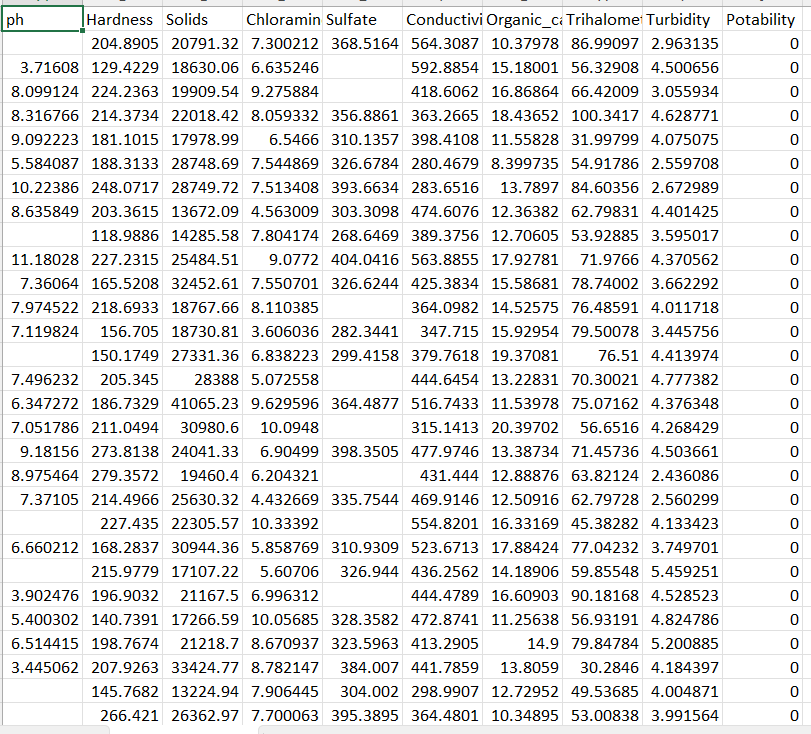
Name: Trung Kien Nguyen

Student ID: 104053642

Studio: 1 - 3

PORTFOLIO REPORT – WEEK 2

# **Chosen Dataset**: Water quality



My major is AI under the Bachelor of Computer Science, so there aren't any datasets available that are really related to it. Therefore, I decided to choose a dataset based on my preference, which is water\_potability. More specifically, I have always paid considerable attention to the problem of saltwater intrusion occurring around the world, including my home country. Familiarizing and accessing this Dataset, in addition to helping me improve my skills in Data and ML engineering, can also help me better understand what properties affect the potability of water.

# EDA:

1. Variable Identification:
   1. Target variable: Potability, which indicates
   2. Predictors (Input variables): ph, Hardness, Solids, Chloramines, Sulfate, Conductivity, Organic\_carbon, Trihalomethanes, Turbidity
2. Univariate analysis

I have used the following code to visualize the distribution of each feature (other than the target – Potability) of the dataset:

# Define a function to plot the distribution with mean line

def plot\_distribution(data, column\_name):

    plt.figure(figsize=(13, 6))

    sns.histplot(data[column\_name], color="b", kde=True)

    plt.axvline(data[column\_name].mean(), linestyle="dashed", color="k", label="mean", linewidth=2)

    plt.legend(loc="best", prop={"size": 14})

    plt.title(f"{column\_name} Distribution")

    plt.show()

# List of columns to plot (excluding 'Potability')

columns = [col for col in data\_no\_outliers.columns if col != 'Potability']

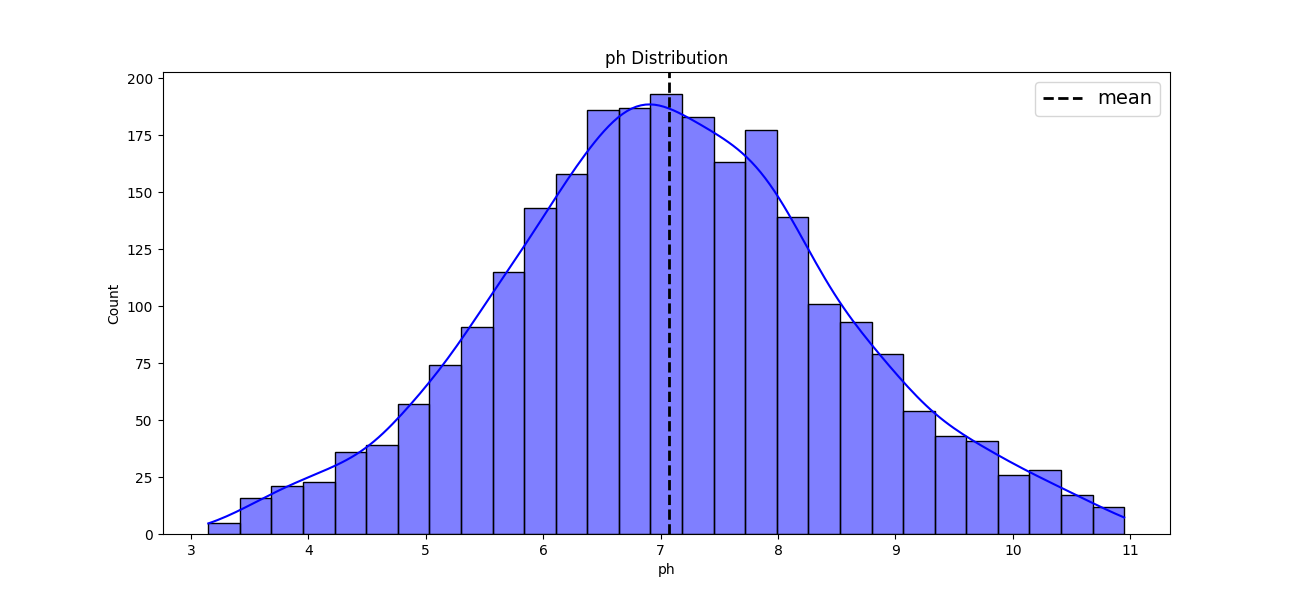
# Plot each column

for col in columns:

    plot\_distribution(data\_no\_outliers, col)

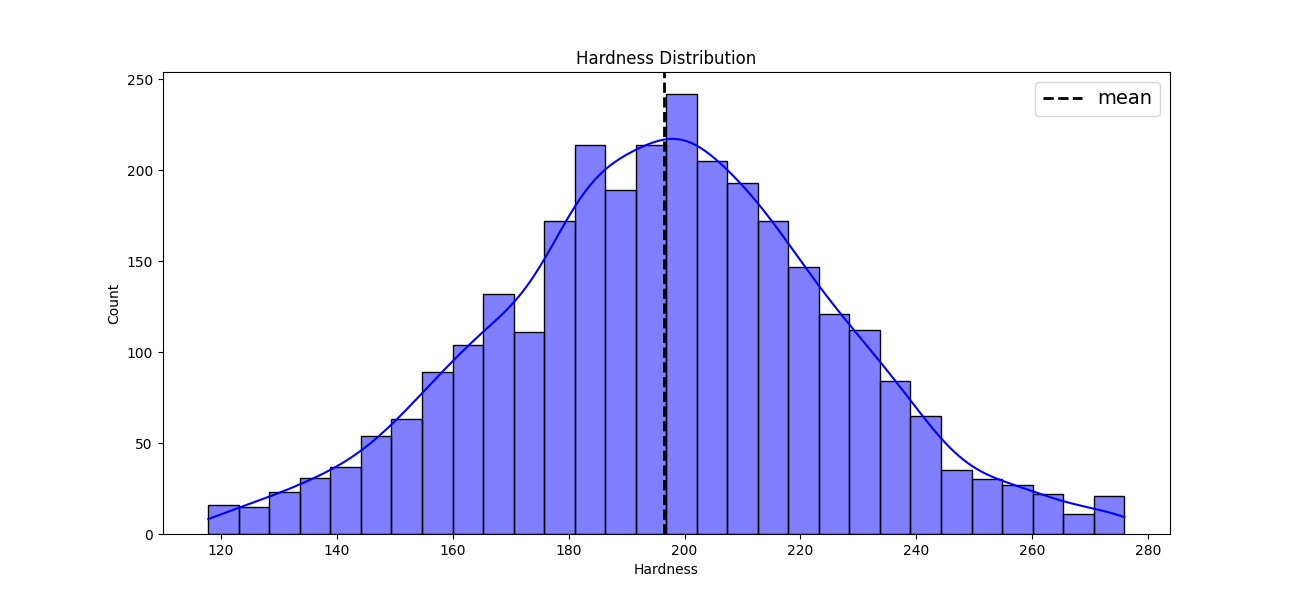
Here is the visualization of distribution and its analysis:

* 1. ph



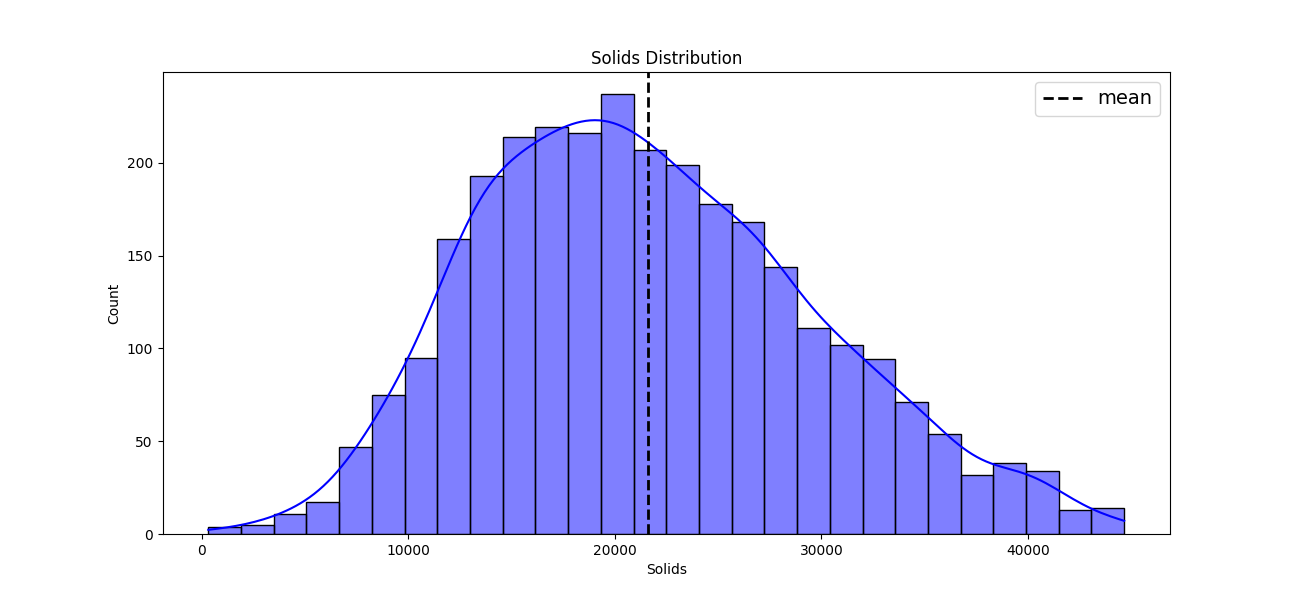
It is slightly right-skewed. Most water samples have pH value near neutral (pH = 7)

* 1. Hardness



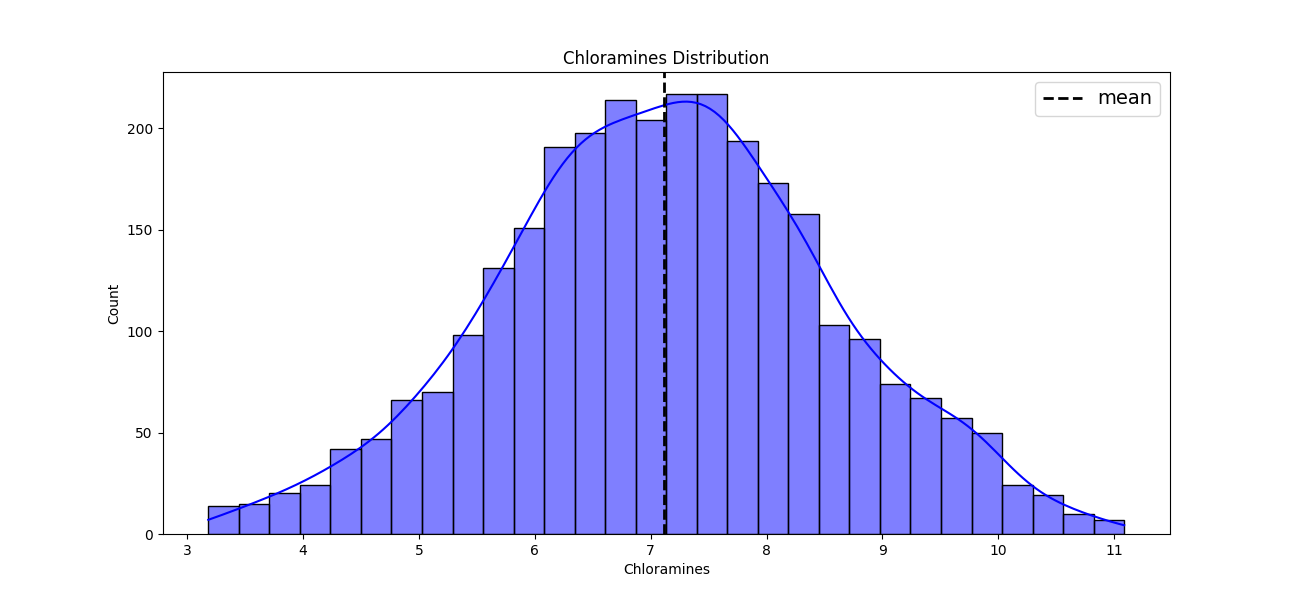
There is no significant skewness (Symmetrical spread of values)

* 1. Solids



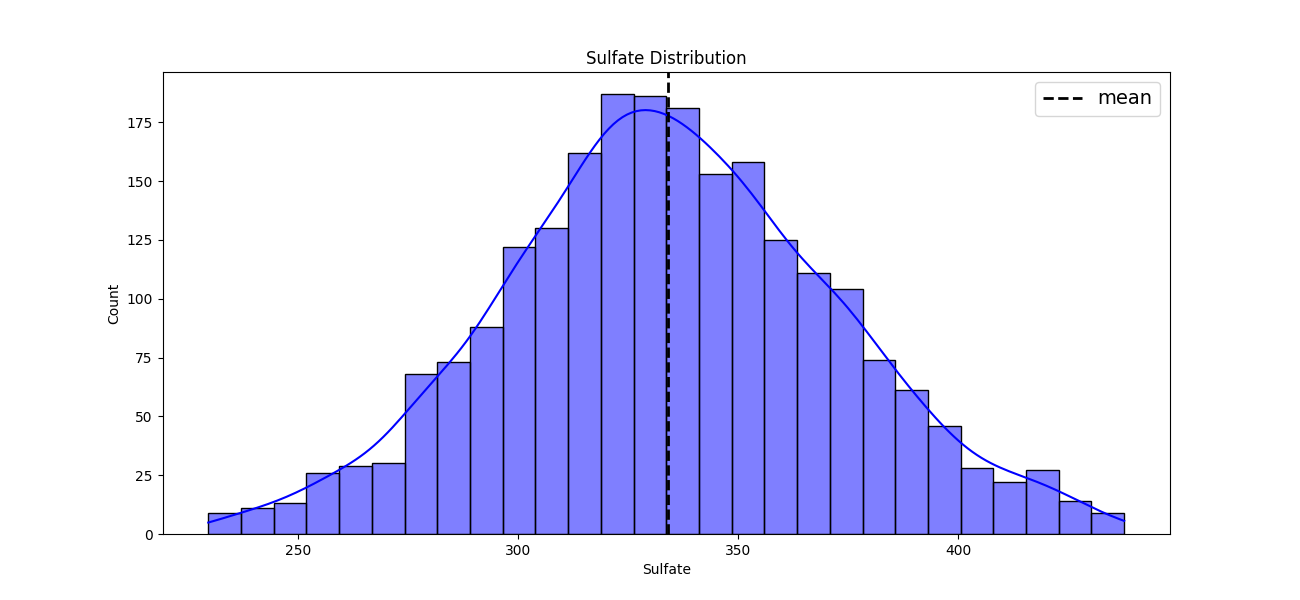
It is moderately right-skewed, showing that the majority of the samples had **solids** features values that are below the mean, with the frequency gradually decreasing as the **solids** rises.

* 1. Chloramines



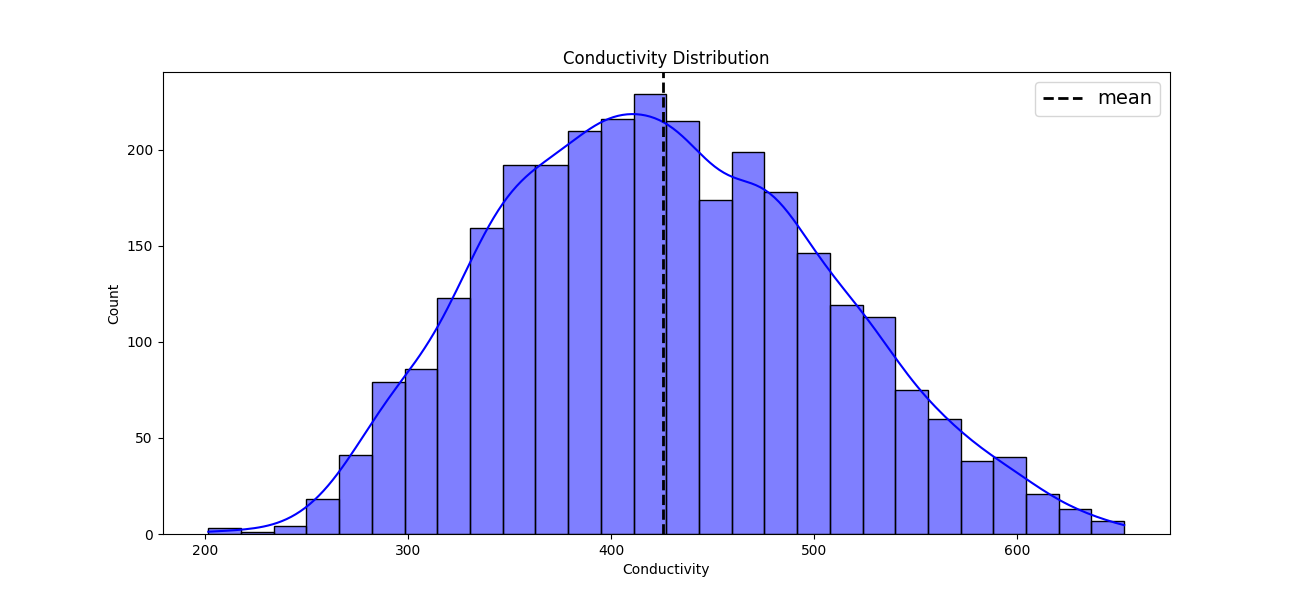
It is overally symmetric, centered about the mean value.

* 1. Sulfate



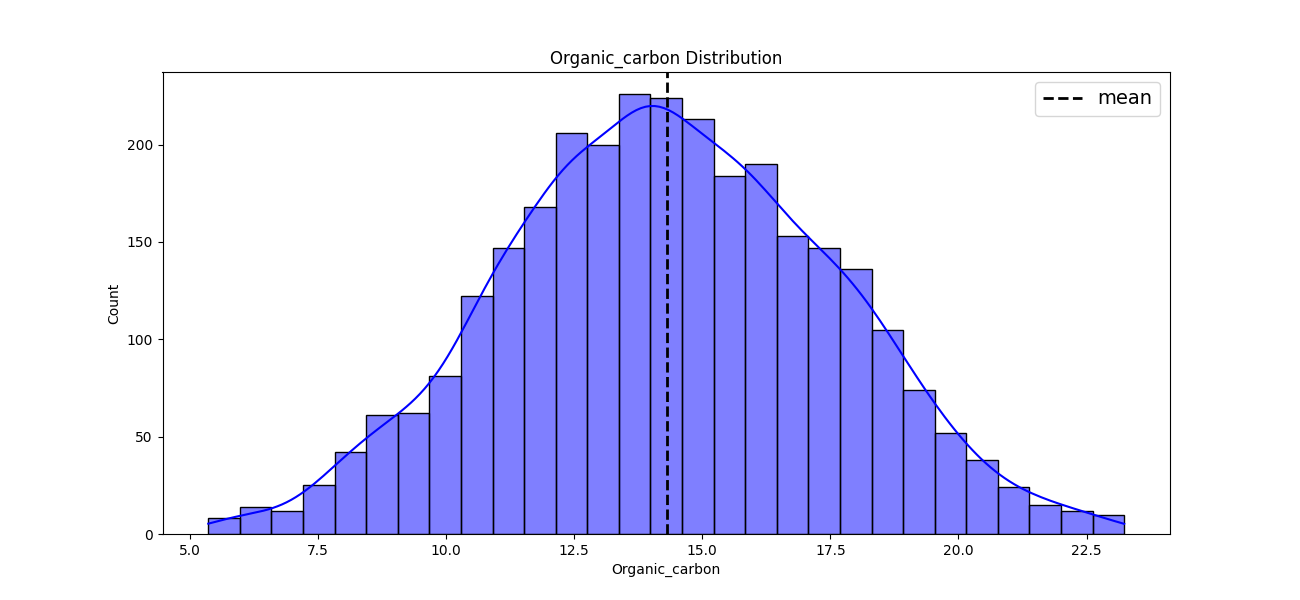
It is slightly right skewd, but overally, it is still symmetric

* 1. Conductivity



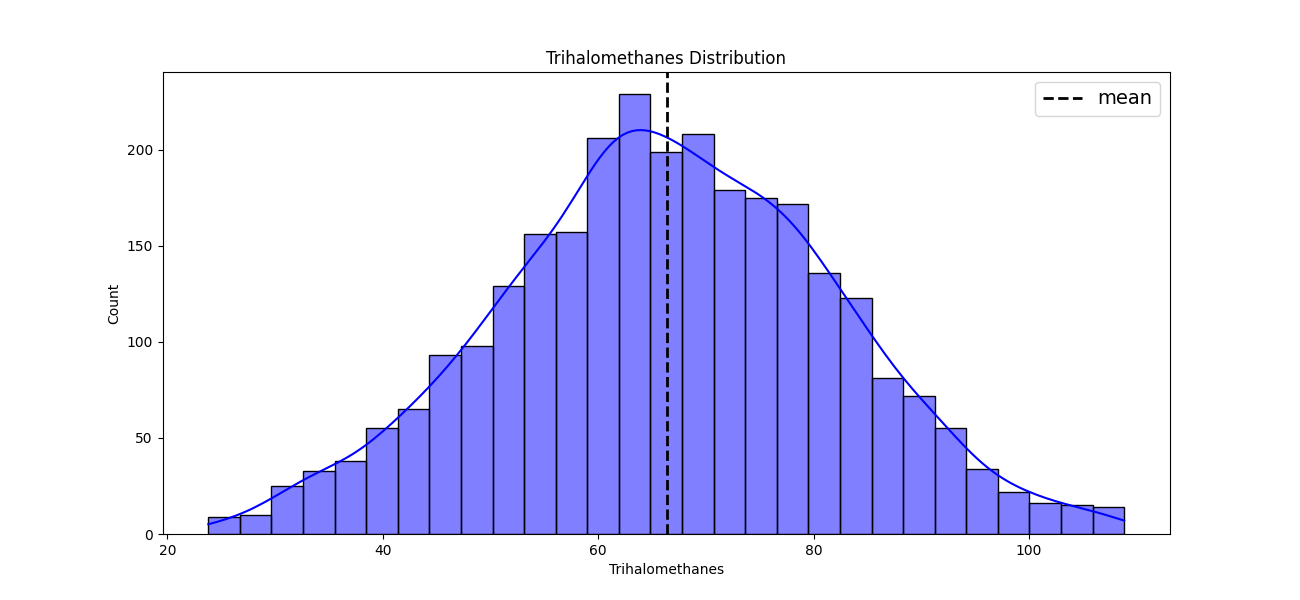
It is symmetric, the values are centered around the mean one.

* 1. Organic\_Carbon



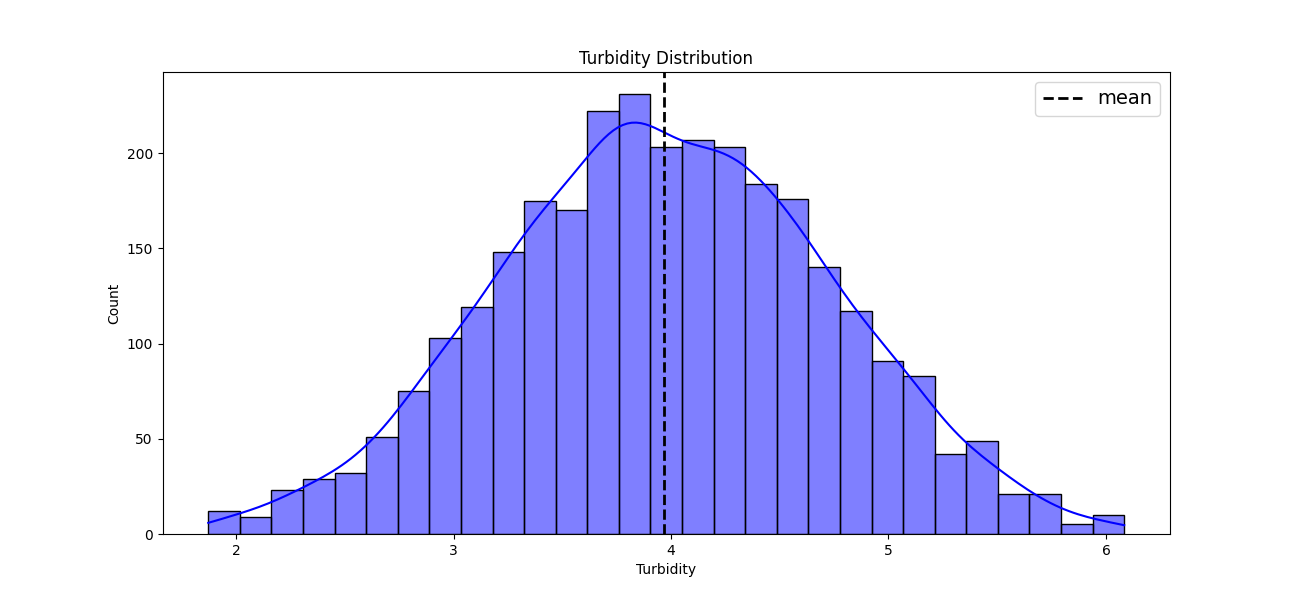
This is also a symmetric distribution

* 1. Trihalomethanes



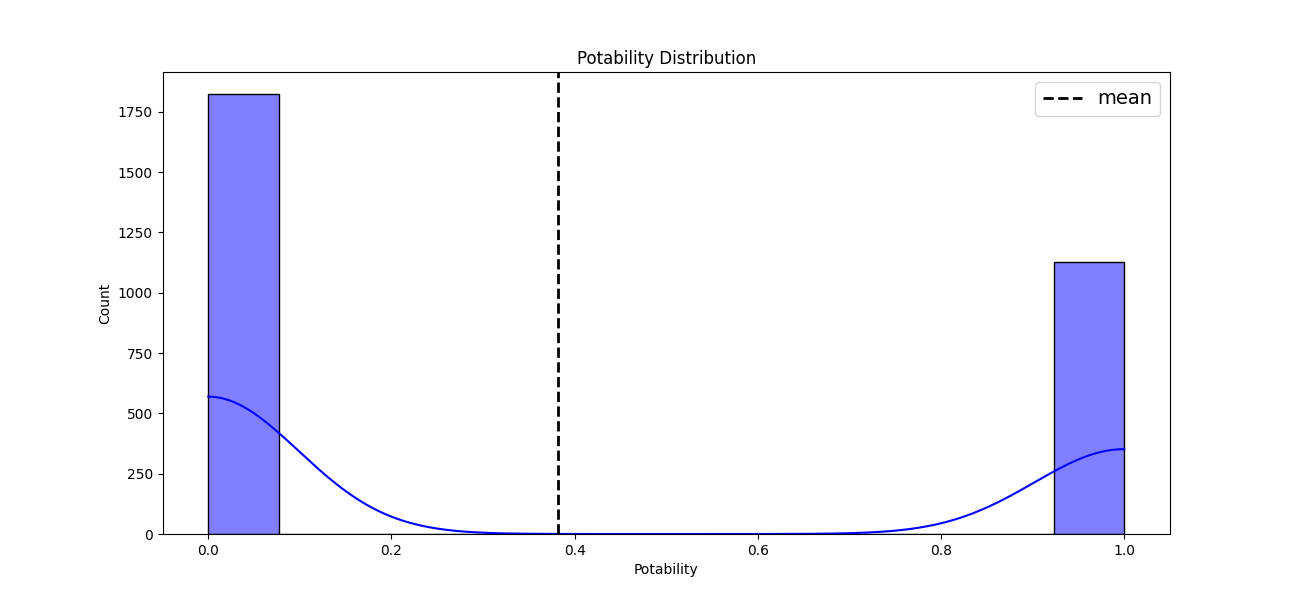
It is just moderately right-sked. Trihalomethanes values of most samples lower than the mean

* 1. Turbidity



This is also symmetric and center around the mean.

* 1. Portability



There is a noticeable class disparity in the Potability feature distribution:

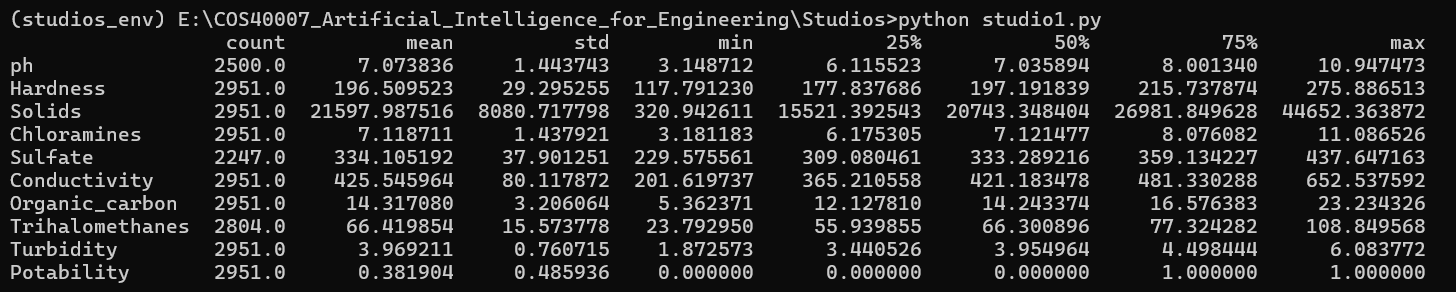
* The majority of samples (about 1750+) fall into category 0 (non-potable)
* Some of them (about 1200) belong to category 1 (potable).

1. Summary statictics

To view the summary of the statistics (No-outliers data) for all features, I have used the following code:

print(data.describe().T)

And here is the output:



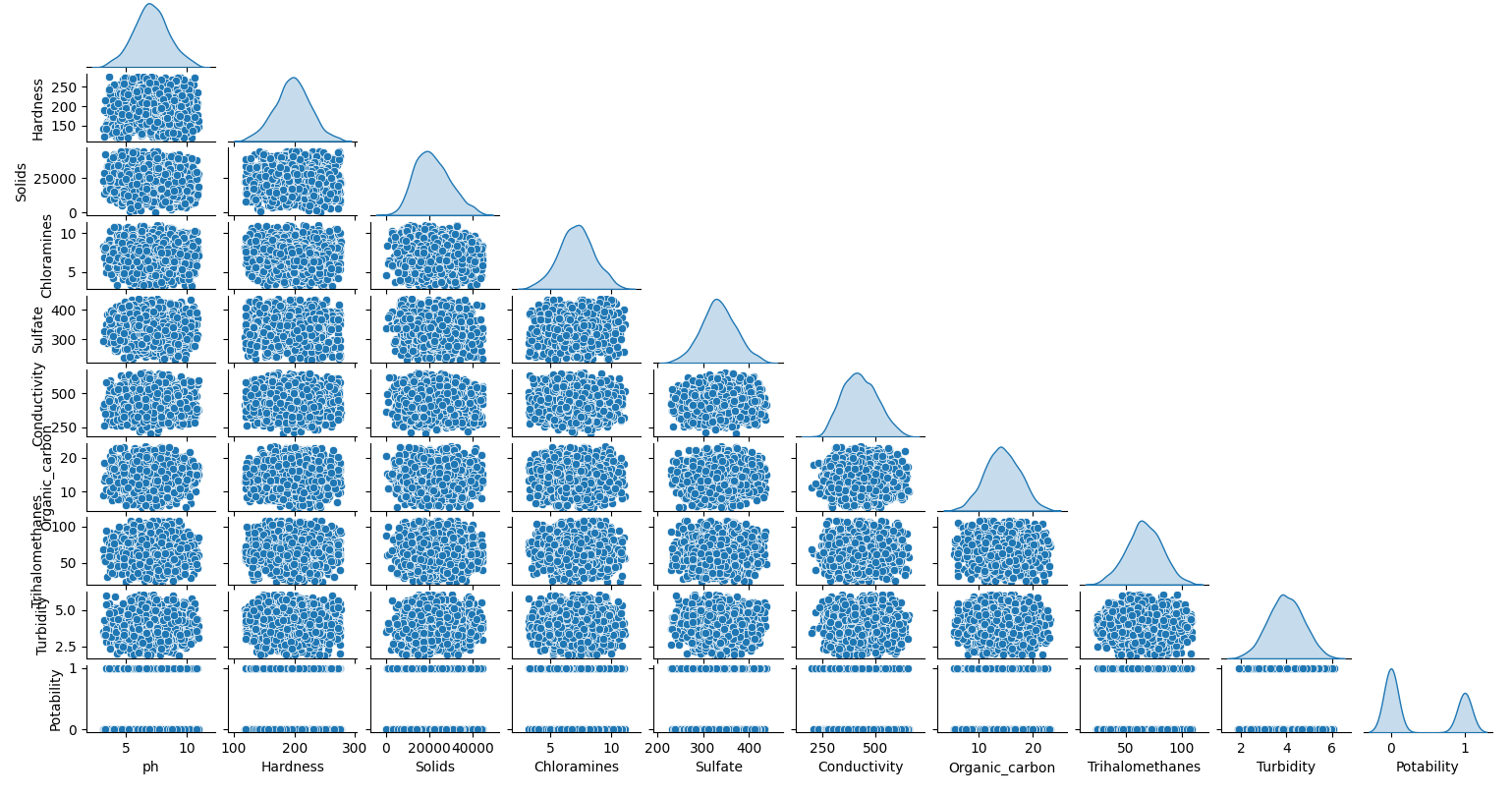
1. Multivariate analasys

To visualize the relationships between variables of the dataset, I have visualized a grid of pairwise plots, where all of the dataset's variables are shown against one another. Here is the code I have implemented:

sns.pairplot(data\_no\_outliers, diag\_kind='kde', corner=True)

plt.show()

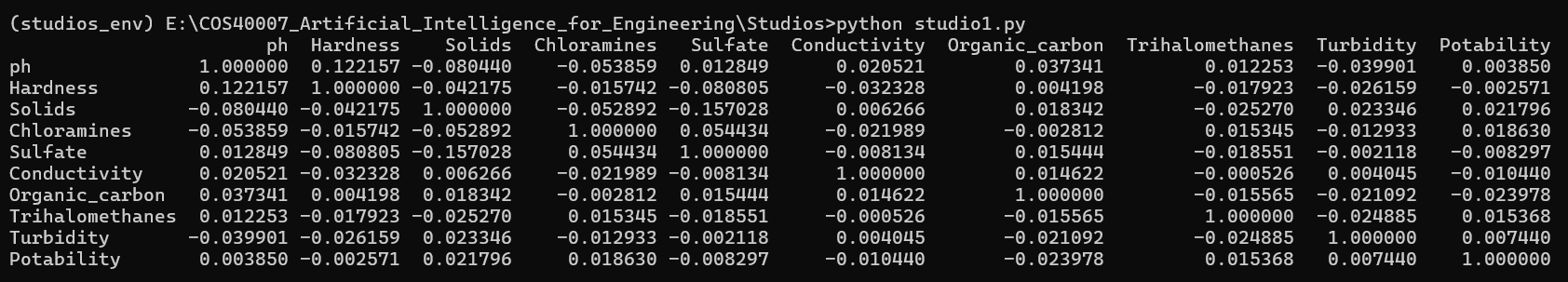
And the output of plots:



* Diagonal Analysis:
  + Gaussian Analysis
    - The Kernel Density Estimation (KDE) plots on the diagonal provides an estimate of the distribution for each input variable
    - The Gaussian distribution of the majority of input variables seems to be at least one, suggesting that they are probably unimodal. Among those, s ulphate and trihalomethane’s distributiosn point to at least two Gaussians, suggesting that the data may be organised into subgroups or clusters.
  + Range of Clusters: seem to be 2 to 4 for the majority of the input variables. On the other hannd, due to its binary nature, the potability variable naturally clusters into two different groups at 0 and 1.
* Off-diagonals anaalysis:
  + Scatter Plot Relationships:
    - ph vs others: There is little (or perhaps no) linear association, as evidenced by the scattered pattern pH displays with other variables
    - Hardness vs other: Hardness and other variables do not show strong linear connections, like pH. The wide dispersion of the scatter points to weak correlations.
    - Solids vs others: Solids still have a generally poor relationship with other factors, although their distribution is slightly better (some slight clustering is visible.
    - Chloramines vs others: Some weak positive correlation: a little cluster of features such as Organic Carbon and Conductivity. The connection to others remain undramatical
    - Sulfate vs others: It also has some clustering, but again the relationship are not strong
    - Conductivity vs other: With solids and sulphate, conductivity exhibits somewhat more clear inteactions, indicating weakly positive correlations.
    - Trihalomethanes vs Others: Although there is clustering, the scatter plots do not demonstrate strong linear correlations
    - Turbidiy vs other: There is weak connection to others indicated by the scatter's wide dispersion.
  + **Potability** vs other input features: The target contains binary values (0/1). The separation between those values may suggest that a mix of characteristics will possibly be helpful in forecasting **potability**.

1. Pariwise correlations among the variables

I have implemented the code for this step. First is the display of correlation matrix: print(data.corr())



Next, I have implemented its visualization of heatmap for better correlation demo:

# Calculate the absolute correlation matrix

corr = abs(data.corr())

# Mask the upper triangle of the heatmap

mask = np.triu(np.ones\_like(corr, dtype=bool))

# Plotting the heatmap

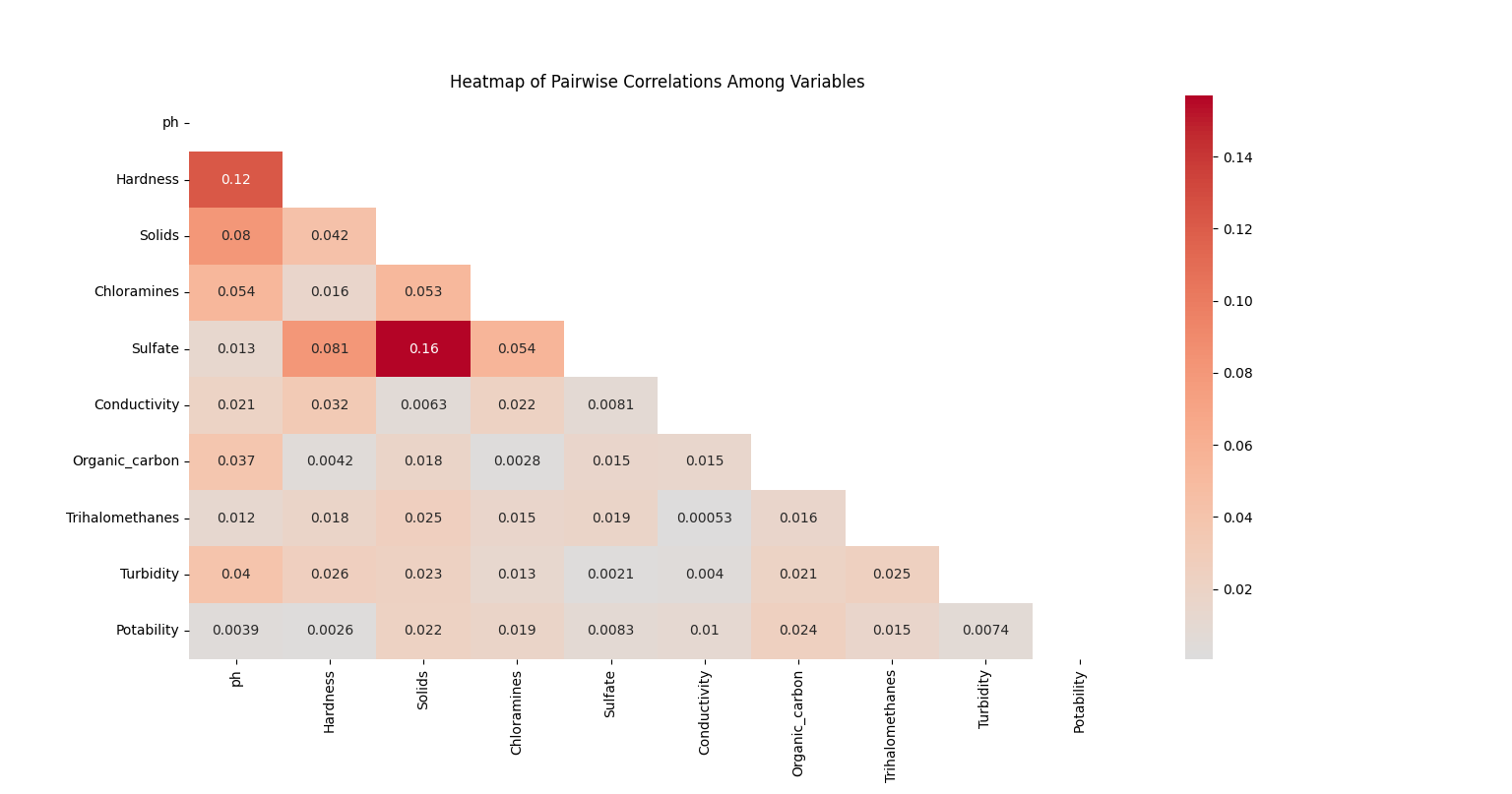
plt.figure(figsize=(12, 10))

sns.heatmap(corr, annot=True, mask=mask, cmap='coolwarm', center=0)

plt.title('Heatmap of Pairwise Correlations Among Variables')

plt.show()

And, here is its output:



Some analysis on the above heatmap is:

* Low correlations in general:
* Potability: Has overally weak connection to all other features. Additionally, the one that has the weakest relationship include **ph**, **hardness**, **Sulfate**, and **turbidity**
* Therefore, it is recommened to create new input features by combining the existing features:
  + Hardness and pH: Despite their poor correlations with potability and each other, these characteristics are essential markers of water quality. Their covariance may show a link that separate analysis does not show.
  + Conductivity and Sulphate: The heatmap displayed a moderate association between these features. Sulphates have an impact on conductivity because they add to the ionic content of water. Their combined fluctuation may be substantila
  + Turbidity and Organic cacbon Species: The physical and chemical properties of water are correlated with both Turbidity and Organic\_carbon. Their interactions could reveal unnoticed trends about the organic content and water purity
  + Trihalomethanes and Chloramines: Trihalomethanes are byproducts of disinfection, and chloramines are used to treat water. Their covariance may capture cooperative effects associated with processes of disinfection.

# Class labelling for target variable / developing ground truth data

For the dataset of “Water Portability”, the target variable – Portability, is not necessary, in my opinion, as it is binary (values of 0 and 1). In particular, the binary nature already offers distinct and well-defined classes, which are adequate for the majority of classification models.

On top of that, I have implemented the code to visualize the bar chart that shows the distribution of classes for the Potability variable: 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')

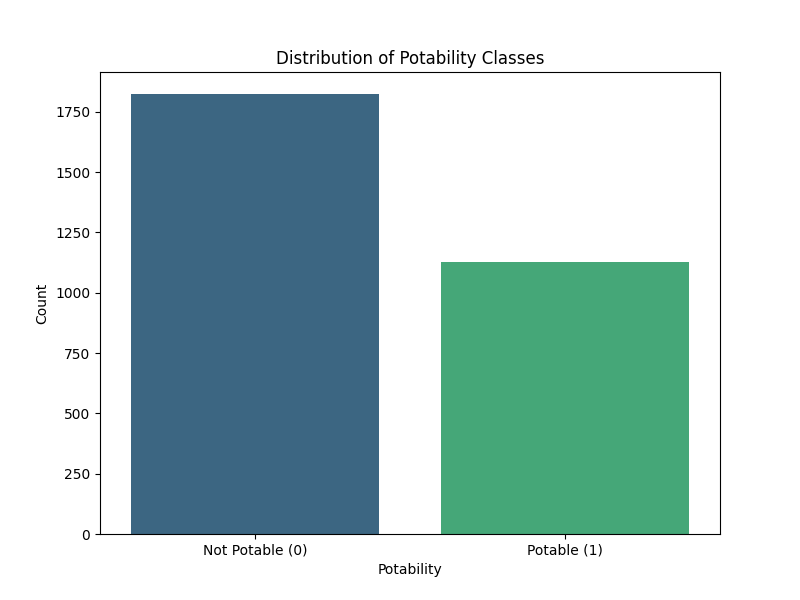
plt.title('Distribution of Potability Classes')

plt.xlabel('Potability')

plt.ylabel('Count')

plt.xticks(ticks=[0, 1], labels=['Not Potable (0)', 'Potable (1)'])

plt.show()



It is clear from the bar chart that the distribution of the two classes of potability is imbalanced:

* Not Potable: The class in which most of the samples are found to be non-potable, as indicated by the greater number of samples
* Portable : A smaller percentage of the water samples are drinkable, as indicated by the minority class, whih has fewer samples

# Feature Engineering

1. Convert Portability values to categorical values (1, 2, 3)

This is not necessary, as like I have mentioned, for classification tasks, the current binary labels—0 for non-potable and 1 for potable—are already suitable.

1. Normalisation:

Here's my code to normalize all input features except for the Potability feature:

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

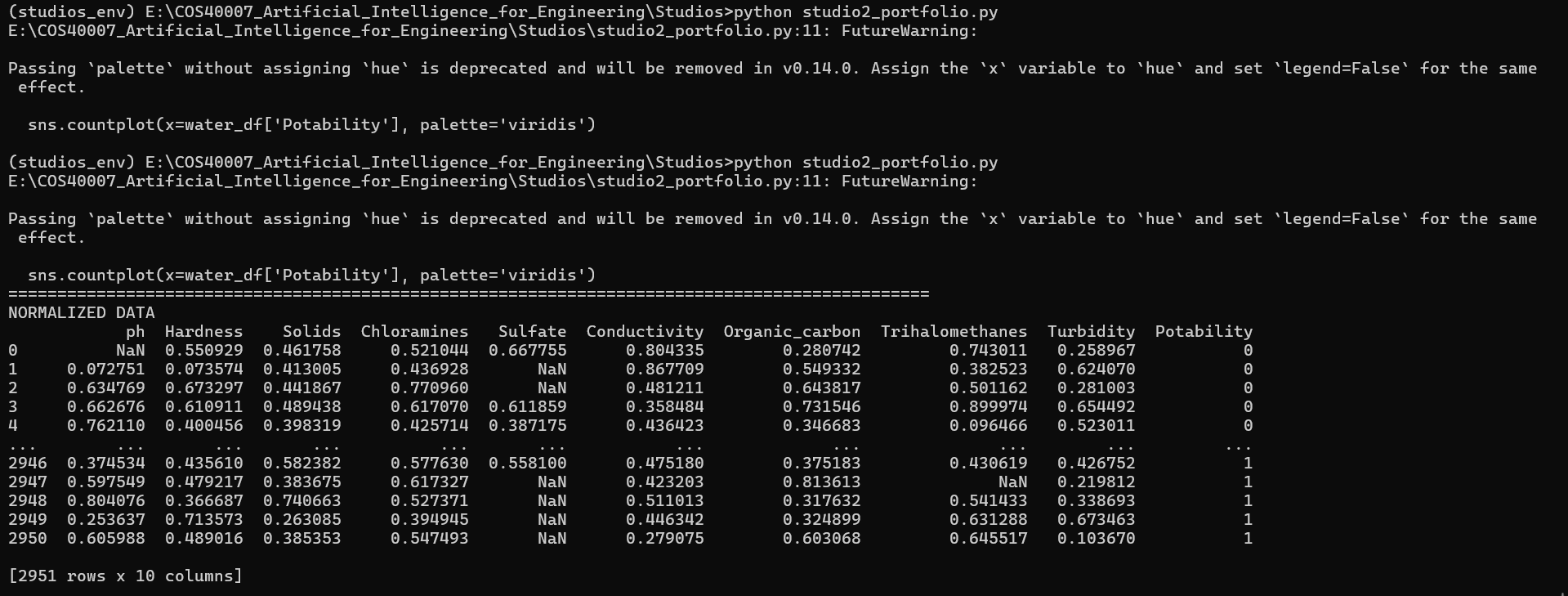
normalised\_water\_df.to\_csv("normalised\_water\_potability.csv", index=False)

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

print("NORMALIZED DATA")

print(normalised\_water\_df)

And its output:



1. Composite features using covariance

Based on the suggestion of the EDA, I have created 4 new input features by compositing 4 pairs of exisitng features. Here is the code and its output:

# 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\_Cov'] = (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())

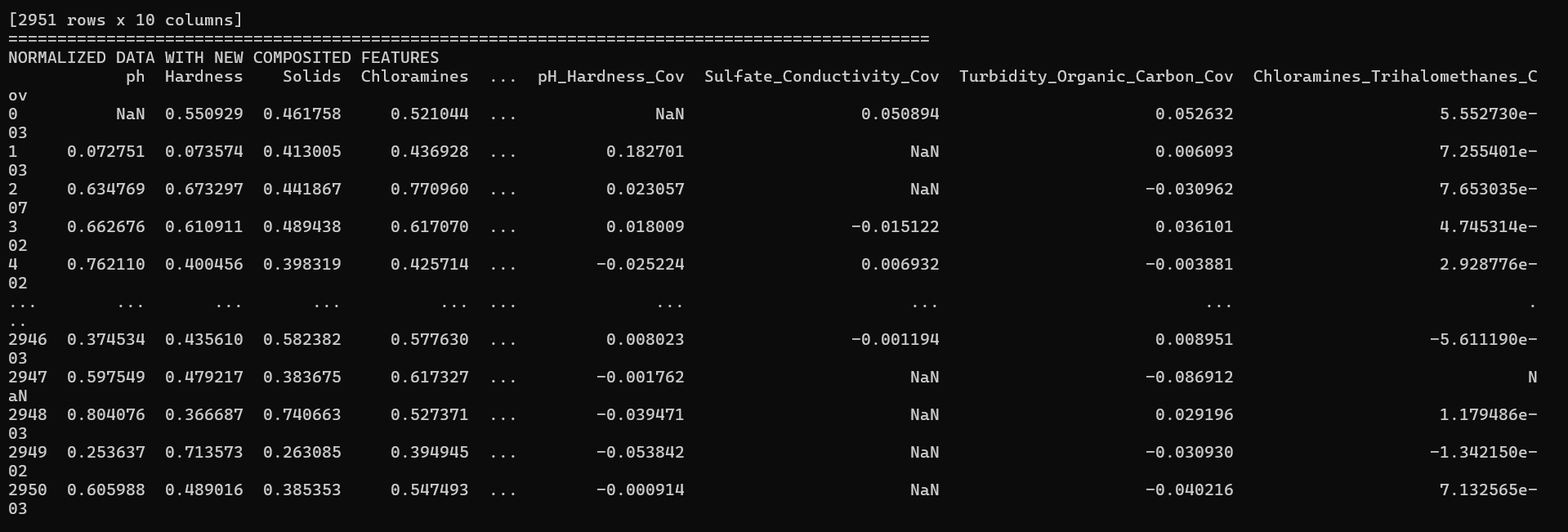
# Save the dataset with the new composite features

normalised\_water\_df\_with\_compositing\_features.to\_csv("normalised\_water\_potability\_with\_composites.csv", index=False)

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

print("NORMALIZED DATA WITH NEW COMPOSITED FEATURES")

print(normalised\_water\_df\_with\_compositing\_features)



# Feature selection

In the EDA section, I have stated that the features that has the weakest relationship include ph, hardness, Sulfate, and turbidity. So, I have created a dataset with selected features only (for both normalised and non-normalised dataset).

* For non-normalised 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)

* For normalised 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\_normalised\_water\_df = normalised\_water\_df.drop(columns=features\_to\_drop)

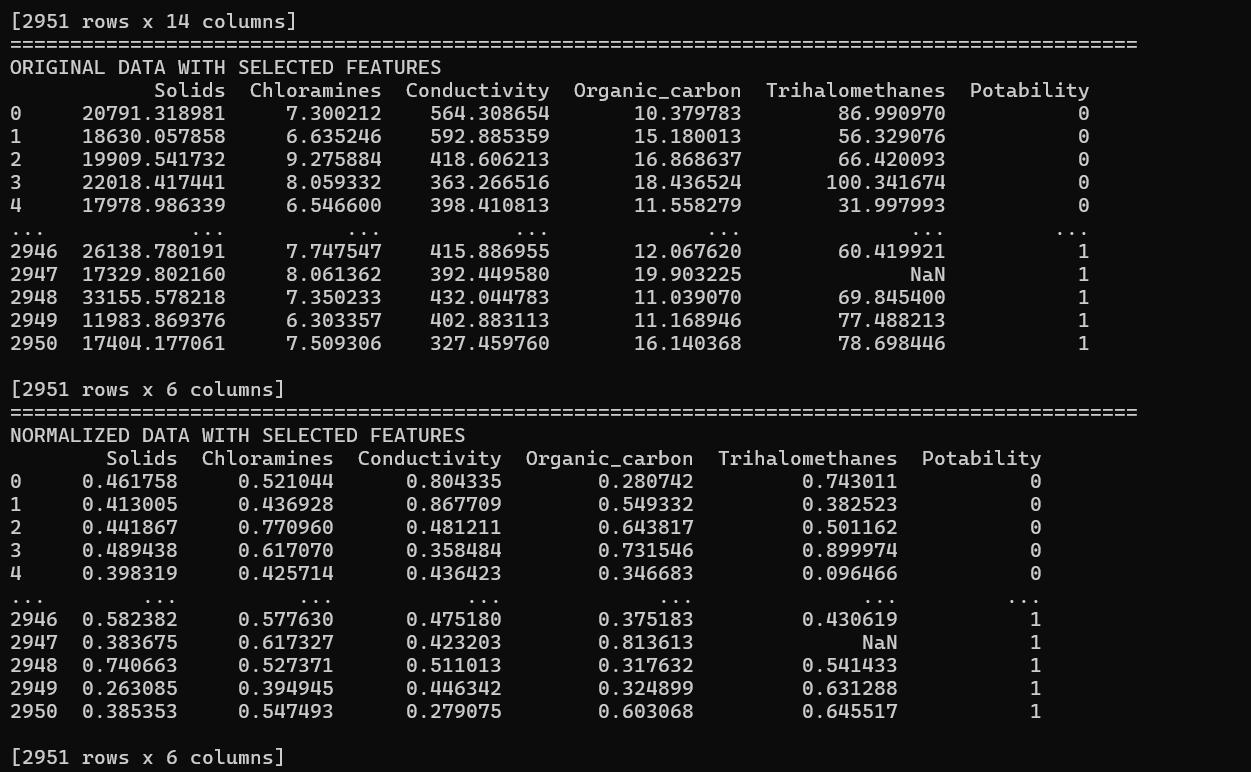
# Save the new dataframe to a CSV file

selected\_normalised\_water\_df.to\_csv("selected\_normalised\_features\_water\_potability.csv", index=False)

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

print("NORMALIZED DATA WITH SELECTED FEATURES")

print(selected\_normalised\_water\_df)

And here is the output: 

# Model development

So far, I have made 4 news dataset based on the original dataset (all features without normalisation and without composite features – “water\_potability\_no\_outliers.csv”), including:

* all features with normalisation and without composite features: “normalised\_water\_potability.csv”
* all features with normalisation and containing composite features: “normalised\_water\_potability\_with\_composites.csv”
* selected features with normalisation: “selected\_normalised\_features\_water\_potability.csv”
* selected feature without normalisation: “selected\_features\_water\_potability.csv”

Now, I will develop a decisionTree classifier with the 5 above dataset. This is my code:

# 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}")

# 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

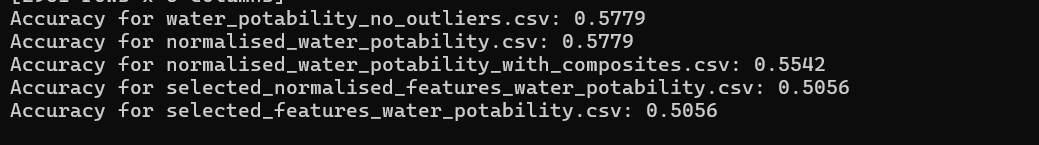
for bar, accuracy in zip(bars, accuracies):

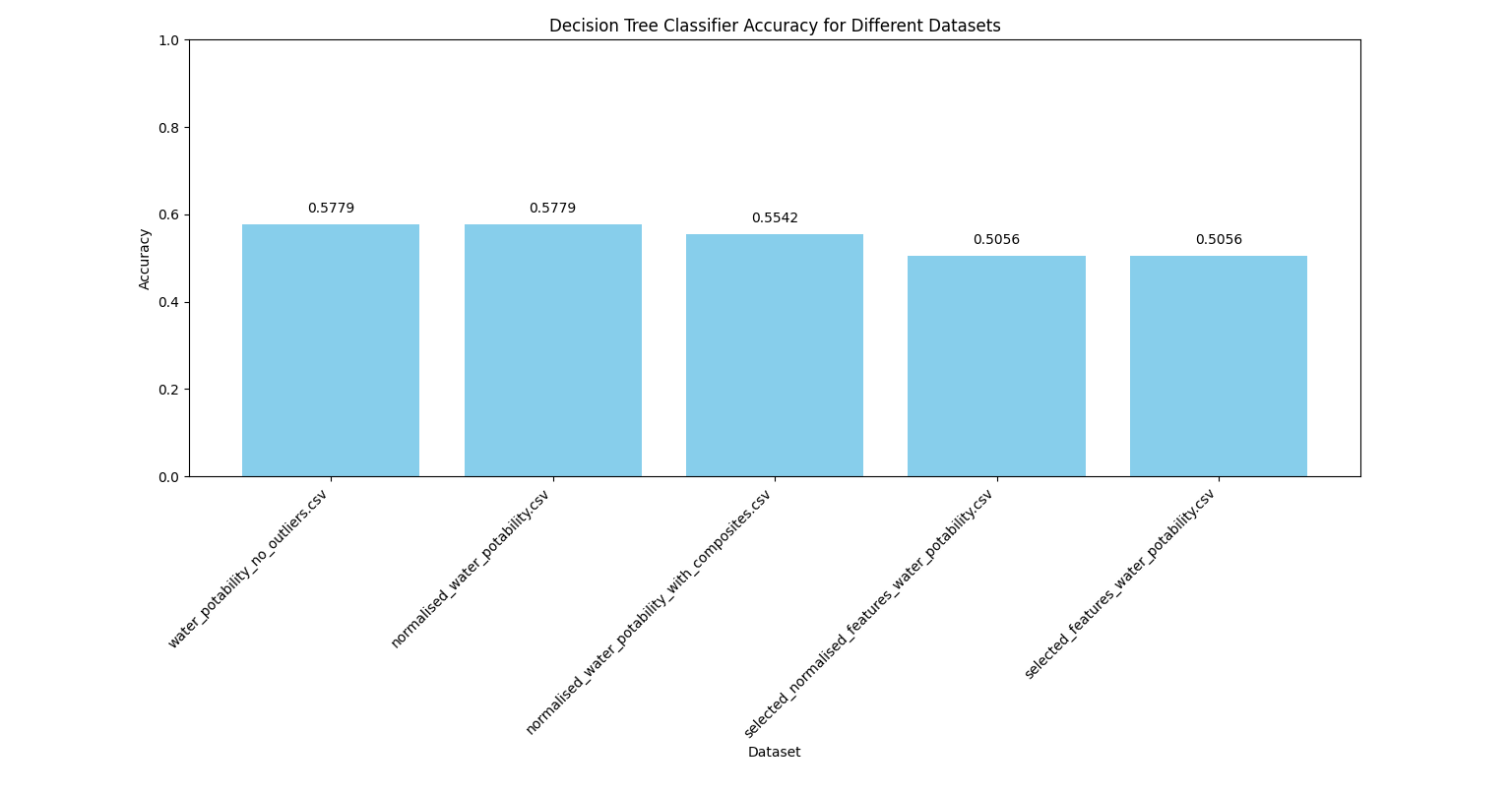
    yval = bar.get\_height()

    plt.text(bar.get\_x() + bar.get\_width()/2, yval + 0.02, f'{accuracy:.4f}', ha='center', va='bottom')

plt.show()

And my output:





# Summarisation

This is the summary for accuracies for different datasets:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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% |

Conclusion:

* Best Accuracy: The datasets "normalised\_water\_potability.csv" and "water\_potability\_no\_outliers.csv" yielded the best accuracy, both at about 0.5779
* Impact of Composite Features: "normalised\_water\_potability\_with\_composites.csv," the dataset containing composite features, has a bit worse accuracy of appro 0.5542, indicating that the model's performance was not appreciably enhanced by the addition of composite features.
* Feature Selection: For both the normalised and non-normalized selected feature datasets, the feature selection procedure (removing weakly correlated features) generated a lower accuracy of about 0.5056

# Appendix

Link to the code for Portfolio – Week 2:

* Studio 1 – No outliers data and EDA:

<https://github.com/trungkiennguyen22082004/COS40007_Artificial_Intelligence_for_Engineering/blob/main/Studios/studio1.py>

* Studio 2 – Model Development and related things: <https://github.com/trungkiennguyen22082004/COS40007_Artificial_Intelligence_for_Engineering/blob/main/Studios/studio2_portfolio.py>