Report on Naive Bayes and Logistic Regression for Demo Weather Dataset

In this report, we present the implementation and evaluation of Naive Bayes and Logistic Regression algorithms on the Demo Weather Dataset. The goal is to classify weather conditions based on a set of features. We will discuss the code structure, algorithms used, data preprocessing steps, and the results obtained.

◆ Code Structure:

1. **`naive_bayes.py`**: Contains the **`NaiveBayes`** class that implements the Naive Bayes algorithm. It provides methods for calculating posterior probabilities and classifying instances.
2. **`logistic_regression.py`**: Includes the **`BinaryLogisticRegression`** and `MulticlassLogisticRegression` classes. **`BinaryLogisticRegression`** implements binary logistic regression, while **`MulticlassLogisticRegression`** extends it for multiclass classification. These classes handle model training and prediction.
3. **`experiment.py`**: Defines the `Experiment` class, which performs data preprocessing and evaluation tasks. It includes methods for loading data from CSV files, calculating class priors, likelihoods, and accuracy.
4. **`results.py`**: Demonstrates how to use the implemented algorithms. It includes functions for testing Naive Bayes and Logistic Regression on the dataset and visualizing the results using matplotlib.

◆ Data Preprocessing:

1. Loading CSV: The `Experiment` class provides a method to load data from CSV files, which returns a list of rows.
2. Preprocessing Data: The `preprocess_data` method in `Experiment` class separates the features and labels from the loaded data.
3. Encoding Features and Labels: The `one_hot_encode` method converts categorical features into one-hot encoded vectors. Labels are encoded as well.
4. Calculating Class Priors: The `calculate_class_priors` method in `Experiment` class calculates the prior probabilities of each class based on the labels.

◆ Naive Bayes:

The Naive Bayes algorithm is implemented in the `NaiveBayes` class in `naive_bayes.py`. It calculates posterior probabilities for each class and classifies instances based on these probabilities.

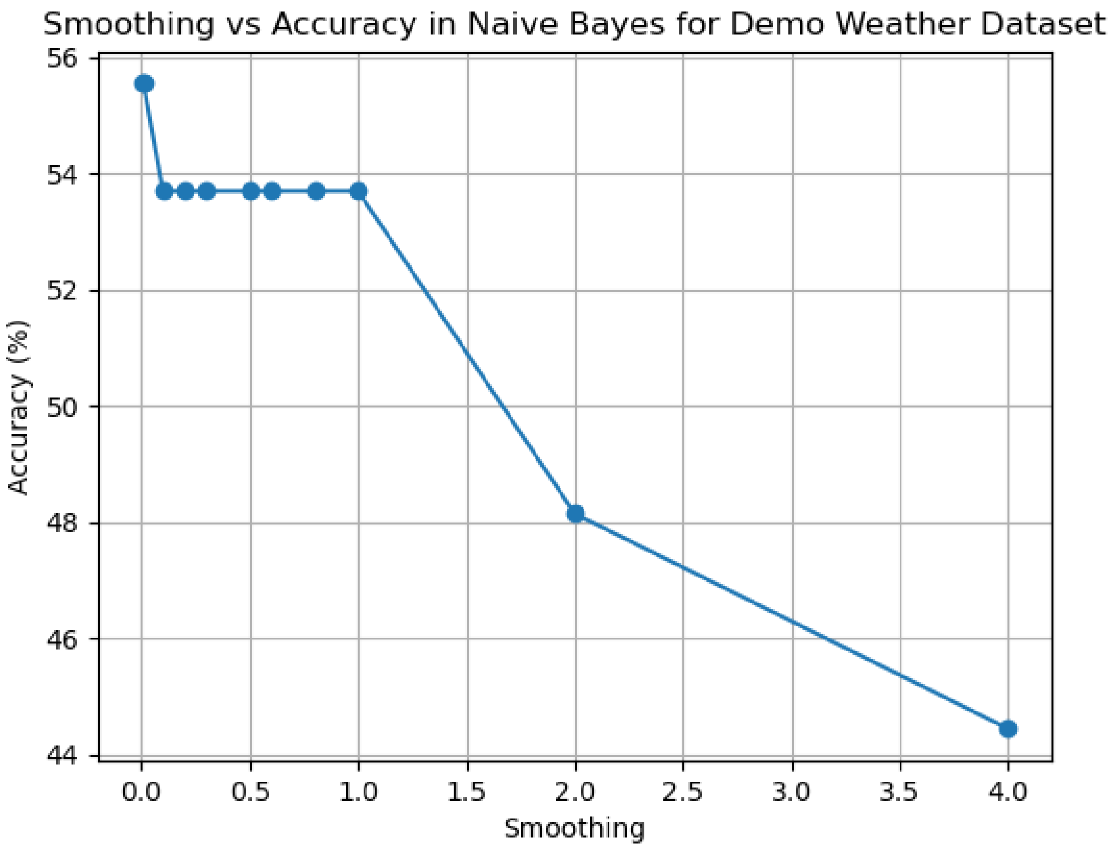
◆ Logistic Regression:

The Logistic Regression algorithm is implemented in the **`BinaryLogisticRegression`** and **`MulticlassLogisticRegression`** classes in **`logistic_regression.py`**. **`BinaryLogisticRegression`** handles binary classification, while `MulticlassLogisticRegression` extends it for multiclass classification. These classes perform model training using gradient descent and make predictions based on the learned parameters.

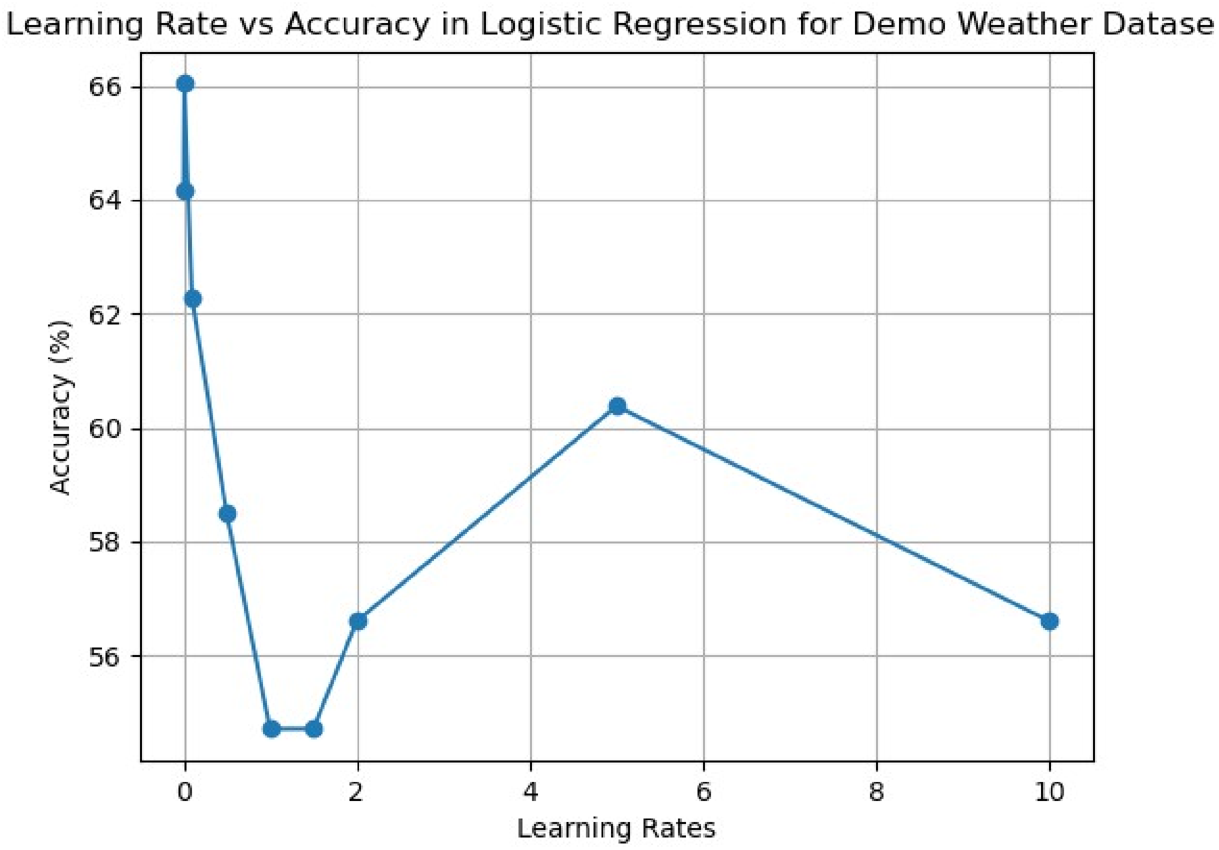
◆ Evaluation:

The `results.py` file demonstrates how to use the implemented algorithms for evaluation. It includes two functions: `try_naive_bayes` and `try_logistic`.

- **`try_naive_bayes`**: Loads the train and test data, calculates class priors, and performs Naive Bayes classification with different smoothing parameters. The accuracy of the predictions is calculated and plotted against the smoothing parameter. A sample plot is given below:



- `try_logistic`: Loads the train and test data, encodes features and labels, trains a Logistic Regression model with different learning rates, and calculates the accuracy of the predictions. The accuracy is plotted against the learning rates. Sample plot is given below:



◆ **Results**:

Based on the provided code, the accuracy achieved on the Demo Weather Dataset is 55.55% for Naive Bayes and 66.2% for Logistic Regression. These accuracy values are relatively low, due to the small size of the provided dataset. It's important to note that the performance of these algorithms highly depends on the quality and quantity of the data. With larger and more diverse datasets, the accuracy of both algorithms can potentially improve.

In conclusion, the implemented Naive Bayes and Logistic Regression algorithms provide a basis for classifying weather conditions in the Demo Weather Dataset. Further improvements can be made if more extensive datasets are provided.