xc3g83feg

December 5, 2024

[34]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
import numpy as np
import joblib
```

[16]:
```
# Step 1: Load the updated CSV file with balanced water quality data
file_path = 'balanced_water_quality_with_nanoparticles.csv'  # Replace with the␣
 ↪correct path
data = pd.read_csv(file_path)
```

[17]:
```
# Step 2: Data Overview
print("Data Overview:")
print(data.head())
```

```
Data Overview:
         pH  Lead (mg/L)  Mercury (mg/L)  Bacteria (cfu/mL)  Arsenic (mg/L)  \
0  9.065749     0.004525        0.000441         162.574511        0.010283
1  6.733394     0.001154        0.001659          69.677048        0.001700
2  7.782632     0.003603        0.000154          84.726990        0.003943
3  7.586133     0.002265        0.000172           3.167012        0.001132
4  7.975943     0.013698        0.000356         169.508169        0.002695

   Contaminant Level Nanomaterial Required Water Safe to Drink
0           0.799450  Silver Nanoparticles                  No
1           0.910240  Silver Nanoparticles                  No
2           0.584275        Graphene Oxide                 Yes
3           0.207570        Graphene Oxide                 Yes
4           0.075436  Silver Nanoparticles                  No
```

[18]:
```
# Step 3: Data Preprocessing
# Convert the 'Nanomaterial Required' and 'Water Safe to Drink' columns to␣
 ↪categorical variables
data['Nanomaterial Required'] = data['Nanomaterial Required'].
 ↪astype('category').cat.codes
```

```
data['Water Safe to Drink'] = data['Water Safe to Drink'].map({'Yes': 1, 'No':␣
↪0})

# Define the features (input variables) and the target (output variable)
features = ['pH', 'Lead (mg/L)', 'Mercury (mg/L)', 'Bacteria (cfu/mL)',␣
↪'Arsenic (mg/L)']
target = 'Nanomaterial Required'

X = data[features]
y = data[target]
```

[19]:
```
# Step 4: Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[20]:
```
# Step 5: Train a RandomForest Classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

[20]:
```
RandomForestClassifier(random_state=42)
```

[21]:
```
# Step 6: Make predictions on the test set
y_pred = model.predict(X_test)
```

[30]:
```
# Step 7: Evaluate the model's performance
accuracy = accuracy_score(y_test, y_pred)
print(f"Nanomaterial Prediction Model Accuracy: {accuracy * 100:.2f}%")

# Ensure the predicted and true labels cover all classes
unique_classes_in_test = np.unique(y_test)
unique_classes_in_pred = np.unique(y_pred)

print(f"Unique classes in test data: {unique_classes_in_test}")
print(f"Unique classes in predictions: {unique_classes_in_pred}")

# Classification Report for nanomaterial prediction
# Dynamically adjusting target names based on unique predicted classes
target_names = ['Graphene Oxide', 'Silver Nanoparticles', 'Zinc Oxide␣
↪Nanoparticles']
report = classification_report(y_test, y_pred, labels=unique_classes_in_pred,␣
↪target_names=[target_names[i] for i in unique_classes_in_pred])
print("Nanomaterial Classification Report:\n", report)
```

```
Nanomaterial Prediction Model Accuracy: 100.00%
Unique classes in test data: [0 1]
Unique classes in predictions: [0 1]
Nanomaterial Classification Report:
```

```
                         precision    recall  f1-score   support

       Graphene Oxide       1.00      1.00      1.00       581
Silver Nanoparticles       1.00      1.00      1.00       419

            accuracy                           1.00      1000
           macro avg       1.00      1.00      1.00      1000
        weighted avg       1.00      1.00      1.00      1000
```

[32]:
```
# Step 8: Predict whether the water is safe to drink (binary classification)
# Define a new target for water safety prediction
target_safety = 'Water Safe to Drink'
y_safety = data[target_safety]

# Split the data for water safety prediction
X_train_safety, X_test_safety, y_train_safety, y_test_safety =␣
 ↪train_test_split(X, y_safety, test_size=0.2, random_state=42)

# Train a RandomForest Classifier for water safety prediction
model_safety = RandomForestClassifier(n_estimators=100, random_state=42)
model_safety.fit(X_train_safety, y_train_safety)

# Make predictions for water safety
y_pred_safety = model_safety.predict(X_test_safety)

# Evaluate the water safety model
accuracy_safety = accuracy_score(y_test_safety, y_pred_safety)
print(f"Water Safety Model Accuracy: {accuracy_safety * 100:.2f}%")

# Classification Report for water safety
# Ensure the correct number of target names for the binary classification
report_safety = classification_report(y_test_safety, y_pred_safety,␣
 ↪target_names=['Not Safe', 'Safe'])
print("Water Safety Classification Report:\n", report_safety)
```

```
Water Safety Model Accuracy: 100.00%
Water Safety Classification Report:
               precision    recall  f1-score   support

    Not Safe       1.00      1.00      1.00       442
        Safe       1.00      1.00      1.00       558

    accuracy                           1.00      1000
   macro avg       1.00      1.00      1.00      1000
weighted avg       1.00      1.00      1.00      1000
```

[37]:
```
# Save the trained models as .pkl files
joblib.dump(model, 'model_nanomaterial.pkl')
joblib.dump(model_safety, 'model_safety.pkl')

print("Models have been saved as 'model_nanomaterial.pkl' and 'model_safety.
 ↪pkl'")
```

```
Models have been saved as 'model_nanomaterial.pkl' and 'model_safety.pkl'
```

[ ]:
```
```