```
import numpy as np

# Generate dummy weather data (replace this with your actual data loading)
def generate_dummy_data(num_samples, num_features):
    return np.random.rand(num_samples, num_features)

def generate_dummy_labels(num_samples):
    return np.random.rand(num_samples)

# Generate dummy data
num_samples = 1000
num_features = 5
features = generate_dummy_data(num_samples, num_features)
labels = generate_dummy_labels(num_samples)

# Split data into training and testing sets
split_index = int(0.8 * num_samples)
train_features, test_features = features[:split_index], features[split_index:]
train_labels, test_labels = labels[:split_index], labels[split_index:]

# Define a simple neural network class
class NeuralNetwork:
    def __init__(self, num_features, num_hidden_units=64):
        self.num_features = num_features
        self.num_hidden_units = num_hidden_units
        self.weights_input_hidden = np.random.randn(num_features, num_hidden_units)
```

```
        self.weights_hidden_output = np.random.randn(num_hidden_units)

    def sigmoid(self, x):
        return 1 / (1 + np.exp(-x))

    def forward_pass(self, inputs):
        hidden_layer = self.sigmoid(np.dot(inputs, self.weights_input_hidden))
        output = self.sigmoid(np.dot(hidden_layer, self.weights_hidden_output))
        return output

    def train(self, X_train, y_train, learning_rate=0.01, epochs=10):
        for epoch in range(epochs):
            for i in range(len(X_train)):
                inputs = X_train[i]
                target = y_train[i]

                # Forward pass
                hidden_layer = self.sigmoid(np.dot(inputs, self.weights_input_hidden))
                output = self.sigmoid(np.dot(hidden_layer, self.weights_hidden_output))

                # Backpropagation (omitted for simplicity)

# Train the neural network
nn = NeuralNetwork(num_features)
nn.train(train_features, train_labels)
```

```
# Evaluate the neural network
predictions_nn = np.array([nn.forward_pass(inputs) for inputs in test_features])
mse_nn = np.mean((predictions_nn - test_labels) ** 2)
print("Neural Network Mean Squared Error:", mse_nn)

# Implement a simple decision tree ensemble method
# (You can implement a simpler version of decision tree or ensemble method here)
```