Duc Thinh Pham - 104169675

Portfolio – Week 2

Studio 1, 2

Dataset selected in Studio 1: Water Quality in Water Engineering

All the datasets in Studio 1 are from the Engineering field, which is not my major (I'm studying Computer Science, majoring in Data Science). Therefore, I chose a dataset based on my personal interest: the water potability dataset. Specifically, I've always been concerned about the global issue of saltwater intrusion, including its impact in my home country. By working with this dataset, I aim to enhance my skills in Data and ML engineering while also gaining a deeper understanding of the factors that influence water potability.

EDA (Exploratory Data Analysis) Summary

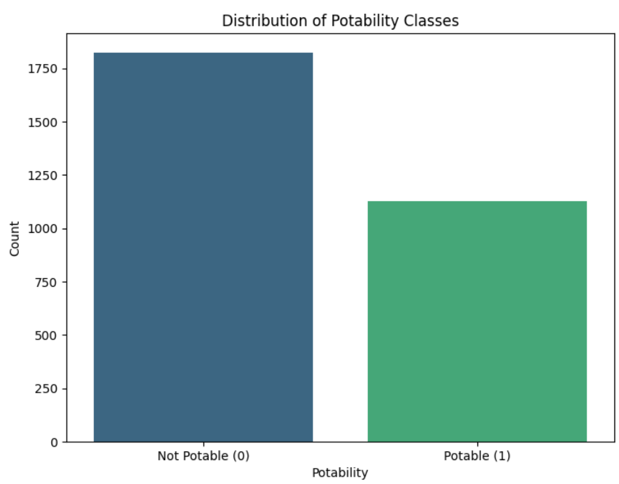
1. Except for the 'Conductivity', 'Sulfate', 'Organic_carbon', and 'Trihalomethanes' features, all other features show very weak relationships with the 'Potability' target variable, making them less significant for correlation-based decision-making.
2. The 'Conductivity' feature has a low positive correlation with the 'Sulfate' feature, suggesting that we can create additional features like (Conductivity + Sulfate) to improve the prediction of water potability.
3. The 'Turbidity' feature shows a low positive correlation with the 'Organic_carbon' feature, indicating the possibility of creating new features like (Turbidity + Organic_carbon) to enhance predictions.
4. The 'Trihalomethanes' feature exhibits a low positive correlation with the 'Chloramines' feature, suggesting that combining these two features (Trihalomethanes + Chloramines) could improve model performance.

Class labelling for target variable / developing ground truth data

For the "Water Portability" dataset, I believe the target variable, 'Portability', doesn't need to be relabeled for class classification, given its binary nature (0 and 1). The binary format already provides clear and distinct classes, which are sufficient for most classification models.

Additionally, I have implemented a bar chart to visualize the distribution of classes for the Potability variable, showing the counts for Non-portable (0) and Portable (1).

```
# Load the dataset again for continuity
file_path = 'water_potability_no_outliers.csv'
water_df = pd.read_csv(file_path)

# Plotting the distribution of classes for Potability
plt.figure(figsize=(8, 6))
sns.countplot(x=water_df['Potability'], palette='viridis')
plt.title('Distribution of Potability Classes')
plt.xlabel('Potability')
plt.ylabel('Count')
plt.xticks(ticks=[0, 1], labels=['Not Potable (0)', 'Potable (1)'])
plt.show()
```



The bar chart clearly shows an imbalance in the distribution of the two potability classes:

1. **Not Potable:** This class has a larger number of samples, indicating that most of the samples are non-potable.
2. **Portable:** This class represents a smaller proportion of the samples, as it has fewer instances, indicating that a minority of the water samples are drinkable.

Feature engineering

1. Convert the target values to the new categorial values (0,1, 2,...)

This conversion is unnecessary because, as mentioned, the existing binary labels—0 for non-potable and 1 for potable—are already appropriate for classification tasks.

2. Normalization

Normalizing input features, except for the Potability feature, helps ensure that all numerical features are on a similar scale, which can improve the performance and convergence of machine learning models. Then, I saved it as "normalised_water_potability.csv" file.

```
from sklearn.preprocessing import MinMaxScaler


normalised_water_df = water_df.copy()


# Initialize MinMaxScaler

scaler = MinMaxScaler()


# List of features to normalize (excluding Potability)

features_to_normalize = normalised_water_df.columns.drop('Potability')


# Normalize the features

normalised_water_df[features_to_normalize] =
scaler.fit_transform(normalised_water_df[features_to_normalize])


# Save the normalized data to a new CSV file

normalised_water_df.to_csv("normalised_water_potability.csv",
index=False)


print("===============================================================
===============================")

print("NORMALIZED DATA")

print(normalised_water_df)
```

And its output:

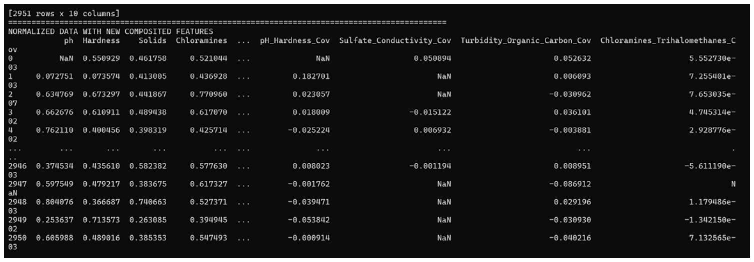
```
  sns.countplot(x=water_df['Potability'], palette='viridis')
==========================================================================================
NORMALIZED DATA
            ph  Hardness    Solids  Chloramines   Sulfate  Conductivity  Organic_carbon  Trihalomethanes  Turbidity  Potability
0          NaN  0.550929  0.461758     0.521044  0.667755      0.804335        0.280742         0.743011   0.258967           0
1     0.072751  0.073574  0.413005     0.436928       NaN      0.867709        0.549332         0.382523   0.624070           0
2     0.634769  0.673297  0.441867     0.770960       NaN      0.481211        0.643817         0.501162   0.281003           0
3     0.662676  0.610911  0.489438     0.617070  0.611859      0.358484        0.731546         0.899974   0.654492           0
4     0.762110  0.400456  0.398319     0.425714  0.387175      0.436423        0.346683         0.096466   0.523011           0
...        ...       ...       ...          ...       ...           ...             ...              ...        ...         ...
2946  0.374534  0.435610  0.582382     0.577630  0.558100      0.475180        0.375183         0.430619   0.426752           1
2947  0.597549  0.479217  0.383675     0.617327       NaN      0.423203        0.813613              NaN   0.219812           1
2948  0.804076  0.366687  0.740663     0.527371       NaN      0.511013        0.317632         0.541433   0.338693           1
2949  0.253637  0.713573  0.263085     0.394945       NaN      0.446342        0.324899         0.631288   0.673463           1
2950  0.605988  0.489016  0.385353     0.547493       NaN      0.279075        0.603068         0.645517   0.103670           1

[2951 rows x 10 columns]
```

3. Composite features using covariance

Based on the insights from the EDA, I have created four new input features by combining four pairs of existing features and then saved as "normalised_water_potability_with_composites.csv" file

```
# DATASET WITH COMPOSITING FEATURES


import numpy as np


normalised_water_df_with_compositing_features = normalised_water_df.copy()


# Create composite features by calculating covariance

normalised_water_df_with_compositing_features['pH_Hardness_Cov'] =
(normalised_water_df_with_compositing_features['ph'] -
normalised_water_df_with_compositing_features['ph'].mean()) *
(normalised_water_df_with_compositing_features['Hardness'] -
normalised_water_df_with_compositing_features['Hardness'].mean())

normalised_water_df_with_compositing_features['Sulfate_Conductivity_Cov'] =
(normalised_water_df_with_compositing_features['Sulfate'] -
normalised_water_df_with_compositing_features['Sulfate'].mean()) *
(normalised_water_df_with_compositing_features['Conductivity'] -
normalised_water_df_with_compositing_features['Conductivity'].mean())

normalised_water_df_with_compositing_features['Turbidity_Organic_Carbon_Cov']
= (normalised_water_df_with_compositing_features['Turbidity'] -
normalised_water_df_with_compositing_features['Turbidity'].mean()) *
(normalised_water_df_with_compositing_features['Organic_carbon'] -
normalised_water_df_with_compositing_features['Organic_carbon'].mean())

normalised_water_df_with_compositing_features['Chloramines_Trihalomethanes_Co
v'] = (normalised_water_df_with_compositing_features['Chloramines'] -
normalised_water_df_with_compositing_features['Chloramines'].mean()) *
(normalised_water_df_with_compositing_features['Trihalomethanes'] -
normalised_water_df_with_compositing_features['Trihalomethanes'].mean())


# Save the dataset with the new composite features

normalised_water_df_with_compositing_features.to_csv("normalised_water_potabi
lity_with_composites.csv", index=False)


print("=========================================================================
=======================")

print("NORMALIZED DATA WITH NEW COMPOSITED FEATURES")

print(normalised_water_df_with_compositing_features)
```

And its output:



```
[2951 rows x 10 columns]
==================================================================================================
NORMALIZED DATA WITH NEW COMPOSITED FEATURES
            ph  Hardness    Solids  Chloramines  ...  pH_Hardness_Cov  Sulfate_Conductivity_Cov  Turbidity_Organic_Carbon_Cov  Chloramines_Trihalomethanes_C
ov
0          NaN  0.550929  0.461758     0.521044  ...              NaN                  0.050894                      0.052632                   5.552730e-
03
1     0.072751  0.073574  0.413005     0.436928  ...         0.182701                       NaN                      0.006093                   7.255401e-
03
2     0.634769  0.673297  0.441867     0.770960  ...         0.023057                       NaN                     -0.030962                   7.653035e-
07
3     0.662676  0.610911  0.489438     0.617070  ...         0.018009                 -0.015122                      0.036101                   4.745314e-
02
4     0.762110  0.400456  0.398319     0.425714  ...        -0.025224                  0.006932                     -0.003881                   2.928776e-
02
...        ...       ...       ...          ...  ...              ...                       ...                           ...                            .
..
2946  0.374534  0.435610  0.582382     0.577630  ...         0.008023                 -0.001194                      0.008951                  -5.611190e-
03
2947  0.597549  0.479217  0.383675     0.617327  ...        -0.001762                       NaN                     -0.086912                            N
aN
2948  0.804076  0.366687  0.740663     0.527371  ...        -0.039471                       NaN                      0.029196                   1.179486e-
03
2949  0.253637  0.713573  0.263085     0.394945  ...        -0.053842                       NaN                     -0.030930                  -1.342150e-
02
2950  0.605988  0.489016  0.385353     0.547493  ...        -0.000914                       NaN                     -0.040216                   7.132565e-
03
```

Feature selection

In the EDA section, I noted that the features with the weakest relationships are ‘ph’, ‘Hardness’, ‘Sulfate’, and ‘Turbidity’. Therefore, I have created datasets with only the selected features, for both normalized and non-normalized versions.

Non-normalized dataset:

```
# DATASET WITH SELECTED NORMALISED FEATURES


# List of features to drop (weakest relationships)

features_to_drop = ['ph', 'Hardness', 'Sulfate', 'Turbidity']


# Create a new dataframe by dropping the selected features

selected_water_df = water_df.drop(columns=features_to_drop)


# Save the new dataframe to a CSV file

selected_water_df.to_csv("selected_features_water_potability.csv",
index=False)


print("=====================================================================
========================")

print("ORIGINAL DATA WITH SELECTED FEATURES")

print(selected_water_df)
```

Normalized dataset:

Duc Thinh Pham - 104169675

```
# DATASET WITH SELECTED NORMALISED FEATURES


# List of features to drop (weakest relationships)

features_to_drop = ['ph', 'Hardness', 'Sulfate', 'Turbidity']


# Create a new dataframe by dropping the selected features

selected_normalised_water_df =
normalised_water_df.drop(columns=features_to_drop)


# Save the new dataframe to a CSV file

selected_normalised_water_df.to_csv("selected_normalised_features_water_potab
ility.csv", index=False)


print("=============================================================================
========================")

print("NORMALIZED DATA WITH SELECTED FEATURES")

print(selected_normalised_water_df)
```

And its output:

```
[2951 rows x 14 columns]
===============================================================================================
ORIGINAL DATA WITH SELECTED FEATURES
            Solids  Chloramines  Conductivity  Organic_carbon  Trihalomethanes  Potability
0     20791.318981     7.300212    564.308654       10.379783        86.990970           0
1     18630.057858     6.635246    592.885359       15.180013        56.329076           0
2     19909.541732     9.275884    418.606213       16.868637        66.420093           0
3     22018.417441     8.059332    363.266516       18.436524       100.341674           0
4     17978.986339     6.546600    398.410813       11.558279        31.997993           0
...            ...          ...           ...             ...              ...         ...
2946  26138.780191     7.747547    415.886955       12.067620        60.419921           1
2947  17329.802160     8.061362    392.449580       19.903225              NaN           1
2948  33155.578218     7.350233    432.044783       11.039070        69.845400           1
2949  11983.869376     6.303357    402.883113       11.168946        77.488213           1
2950  17404.177061     7.509306    327.459760       16.140368        78.698446           1

[2951 rows x 6 columns]
===============================================================================================
NORMALIZED DATA WITH SELECTED FEATURES
        Solids  Chloramines  Conductivity  Organic_carbon  Trihalomethanes  Potability
0     0.461758     0.521044      0.804335        0.280742         0.743011           0
1     0.413005     0.436928      0.867709        0.549332         0.382523           0
2     0.441867     0.770960      0.481211        0.643817         0.501162           0
3     0.489438     0.617070      0.358484        0.731546         0.899974           0
4     0.398319     0.425714      0.436423        0.346683         0.096466           0
...        ...          ...           ...             ...              ...         ...
2946  0.582382     0.577630      0.475180        0.375183         0.430619           1
2947  0.383675     0.617327      0.423203        0.813613              NaN           1
2948  0.740663     0.527371      0.511013        0.317632         0.541433           1
2949  0.263085     0.394945      0.446342        0.324899         0.631288           1
2950  0.385353     0.547493      0.279075        0.603068         0.645517           1

[2951 rows x 6 columns]
```

Duc Thinh Pham - 104169675

Model development

So far, I have created four new datasets based on the original dataset (which includes all features without normalization or composite features – "water_potability_no_outliers.csv"), as follows:

- All features with normalization and without composite features: "normalised_water_potability.csv"
- All features with normalization and including composite features: "normalised_water_potability_with_composites.csv"
- Selected features with normalization: "selected_normalised_features_water_potability.csv"
- Selected features without normalization: "selected_features_water_potability.csv"

Then, I will develop a decision tree classifier using these five datasets:

```
# MODEL DEVELOPMENT - DECISION TREE CLASSIFIER


from sklearn.tree import DecisionTreeClassifier

from sklearn.model_selection import train_test_split

from sklearn import metrics


# Define a function to train and evaluate a Decision Tree classifier

def train_and_evaluate_decision_tree(dataset_path, feature_cols, target_col='Potability'):

    # Load dataset

    data = pd.read_csv(dataset_path)


    # Define features (X) and target (y)

    X = data[feature_cols]

    y = data[target_col]


    # Split dataset into training set and test set (70% training, 30% test)

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
```

```
# Create Decision Tree classifier object
    clf = DecisionTreeClassifier(random_state=1)

    # Train Decision Tree classifier
    clf = clf.fit(X_train, y_train)

    # Predict the response for the test dataset
    y_pred = clf.predict(X_test)

    # Calculate and return the accuracy
    accuracy = metrics.accuracy_score(y_test, y_pred)
    return accuracy

# Paths to your datasets
datasets = [
    "water_potability_no_outliers.csv",
    "normalised_water_potability.csv",
    "normalised_water_potability_with_composites.csv",
    "selected_normalised_features_water_potability.csv",
    "selected_features_water_potability.csv"
]

# Define the feature columns for each dataset
# Since 'Potability' is the target, it should be excluded from feature
columns
all_features = ['ph', 'Hardness', 'Solids', 'Chloramines', 'Sulfate',
'Conductivity',
                'Organic_carbon', 'Trihalomethanes', 'Turbidity']
```

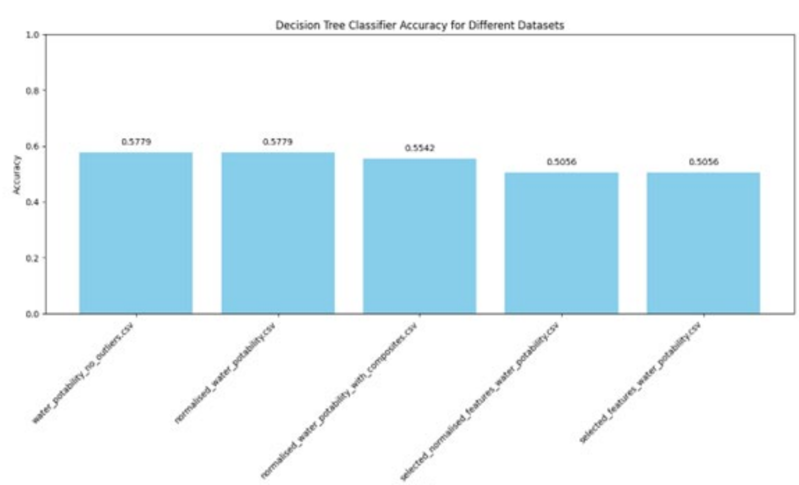
```
composite_features = ['ph', 'Hardness', 'Solids', 'Chloramines', 'Sulfate', 
'Conductivity',
                      'Organic_carbon', 'Trihalomethanes', 'Turbidity',
                      'pH_Hardness_Cov', 'Sulfate_Conductivity_Cov',
                      'Turbidity_Organic_Carbon_Cov', 
'Chloramines_Trihalomethanes_Cov']


selected_features = ['Solids', 'Chloramines', 'Conductivity', 
'Organic_carbon', 'Trihalomethanes']


# Feature sets for each dataset
feature_sets = {
    "water_potability_no_outliers.csv": all_features,
    "normalised_water_potability.csv": all_features,
    "normalised_water_potability_with_composites.csv": composite_features,
    "selected_normalised_features_water_potability.csv": selected_features,
    "selected_features_water_potability.csv": selected_features
}


# Store accuracies for plotting
accuracies = []


# Train and evaluate the Decision Tree on each dataset, and store the 
accuracy
for dataset in datasets:
    accuracy = train_and_evaluate_decision_tree(dataset, 
feature_sets[dataset])
    accuracies.append(accuracy)
    print(f"Accuracy for {dataset}: {accuracy:.4f}")
```

Duc Thinh Pham - 104169675

```
# Plot the accuracies
plt.figure(figsize=(10, 6))
bars = plt.bar(datasets, accuracies, color='skyblue')
plt.xlabel('Dataset')
plt.ylabel('Accuracy')
plt.title('Decision Tree Classifier Accuracy for Different Datasets')
plt.xticks(rotation=45, ha='right')
plt.ylim(0, 1)  # Set the y-axis limits between 0 and 1

# Annotate bars with accuracy values
for bar, accuracy in zip(bars, accuracies):
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval + 0.02, f'{accuracy:.4f}', ha='center', va='bottom')

plt.show()
```

And its output to compare 5 different accuracy value:

```
Accuracy for water_potability_no_outliers.csv: 0.5779
Accuracy for normalised_water_potability.csv: 0.5779
Accuracy for normalised_water_potability_with_composites.csv: 0.5542
Accuracy for selected_normalised_features_water_potability.csv: 0.5056
Accuracy for selected_features_water_potability.csv: 0.5056
```



Duc Thinh Pham - 104169675

Summarisation

I have made a table to compare 5 different models:

Model 1: **WATER_POTABILITY_NO_OUTLIERS.CSV**	**Model 2:** **NORMALISED_WATER_POTABILITY.CSV**	**model 3:** **NORMALISED_WATER_POTABILITY_WITH_COMPOSITES.CSV**	**model 4:** **SELECTED_NORMALISED_FEATURES_WATER_POTABILITY.CSV**	**Model 5:** **SELECTED_FEATURES_WATER_POTABILITY.CSV**
57.79%	57.79%	55.42%	50.56%	50.56%

Observation:

1. **Best Accuracy**: The datasets "normalised_water_potability.csv" and "water_potability_no_outliers.csv" achieved the highest accuracy, both around 0.5779.
2. **Impact of Composite Features**: The dataset "normalised_water_potability_with_composites.csv," which includes composite features, yielded slightly lower accuracy at approximately 0.5542, suggesting that the addition of composite features did not significantly improve model performance.
3. **Feature Selection**: The feature selection process, which involved removing weakly correlated features, resulted in lower accuracy for both normalized and non-normalized selected feature datasets, with an accuracy of about 0.5056.

Appendix

Link to the source code Portfolio – Week 2:

1. Studio1
   https://github.com/thinhpham1807/COS40007_Artificial_Intelligence_for_Engineering/tree/main/Studios/Studio01
2. Studio 2
   https://github.com/thinhpham1807/COS40007_Artificial_Intelligence_for_Engineering/tree/main/Studios/Studio02