### Documentation for HumidityPredictor Class: A Machine Learning Approach for Humidity Prediction

This document provides a comprehensive overview of the HumidityPredictor class, a machine learning pipeline designed to predict humidity levels based on weather data. This documentation covers the following sections:

1. **Overview**
2. **Algorithms Used**
3. **Class Methods and Implementation**
4. **Instructions for Running the Script**
5. **Conclusion**

### 1. Overview

The HumidityPredictor class is a structured machine learning workflow for classifying weather data based on humidity levels. Given a weather dataset, it applies preprocessing, feature engineering, model training, evaluation, and prediction steps, with the goal of accurately classifying humidity into two categories: low (below 25%) and high (above 25%).

### 2. Algorithms Used

The HumidityPredictor class employs six popular machine learning algorithms:

* **Decision Tree**: A tree-based algorithm that recursively splits the data based on feature values to create a model that classifies data points into predefined categories.
* **Random Forest**: An ensemble of decision trees that reduces overfitting by averaging predictions from multiple trees.
* **Gradient Boosting**: An ensemble method that builds models sequentially, correcting errors from previous models to improve accuracy.
* **Support Vector Machine (SVM)**: A classification model that seeks an optimal hyperplane to separate classes in high-dimensional space.
* **K-Nearest Neighbors (KNN)**: A model that classifies data points based on the labels of their closest neighbors in the feature space.
* **Logistic Regression**: A statistical model that predicts the probability of a binary outcome using a logistic function.

### 3. Class Methods and Implementation

Each method in the HumidityPredictor class is designed to handle a specific task in the prediction pipeline. Below is a summary of the key methods and their purposes:

#### \_\_init\_\_

* Initializes the class by setting up the models, a standard scaler for feature scaling, and configuring logging.

#### load\_data(file\_path: str) -> pd.DataFrame

* Loads the dataset from a CSV file.
* Logs a message indicating successful loading or an error if loading fails.

#### preprocess\_data(data: pd.DataFrame) -> pd.DataFrame

* Preprocesses data by:
  + Removing unnecessary columns.
  + Filling missing values with median values.
  + Handling outliers using the Interquartile Range (IQR) method.
* Returns a cleaned and processed DataFrame.

#### create\_target(data: pd.DataFrame) -> Tuple[pd.DataFrame, pd.Series]

* Creates a binary target variable, humidity\_class, based on the humidity threshold of 25%.
* Returns the feature set X and target variable y.

#### train\_models(X: pd.DataFrame, y: pd.Series) -> Dict

* Splits the data into training and testing sets.
* Scales the feature data.
* Trains each model and evaluates it using:
  + Test set accuracy.
  + Cross-validation (CV) scores.
  + Confusion matrix and classification report.
* Logs model performance metrics and returns results for further analysis.

#### plot\_results(results: Dict)

* Plots test and CV accuracy scores for each model.
* Displays confusion matrices to visualize model performance on the test set.

#### predict\_future\_humidity(X\_new: pd.DataFrame, model\_name: str = 'Random Forest') -> pd.DataFrame

* Uses a trained model (default: Random Forest) to predict humidity classes for new data.
* Scales the input data and returns predictions with probability scores and an interpretation for each predicted class.

### 4. Instructions for Running the Script

#### Prerequisites

To run this code, install the following libraries if not already installed:

pip install pandas numpy scikit-learn matplotlib seaborn

#### Dataset

* Place your weather dataset in the same directory or specify the correct path in the file\_path argument of load\_data.
* The dataset should contain a column relative\_humidity\_3pm and any other relevant weather features.

#### Running the Script

* Run the script by executing main(), which performs the following steps:
  1. Initializes the HumidityPredictor object.
  2. Loads and preprocesses the data.
  3. Creates features and the target variable.
  4. Trains and evaluates each model.
  5. Plots the model performance metrics.
  6. Provides an example prediction on new data.

#### Example Usage

main()

### 5. Conclusion

The HumidityPredictor class provides a flexible and modular approach for predicting humidity levels using several machine learning algorithms. It offers insights into model performance through logging and visualization and enables easy prediction on new data. By comparing algorithms such as Decision Tree, Random Forest, and Gradient Boosting, users can identify the model best suited for their specific dataset and prediction needs.