**Weather Type Classification**

Yucheng Ma

School of Computer Science

https://github.com/markfromcd/Weather-Type-Classification.git

**1. Introduction**

Weather predictions impact a wide range of industries and everyday life decisions. For instance, real-world applicability, safety and disaster preparedness, environmental protection. In detail, accurate weather predictions are crucial for agriculture as farmers plan planting, irrigation, and harvesting schedules to optimize crop growth and minimize risks; in transportation, where airlines, shipping companies, and logistics firms plan routes to ensure safety and efficiency; and in retail, where businesses adjust inventory and marketing strategies based on forecasted weather conditions, such as stocking more umbrellas and raincoats in anticipation of rain. As a result, this is a good and useful problem to be developed and solved.

The dataset contains temperature, humidity, wind speed, precipitation percentage, cloud cover, atmospheric pressure, UV index, season, visibility, and location, the objective is to predict the weather type (e.g., Rainy, Cloudy, Sunny, Snowy). This is a multiclass classification problem, where each weather type is a discrete category.1

A highly rated previous work2 used SVC, decision tree and random forest to build models and train the dataset. The accuracies turned out to be 0.8745, 0.8981, 0.9082. His best model is random forest with no specific parameters. After having my models, the best model score turned out to be 0.9129, which is a little better than previous work.

**2. EDA**

**2.1 Data frame of the dataset:**

* There are 13, 200 data points in total, with 11 original columns.
* The target feature is uniformly distributed across 'Rainy', 'Cloudy', 'Sunny', and 'Snowy', each with 3300 instances.
* ‘Weather Type’, 'Location', 'season', and 'cloud cover' are categorical features, while the remaining features are continuous.
* There are no missing values.
* I also checked the number of unique values in each feature to determine if any of them are needed to be removed. However, only 'Atmospheric Pressure' has too many values (5456). But since it behaves differently under various weather conditions, it should not be removed. The figure 1 is shown below.

A graph with text on it

Description automatically generated

Figure 1: Number of Unique Values in Each Column

**2.2 Findings among features:**

* The distribution of categorical features is as follows (Figure 2):

A graph showing different seasons

Description automatically generated

Figure 2: The distribution of categorical features

* Categorical Features vs. Weather Type. Figure 3 illustrates how weather types vary significantly with cloud cover, season, and location, indicating clear, overcast, and partly cloudy skies are predominantly sunny, while seasonal and geographical differences profoundly impact the prevalence of sunny, rainy, snowy, and cloudy weather.

A graph with different colored bars

Description automatically generated

Figure 3: Categorical Features vs. Weather Type

* Continuous features vs. Weather Type. Figure 4 below shows some information about continuous features. Sunny weather typically has higher temperatures and UV index values, while snowy weather shows the lowest in both categories, indicating the intensity of sunlight and ambient temperature variations with weather conditions. Rainy weather is characterized by high humidity and precipitation levels, which decrease significantly in sunny conditions and are even lower in snowy weather, highlighting the moisture content associated with different weather types. Rainy and snowy weather is associated with higher wind speeds and reduced visibility compared to other weather types.

A group of boxes with different colored squares

Description automatically generated

Figure 4: Continuous Features vs. Weather Type

**3. Methods**

**3.1 Splitting**

I used ‘StratifiedShuffleSplit’ and ‘StratifiedKFold’ to split the dataset. First, the dataset is split into other (80%) for training and validation and testing (20%) by stratified shuffle split to ensure the class distribution is preserved in both subsets. Additionally, Stratified K-Fold, a cross-validation strategy, is used during model training to assess performance consistently across multiple splits of the training data by using each split as validation set once.

**3.2 Preprocessing**

* For continuous features, I used ‘StandarScaler’ to to ensure they have a mean of 0 and a standard deviation of 1. This step is critical for algorithms sensitive to feature scaling.
* For categorical features, I used ‘One-Hot Encoding’ to transform ‘Location’ as it is a string feature with no correlation with values. I used ‘Ordinal Encoding’ to transform ‘Season’ and ‘Cloud Cover’ as these two features have inherent order with their values.
* Since I used XGBoost, I translated target variables using label encoding.
* I also checked missing values, but there does not contain any of them. After that, I combined a column transformer pipeline to preprocess dataset.

**3.3 Machine Learning Pipeline**

* Preprocessing
* Stratified K-Fold Cross-Validation
* ParamsTuning (GridSearchCV)
* Testing
* Store scores and find the best model with best parameters

**3.4 Metric for evaluating model’s performance**

Since the class distribution in the dataset is relatively balanced, **accuracy** provides a straightforward measure of overall model performance. For imbalanced datasets, other metrics such as F1-Score, precision, and recall could be used. I also checked the F1-Score showing the similar situation as accuracy.

**3.5 Algorithms**

I tried **Logistic Regression**, **Support Vector Machines (SVM)**, **Random Forest**. **KNN** and **XGBoost,** and the parameters and the best combinations I tuned as shown as follows (Figure 5):

|  |  |
| --- | --- |
| Algorithms | Parameterss |
| Logistic Regression | C = [0.01, 0.1, 1, 10, 100]  penalty = ['l2']  solver = ['lbfgs', 'sag', 'newton-cg'] |
| Random Forest | max\_depth = [5, 10, None]  max\_features = ['sqrt', 'log2', None]  min\_samples\_split = [2, 5, 10] |
| SVM | C = [0.01, 0.1, 1, 10]  gamma = [0.001, 0.01, 0.1, 1, 10, 100]  kernel = ['rbf', 'sigmoid'] |
| XGBoost | learning\_rate = [0.01, 0.1, 0.3]  reg\_lambda = [0e0, 1e-2, 1e-1, 1e0, 1e1, 1e2]  max\_depth = [3, 6, 9] |
| KNN | n\_neighbors = [3, 5, 7, 10]  weights = ['uniform', 'distance']  metric = ['euclidean', 'manhattan', 'minkowski'] |

Figure 5: Algorithms and Parameters

**3.6 Considerations**

* Splitting Strategy: Stratification was crucial to maintain class balance and multiple random states were used to ensure robustness.
* Preprocessing: Proper handling of categorical and numerical features ensures compatibility with all models.
* Algorithm Selection: A mix of linear and non-linear models was used to compare performance.
* Non-deterministic Algorithms: like randomized algorithms, for example, Random Forest and XGBoost, introduce stochasticity so that multiple runs are averaged to account for variability.
* Hyperparameter Tuning: Parameters were chosen based on their potential impact on model performance and computational cost.

**4 Results**

As I said, this dataset is nicely balanced by target feature, so I used Accuracy as my score. And the baseline score is 0.25. The comparison of models with baseline score is shown below in Figure 6.

Each score of models I trained is better than baseline. The best one is XGBoost (0.9129), which is slightly better than Random Forest (0.9128).

A graph with numbers and points

Description automatically generated with medium confidence

Figure 6: Model Comparison on Test Accuracy

The best combination and score of each model are shown in Figure 7.

|  |  |  |
| --- | --- | --- |
| Algorithms | Best Combinations | Best Accuracy |
| Logistic Regression | C = 0.1  Penalty = l2  Solver = lbfgs | 0.8659 |
| Random Forest | max\_depth = None  Max\_features = sqrt  Min\_sample\_split = 10 | 0.9128 |
| SVM | C = 1  gamma = 0.1  Kernel = rbf | 0.9057 |
| XGBoost | learning\_rate = 0.3  reg\_lambda = 0.01  Max\_depth = 9 | 0.9129 |
| KNN | n\_neighbors = 7  weights = uniform  metric = manhattan | 0.8942 |

Figure 7: Best Combination and Score of Each Model

1. https://www.kaggle.com/datasets/nikhil7280/weather-type-classification/data
2. https://www.kaggle.com/code/ihabsherbiny/weather-classification-with-3-models#Decision-Tree-Classifier