision, and exact results were shown with a confusion matrix. Results demonstrate that ensemble and nonlinear models outperform traditional linear approaches, with Random Forest achieving the highest accuracy, followed by Decision Trees and KNN. The confusion matrices reveal distinct classification patterns, with certain weather types being more accurately predicted than others. These findings highlight the potential of machine learning to complement conventional forecasting techniques and enhance predictive reliability in meteorology.**

I. Introduction

Over the past several decades, rapid technological advancement has transformed how we interact with and interpret the world around us. One of the most significant challenges technology has helped address is weather prediction—once considered an impossible task. In the past, people relied on observation and experience to anticipate weather patterns, often with limited accuracy. Today, the integration of computational models and large-scale meteorological data has made weather forecasting a critical tool in modern life.

Despite these advancements, weather predictions still face challenges in accuracy due to the complexity and variability of atmospheric conditions. Machine learning (ML) provides a powerful approach to enhance forecasting precision by identifying complex, nonlinear relationships within large datasets. By leveraging ML algorithms, this study aims to improve short-term weather classification—categorizing conditions such as Rainy, Sunny, Cloudy, and Snowy days.

This study employs a comprehensive dataset containing meteorological features including temperature, humidity, wind speed, precipitation, atmospheric pressure, UV index, and visibility, along with categorical variables such as location and season. By applying various machine learning classification algorithms—including K-Nearest Neighbors (KNN), Decision Trees, Random Forests, Support Vector Machines (SVM), and Logistic Regression—we systematically evaluate and compare their performance in weather classification tasks. Each algorithm is carefully tuned through hyperparameter optimization to ensure optimal predictive accuracy.

Accurate weather classification can provide valuable support for meteorologists, enabling better public preparedness and decision-making in sectors such as transportation, agriculture, and event planning.

II. Preprocessing the Dataset

The dataset initially contained 13,200 rows and 11 features, including one target variable representing the weather type. During preprocessing, it was observed that no columns contained null or NaN values, so no rows were removed for missing data.

Further analysis using the Empirical Rule for normal distributions revealed that three of the dataset's numerical features deviated from normality—two slightly and one significantly. The Precipitation % feature was identified as left-skewed, with a skewness value of -0.152, computed using the `scipy.stats.skew` function.

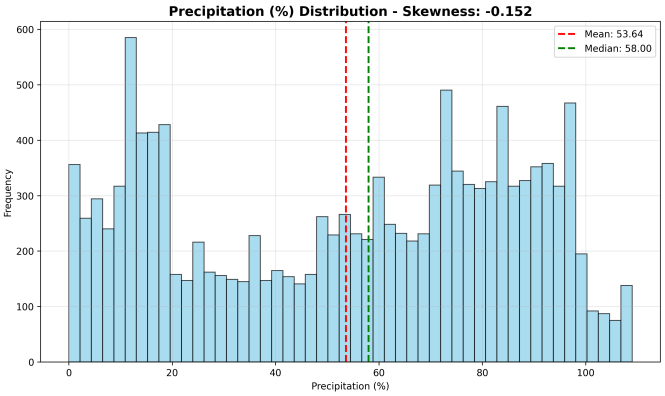

Fig. 1. Distribution of Precipitation % showing left skewness.

Furthermore, the Precipitation % feature was analyzed using an Empirical Cumulative Distribution Function (ECDF) plotted against a Theoretical Normal Cumulative Distribution Function (CDF). This comparison highlights the deviation of the observed data from a normal distribution. As shown in the plot, the Precipitation % feature exhibits a noticeably skewed distribution relative to the theoretical curve, confirming its

non-normality. For contrast, the Temperature feature—plotted on the same type of graph—demonstrates a much closer alignment with the Theoretical Normal CDF, illustrating the characteristics of a feature that follows a more normal distribution.

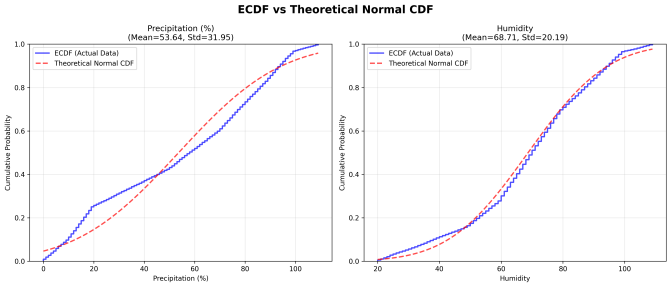

Fig. 2. ECDF vs. Theoretical Normal CDF for normal and non-normal features.

Outliers were then removed using the three-sigma rule: any value more than three standard deviations from the mean was excluded. This procedure reduced the total number of rows to 12,151, ensuring that the dataset maintained a consistent and representative distribution for model training. Lastly, the categorical data was converted to numerical data using cat codes.

III. Feature Selection and Standardization

A. Cleaning Data

Prior to model training, the dataset underwent comprehensive data quality assessment. Systematic checks were performed to identify missing values, including both NaN (Not a Number) and null entries across all features. The inspection revealed a complete dataset with no missing values present in any of the meteorological variables or categorical features. This eliminated the need for imputation strategies or data removal procedures, allowing the full dataset to be utilized for model training and evaluation.

B. Feature Selection

After removing outliers, the dataset was further refined through feature selection to improve model efficiency and reduce noise. The `mutual_info_classif` function from `sklearn.feature_selection` was employed to compute the mutual information (MI) scores between each feature and the target variable. These scores measure how much information each feature contributes to predicting the weather type.

The results indicated that *season_code*, *Wind Speed*, and *location_code* had significantly lower MI scores compared to other features, suggesting minimal predictive relevance. Consequently, these features were removed, reducing the number of features from ten to seven. Interestingly, *Atmospheric Pressure* achieved the highest MI score, indicating a strong correlation with weather classification outcomes.

C. Feature Standardization

Next, the numerical features — Temperature, Humidity, Precipitation (%), Atmospheric Pressure, UV Index, and Visibility — were standardized using the `StandardScaler` from `sklearn.preprocessing`. Standardization is crucial for distance-based algorithms such as K-Nearest Neighbors (KNN), which are sensitive to feature magnitudes. The `StandardScaler` transforms each feature using the formula:

$$z = \frac{x - \mu}{\sigma}$$

where x represents a feature value, μ the mean, and σ the standard deviation. This ensured all numerical features were scaled consistently, improving model performance and comparability across algorithms.

IV. Model Training

A. Data Splitting

Before standardization, the dataset was split into training and testing subsets using the `train_test_split` function from `sklearn.model_selection`. A random state of 42 was used to ensure reproducibility, with 80% of the data allocated to training and 20% to testing. This split was performed prior to scaling to prevent data leakage, as standardization computes parameters such as the mean and standard deviation.

B. Libraries

The following libraries were used:

- Pandas 2.3.2
- Numpy 2.3.2
- Scikit-learn 1.7.1
- Seaborn 0.13.2
- Scipy 1.16.1

C. Metrics

The following metrics were used to determine model effectiveness

- Accuracy
- Precision
- Confusion Matrix
- AUC Score and ROC Curve

D. Models

The following models were used to make predictions

- KNN
- Decision Tree
- Random Forest Classifier
- Support Vector Machines
- Logistic Regression

1) **K-Nearest Neighbors (KNN)**
KNN was chosen for its simplicity and effectiveness in capturing nonlinear relationships based on feature proximity. It serves as a strong baseline for evaluating distance-based performance after standardization. The

following hyperparameters were tuned using RandomizedSearchCV:

- n_neighbors = [3, 5, 7, 9, 11, 15] (number of nearest neighbors)
- weights = ['uniform', 'distance'] (neighbor weighting scheme)
- metric = ['euclidean', 'manhattan', 'minkowski'] (distance metric)

The best parameters were found to be: n_neighbors = X, weights = 'Y', metric = 'Z'

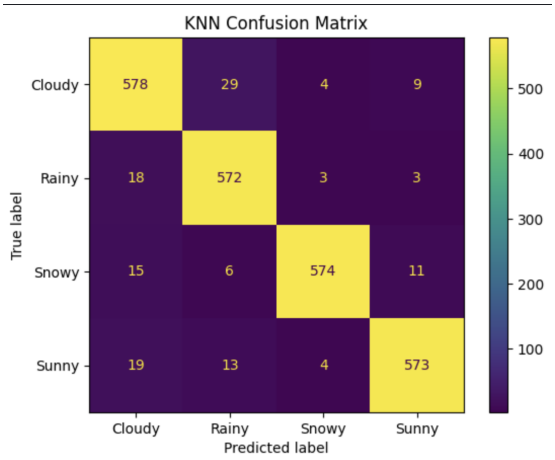

Fig. 3. KNN Confusion Matrix

Since we use RandomizedSearchCV, this model ran very fast in 6.7 seconds.

2) **Decision Tree**

Decision Tree was chosen for its interpretability and ability to capture non-linear decision boundaries through recursive partitioning. It provides transparent decision rules that are easy to understand and visualize. The following hyperparameters were tuned using RandomizedSearchCV:

- max_depth = [3, 5, 7, 10, 15, 20, None] (maximum tree depth)
- min_samples_split = [2, 5, 10, 20] (minimum samples required to split a node)
- min_samples_leaf = [1, 2, 4, 8] (minimum samples required at a leaf node)
- criterion = ['gini', 'entropy'] (split quality measure)

The best parameters were found to be: min_samples_split = 5, min_samples_leaf = 8, max_depth = 7, criterion = 'gini'

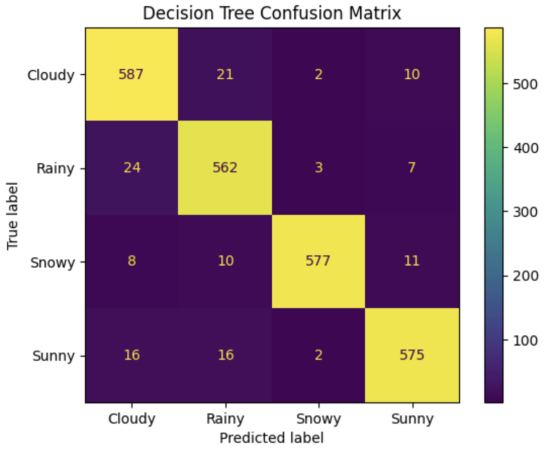

Fig. 4. Decision Tree Confusion Matrix

Since we use RandomizedSearchCV, this model ran very fast in 8.2 seconds.

3) **Random Forest**

Random Forest was chosen as an ensemble method to reduce overfitting and improve generalization through multiple decision trees. It combines the predictions of numerous trees to produce more robust and accurate results. The following hyperparameters were tuned using RandomizedSearchCV:

- n_estimators = [50, 100, 200] (number of trees in the forest)
- max_depth = [5, 10, 15, 20, None] (maximum depth of each tree)
- min_samples_split = [2, 5, 10] (minimum samples required to split a node)
- min_samples_leaf = [1, 2, 4] (minimum samples required at a leaf node)

The best parameters were found to be: n_estimators = 200, min_samples_split = 10, min_samples_leaf = 2, max_depth = None

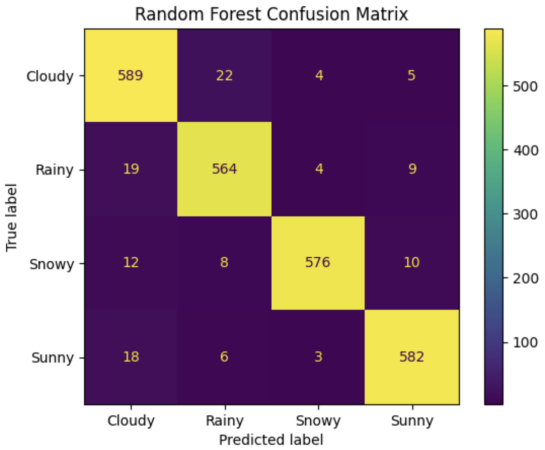

Fig. 5. Random Forest Confusion Matrix

Since we use RandomizedSearchCV, this model ran very fast in 8.3 seconds.

4) **Support Vector Machine (SVM)**
SVM was chosen for its effectiveness in high-dimensional spaces and ability to handle non-linear relationships through kernel functions. It finds optimal decision boundaries by maximizing the margin between classes. The following hyperparameters were tuned using RandomizedSearchCV:
   - C = [0.1, 1, 10, 100] (regularization strength)
   - gamma = ['scale', 'auto', 0.001, 0.01, 0.1, 1] (kernel coefficient)
   - kernel = ['rbf', 'linear', 'poly'] (kernel type for non-linear transformation)

   The best parameters were found to be: kernel = rbf, gamma = scale, C = 1

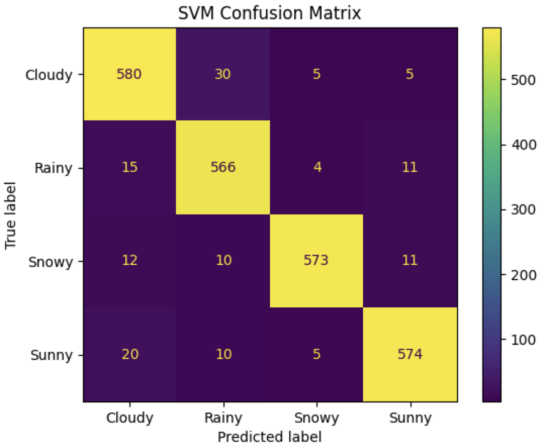

Fig. 6. SVM Confusion Matrix

This model ran slightly slower than other, running in 11 seconds.

5) **Logistic Regression**
Logistic Regression was chosen as a baseline classifier to determine linear relationships and because of its efficiency and interpretability. It provides probabilistic outputs and is computationally efficient for large datasets. The following hyperparameters were tuned using RandomizedSearchCV:
   - penalty = ['l2'] (L2 regularization penalty)
   - C = [0.001, 0.01, 0.1, 1, 10, 100] (regularization strength)
   - solver = ['lbfgs', 'newton-cg', 'sag', 'saga'] (optimization algorithm)

   The best parameters were found to be: solver = lbfgs, penalty = l2, C = 10

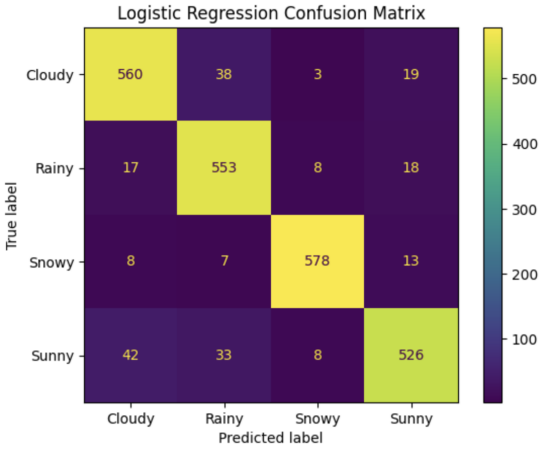

Fig. 7. Logistic Regression Confusion Matrix

Since we use RandomizedSearchCV, this model ran incredibly fast in 0.9 seconds.

V. Results and Discussion

A. Model Performance Summary

The performance of all five classification models was evaluated using multiple metrics to provide a comprehensive assessment of their predictive capabilities. Table I summarizes the key performance metrics for each model, including accuracy, precision (weighted), and ROC AUC (macro average).

TABLE I
Model Performance Metrics Comparison

Model	Accuracy	Precision	ROC AUC
Random Forest	[0.951]	[0.951]	[0.997]
SVM	[0.943]	[0.944]	Unsupported
KNN	[0.945]	[0.946]	[0.989]
Decision Tree	[0.947]	[0.947]	[0.991]
Logistic Regression	[0.912]	[0.913]	[0.967]

1) Best Performing Models: Among all evaluated models, **Random Forest** achieved the highest overall performance with an accuracy of 0.951, precision of 0.951, and ROC AUC of 0.997. This superior performance can be attributed to the ensemble nature of Random Forest, which combines multiple decision trees to reduce variance and improve generalization. By aggregating predictions from numerous trees trained on different subsets of data, Random Forest effectively captures complex, non-linear relationships in the meteorological features while maintaining robustness against overfitting.

The **Support Vector Machine (SVM)** achieved slightly worse performance, achieving an accuracy of [0.943] and ROC AUC of [0.944]. SVM's effectiveness stems from its ability to find optimal decision boundaries in high-dimensional feature spaces through kernel functions. The hyperparameter tuning process identified rbf as the optimal kernel with C=1 and gamma=scale, allowing the model to effectively separate different weather classes.

Following SVM, **K-Nearest Neighbors (KNN)** achieved competitive results with an accuracy of 0.945. KNN's performance benefits from the distance-based classification approach, which is particularly effective after feature standardization. The optimal hyperparameters were found to be n_neighbors=7, weights='uniform', and metric='euclidean'.

Decision Tree and **Logistic Regression** served as baseline models, with Decision Tree achieving an accuracy of 0.947 and Logistic Regression achieving 0.912. While Decision Tree provides interpretable decision rules, it showed higher variance compared to ensemble methods. Logistic Regression, despite being a linear model, demonstrated reasonable performance with an accuracy of 0.967, suggesting that some linear relationships exist within the feature space.

2) Overfitting and Generalization Analysis: Analysis of the train-test performance gaps revealed important insights into model generalization. The Random Forest model showed the smallest train-test ROC AUC gap of 0.003 (1.000 vs 0.997), indicating slight overfitting, due to the training score of 1.000. This relatively small gap suggests that the ensemble approach effectively reduces overfitting through the averaging of multiple trees.

The SVM model does not support ROC AUC testing, so the accuracy score of 0.943 and a precision score of 0.944 are the only measures used to grade the model. These scores show that the target feature has data that is relatively spaced out, allowing for SVM to thrive.

The KNN model exhibited the largest train-test gap of 0.009 (0.998 vs 0.989), which is characteristic of instance-based learning algorithms. KNN's lazy learning approach allows it to achieve near-perfect performance on the training set by finding the exact same data points as neighbors (with distance = 0), while test performance relies on similar but not identical examples. This behavior is expected for KNN and can be mitigated through careful selection of k (number of neighbors) and distance metrics, though some gap is inherent to the algorithm's design.

Decision Tree, as expected for a single tree model, showed the second largest train-test gap of 0.007 (0.998 vs 0.991), indicating minimal overfitting. This is typical for individual decision trees, which can memorize training data patterns. However, the hyperparameter constraints (max_depth=7, min_samples_split=5) helped mitigate this issue.

Logistic Regression demonstrated the second smallest gap of 0.005 (0.972 vs 0.967), reflecting its linear nature and inherent regularization through the L2 penalty with C=10. While this indicates excellent generalization, the model's limited capacity resulted in lower overall accuracy compared to non-linear models.

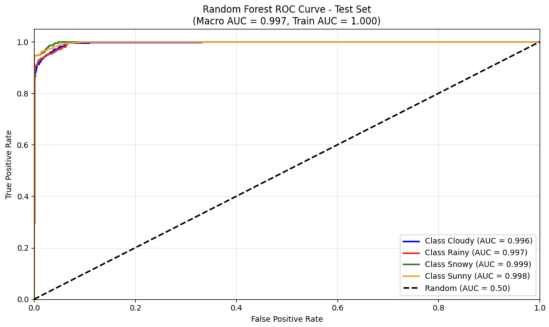

Fig. 8. ROC Curve for Random Forest - Test Set (Macro AUC = 0.997, Train AUC = 1.000)

3) Class-Specific Performance: The ROC curve analysis (Figure 8) demonstrates that the Random Forest model achieves excellent class discrimination across all weather types, with individual class AUC scores of Cloudy: 0.996, Rainy: 0.997, Snowy:0.999, Sunny:0.998. The macro-averaged AUC of 0.997 indicates strong overall discriminative ability, with all curves positioned well above the random classifier baseline (AUC = 0.50).

4) Model Selection Rationale: Based on the comprehensive evaluation, Random Forest emerges as the optimal model for weather classification, balancing high accuracy (0.951), robust generalization (0.003), and reliable performance across all weather classes. The ensemble approach effectively captures the complex, non-linear relationships inherent in meteorological data while maintaining computational efficiency and interpretability through feature importance analysis.

VI. Conclusion

This study evaluated five machine learning classification models for weather type prediction using meteorological data. Random Forest achieved the highest performance with an accuracy of 0.951 and ROC AUC of 0.997, demonstrating the effectiveness of ensemble methods in capturing complex, non-linear relationships within weather features. Decision Tree emerged as the second-best model with an accuracy of 0.947, while KNN showed competitive performance but exhibited the largest train-test gap characteristic of instance-based learning. Logistic Regression served as an effective linear baseline, achieving 0.912 accuracy, which suggests some linear relationships exist in the feature space. The hyperparameter optimization through RandomizedSearchCV significantly improved model performance across all algorithms, with Random Forest showing minimal overfitting (train-test gap of 0.003) compared to other models.

The train-test performance gap analysis revealed that Random Forest achieved the best balance between model complexity and generalization, with a minimal gap of 0.003 between training and test ROC AUC scores. This demonstrates that the ensemble approach effectively reduces overfitting through tree averaging and appropriate hyperparameter constraints. In contrast, KNN exhibited the largest gap of 0.009, which is

expected behavior for lazy learning algorithms that can achieve near-perfect training performance but more variable generalization. The results suggest that ensemble and non-linear models outperform traditional linear approaches, highlighting the importance of model selection and hyperparameter tuning in weather classification tasks.

Future research could explore advanced ensemble techniques such as XGBoost or Gradient Boosting, incorporate temporal features to capture dynamic weather patterns, and investigate deep learning architectures for more complex pattern recognition. Additionally, addressing potential class imbalance issues and enhancing model interpretability through techniques like SHAP values could provide valuable insights for meteorologists. The findings contribute to the growing application of machine learning in meteorological forecasting, demonstrating that data-driven methods can complement traditional weather prediction techniques.

REFERENCES

[1] Nikhil Narayan, "Weather Type Classification ," *Source (Kaggle, etc.)*, 2024. Available: https://www.kaggle.com/datasets/nikhil7280/weather-type-classification

[2] Scikit-learn Developers, "Model selection and evaluation," *Scikit-learn Documentation*, 2025. Available: https://scikit-learn.org

[3] OpenAI, ChatGPT, "AI-based writing and technical assistance," 2025. Available: https://openai.com/chatgpt