Water Potability ML Problem

SOLUTION

Introduction

Water potability prediction is a fundamental task for ensuring access to clean and safe drinking water. In this problem, we are given a dataset containing water quality parameters, and the objective is to predict whether the water is potable (safe to drink) or not, based on these parameters. The target variable, `Potability`, indicates whether the water sample is potable (1) or non-potable (0).

This task involves applying techniques in data preprocessing, feature extraction, model training, and evaluation. In this article, we will walk through the steps of solving the water potability prediction problem using machine learning methods, including Random Forest Classifier.

Dataset Overview

The dataset provided consists of several features (input variables) representing different water quality metrics, such as pH, hardness, conductivity, and other chemical properties. These features are used to predict whether the water is potable or not. The key attributes in the dataset are:

1. **Features (X):**
    - Water quality parameters like `ph`, `hardness`, `solids`, `chloramines`, `sulfate`, `conductivity`, `organic_carbon`, `density`, `pH`, etc.
    - These are continuous numeric values, though some may contain invalid characters or missing values.
2. **Target (y):**
    - `Potability`: A binary variable (0 or 1) indicating whether the water is potable or non-potable.

Steps to Solve the Problem

Step 1: Data Loading

The first step is to load the dataset into memory using Python's Pandas library. You can load the dataset from CSV files, where the `train.csv` contains training data, and `test.csv` contains test data.

```
import pandas as pd

train_df = pd.read_csv("train.csv")

test_df = pd.read_csv("test.csv")
```

Step 2: Data Cleaning and Preprocessing

- **Handle Missing Values:** Missing values are common in real-world datasets. In this case, we handle missing values by filling them with the median value of each numeric feature. This helps preserve the distribution of the data.

  ```
  numeric_features =
  train_df.select_dtypes(include=np.number).columns
  .tolist()

  train_df[numeric_features] =
  train_df[numeric_features].fillna(train_df[numeri
  c_features].median())

  test_df[numeric_features] =
  test_df[numeric_features].fillna(train_df[numeric
  _features].median())
  ```

- **Feature Extraction and Transformation:** Some features might contain non-numeric characters or text. It is crucial to clean these fields by using regular expressions to extract the numeric values, such as `ph`, from the string.

  ```
  import re
  ```

```
def extract_numeric(text):
    match = re.search(r"(\d+\.?\d*)", str(text))
  if match:
      return float(match.group(1))
    else:
      return np.nan
train_df["ph"] =
train_df["ph"].apply(extract_numeric)
test_df["ph"] =
test_df["ph"].apply(extract_numeric)
```

Step 3: Target Variable Encoding

If the target variable (`Potability`) is continuous (in some datasets, it may be a float), you need to convert it into a binary format for classification. For this problem, values greater than or equal to 0.5 can be considered potable (1), and values less than 0.5 as non-potable (0).

```
train_df["Potability"] =
train_df["Potability"].apply(lambda x: 1 if x >= 0.5
else 0)
```

Step 4: Feature and Target Splitting

Now, we separate the features (`X`) from the target variable (`y`). We will also separate the data into training and validation sets for model evaluation.

```
X = train_df.drop(columns=["Potability", "Index"],
errors="ignore")
y = train_df["Potability"]
```

```
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)
```

Step 5: Model Selection

For this problem, we chose a **Random Forest Classifier**, a powerful ensemble learning method that performs well for both regression and classification tasks. The Random Forest algorithm builds multiple decision trees and merges them to improve predictive accuracy and control overfitting.

```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(n_estimators=100, random_state=42)

model.fit(X_train, y_train)
```

Step 6: Model Evaluation

After training the model, we evaluate it using metrics such as accuracy. Accuracy gives us an idea of how well the model is predicting potable vs. non-potable water samples.

```
from sklearn.metrics import accuracy_score

y_val_pred = model.predict(X_val)

accuracy = accuracy_score(y_val, y_val_pred)

print(f"Validation Accuracy: {accuracy:.4f}")
```

Step 7: Predictions on Test Data

After evaluating the model, we use it to make predictions on the test dataset and generate a submission file. The test data does not contain the target variable (`Potability`), so predictions will be generated and saved in the `submission.csv` file.

```
test_predictions = model.predict(X_test)

submission = pd.DataFrame({

    "Index": test_df["Index"],

    "Potability": test_predictions

})

submission.to_csv("submission.csv", index=False)
```

Key Points to Consider

1. **Data Quality:** The dataset can contain missing or corrupted values. Proper handling of missing data (e.g., filling with the median or mean) is critical for model performance.
2. **Feature Engineering:** Sometimes, features in the dataset are not clean (e.g., text or non-numeric characters). These features must be cleaned and converted into usable numeric data, often using regular expressions or other parsing techniques.
3. **Model Selection:** Random Forest Classifier is a good choice for binary classification tasks due to its robustness to overfitting and ability to handle a mix of categorical and continuous features.
4. **Imbalanced Data:** If the data has an imbalance between potable and non-potable samples, techniques like resampling or using different evaluation metrics (such as ROC AUC or F1-score) might be necessary.
5. **Model Evaluation:** It's essential to evaluate the model using multiple metrics like accuracy, precision, recall, and F1-score to ensure it performs well for both classes (potable and non-potable).

Techniques and Algorithms Used

- **Random Forest Classifier:** An ensemble learning technique that constructs multiple decision trees, each based on a random subset of features, and

combines their predictions. This technique is particularly effective for high-dimensional datasets.

- **Data Preprocessing:**
  - Handling missing values using median imputation.
  - Feature extraction from text-based features (using regular expressions).
- **Train-Test Split:** Dividing the dataset into a training set (used for learning) and a validation set (used for evaluation) ensures that the model is generalizable.

Conclusion

Random Forest, along with proper data preprocessing, is an effective combination for this classification problem. After training and validating the model, predictions are generated on test data and saved for submission.

This process highlights the importance of data cleaning, proper model selection, and evaluation to build robust machine learning models for practical problems like water potability prediction.