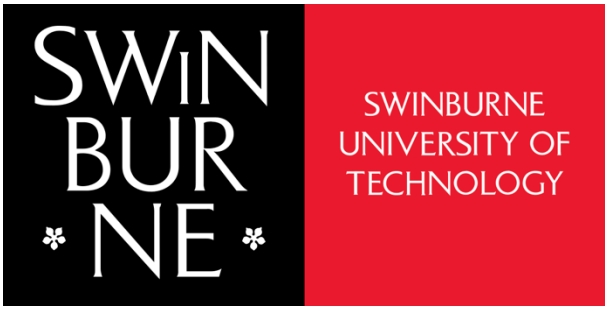

Artificial Intelligence for Engineering

Portfolio Assessment 1

Hello Machine Learning for Engineering

Xuan Tuan Minh Nguyen - 103819212
Studio 1-6

Selected Dataset in Studio 1

1. Selected dataset

In the studio 1, the dataset that I have chosen to go for the rest of the studio and the portfolio assessment is: **Water Quality in Water Engineering.**

2. Reason for the dataset selection

Since most of the provided datasets are for Engineering students, plus I am a Computer Science student, thus I chose the Water Quality dataset based on my personal interest. This dataset has just enough features (10 features in total) to get myself familiar with the process of creating a Machine Learning model. Moreover, water pollution is also one of my big concerns, especially with the downgrade of water quality in my home country. Playing around with this dataset has not only helped me gain a higher level of perspective on the overall process of creating a model, but it has also helped me increase my skill set of choosing the right feature for optimising the training process.

Implementation of Portfolio 1

1. Exploratory Data Analysis (EDA) process

After finished the Exploratory Data Analysis (EDA) process at Studio 1, I have concluded some of the most important aspects of the Water Quality dataset that will be the decider for the **Feature Engineering** step, which consist of:

- The dataset features mostly **right skew distribution**, with an exception for chloramines and sulfate, which are slightly **left skewed**.
- Except for the features of **pH, Solids, Chloramines and Turbidity**, the rest features do not prove **a feasible relationship** with the **target variable**, which is **Potability**. Thus making them **not suitable** for correlation-based decision making.
- On the other hand, the **organic carbon** and **pH** feature does have a **weak positive correlation** with each other. Thus combining them as a new feature (**Organic_carbon_ph**) would increase the prediction percentage for potability.
- Moreover, **Trihalomethanes and solids** also have a weak correlation with each other. Thus both features could be a new feature called **trihalomethanes_solids**.
- Finally, **Chloramines and Hardness** also have a low positive correlation with each other. Thus both features could be combined to be a new feature called **chloramines_hardness**.

2. Class Labelling / Developing ground truth data

After investigating the dataset for a bit, I believe that with the binary nature (0 and 1) of the target variable, which is **Potability**, has provided a clear class to classify the model, which makes relabel redundant.
Moreover, I have created a simple bar chart to provide a better visualisation for the distribution of both non-portable value (0), portable value (1) and any other values, for example NaN values.

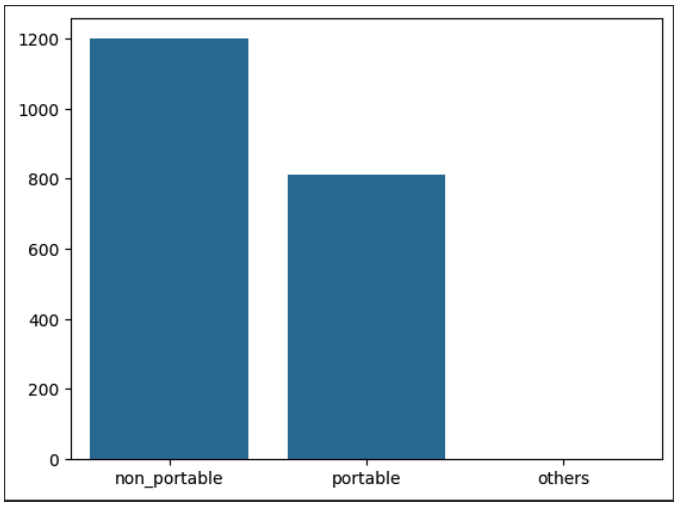

Figure 1: Bar chart for class distributions

Overall, this bar chart illustrates an imbalance in the potability class distributions:

- **Non-potable values:** Clearly shows a larger sample number, which indicates that more than 60% of the dataset are non-potable.
- **Potable values**: Shows a smaller sample than non-portable, indicating that less than 40% of the instances are potable water.
- **Other values**: Since the dataset has been properly cleaned, thus no samples could be found on this section.

3. Feature engineering

Convert target feature to categorical values

Since the target feature, which is **Potability**, is described in the form of binary. Thus converting it to categorical values is not necessary because the current value has already shown that 0 is non-potable value and 1 is potable value.

Normalisation

Normalising all of the features, with an exception for the target feature, will make sure that all of the numerical features will have the same scale from the range of 0 and 1, which boost the overall performance of the model. Once the normalisation

process is finished, it will be saved to a file by using the `to_csv()` function provided by pandas.

```
[ ] from sklearn.preprocessing import MinMaxScaler

[ ] normalised_df = outliered_df.copy()
    scaler = MinMaxScaler()
    normalise_features = [i for i in normalised_df.columns if i != 'Potability']
    normalised_df[normalise_features] = scaler.fit_transform(normalised_df[normalise_features])
    print("Transformed Data")
    normalised_df.head()
```

Figure 2: Normalise the values using `sklearn.MinMaxScaler()`

Transformed Data

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
0	0.6573391	0.6105962	0.4874640	0.6180119	0.6103397	0.3558469	0.7295712	0.8925786	0.6552776	0
1	0.7557987	0.4016907	0.3967126	0.4277278	0.3866565	0.4332130	0.3467532	0.0976235	0.5247641	0
2	0.3103718	0.4469718	0.6386689	0.5532984	0.4658068	0.1735752	0.1709601	0.3642212	0.1675714	0
3	0.8994825	0.8221792	0.6386920	0.5493410	0.7863052	0.1805838	0.4709457	0.7095171	0.1942731	0
4	0.6978530	0.5414555	0.2999522	0.1782153	0.3539968	0.6009516	0.3915864	0.4558844	0.6016891	0

Figure 3: Values after being normalised

```
normalised_df.to_csv(BASE_PATH + "/normalised_water_potability.csv", index=False)
```

Figure 4: Code to save normalised data

Covariance for Composite features

With the investigation from the Exploratory Data Analysis (EDA) process, I have appended four new features by combining **Organic carbon and pH** to be **Organic_carbon_ph**, **Chloramines and Hardness** to be **chloramines_hardness**, and **trihalomethanes_solids** as the weak positive correlation representation of **Trihalomethanes and Solids**. I have combined it using the covariance method and by using the built in `cov()` function from `numpy`.

```
[ ] normalised_df['Organic_carbon_ph'] = np.cov(normalised_df['Organic_carbon'], normalised_df['ph'])[0,1]
    normalised_df['chloramines_hardness'] = np.cov(normalised_df['Chloramines'], normalised_df['Hardness'])[0,1]
    normalised_df['trihalomethanes_solids'] = np.cov(normalised_df['Trihalomethanes'], normalised_df['Solids'])[0,1]

    normalised_df.head()
```

Figure 5: Composite features using Covariance

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability	Organic_carbon_ph	chloramines_hardness	trihalomethanes_solids
0	0.6573391	0.6105962	0.4874640	0.6180119	0.6103397	0.3558469	0.7295712	0.8925786	0.6552776	0	0.0009067	-0.0008935	-0.0006150
1	0.7557987	0.4016907	0.3967126	0.4277278	0.3866565	0.4332130	0.3467532	0.0976235	0.5247641	0	0.0009067	-0.0008935	-0.0006150
2	0.3103718	0.4469718	0.6386689	0.5532984	0.4658068	0.1735752	0.1709601	0.3642212	0.1675714	0	0.0009067	-0.0008935	-0.0006150
3	0.8994825	0.8221792	0.6386920	0.5493410	0.7863052	0.1805838	0.4709457	0.7095171	0.1942731	0	0.0009067	-0.0008935	-0.0006150
4	0.6978530	0.5414555	0.2999522	0.1782153	0.3539968	0.6009516	0.3915864	0.4558844	0.6016891	0	0.0009067	-0.0008935	-0.0006150

Figure 6: New features after finishing the composition

4. Feature selection

As analysed in the EDA process, except the features of **pH, Solids, Chloramines and Turbidity** are having enough effects to the **Potability**, all of the rest features have a very low relationship to the **Potability**. Therefore, I decided to create datasets with the selected features for both normalised and unnormalised datasets.

```
[ ] features_columns = ['ph', "Solids", "Chloramines", "Turbidity", "Potability", "Organic_carbon_ph", "chloramines_hardness", "trihalomethanes_solids"]
    columns_to_drop = [i for i in features_concrete_df.columns if i not in features_columns]
    features_concrete_df.drop(columns=columns_to_drop, inplace=True)
    features_concrete_df.head()
```

Figure 7: Feature selection for normalised dataset

```
[ ] features_columns = ['ph', "Solids", "Chloramines", "Turbidity", "Potability"]
    columns_to_drop = [i for i in outliered_df.columns if i not in features_columns]
    outliered_df.drop(columns=columns_to_drop, inplace=True)
    outliered_df.head()
```

Figure 8: Feature selection for unnormalised dataset

5. Model development

Throughout the Feature Engineering and Feature Selection process, I have generated four different datasets that has been both normalised and composited from the original dataset (**non_outliers_water_potability.csv**) in order to develop our Machine Learning model to train and test on those datasets, which includes:

- **normalised_water_potability.csv**: This is the dataset that contains all features with normalisation and without composite features.
- **features_water_potability.csv**: This is the dataset that contains all features with normalisation and with composite features.
- **selected_features_water_potability.csv**: This is the dataset that contains selected features performed in the Feature Selection process with normalisation and without composite features.
- **selected_features_normalised_water_potability.csv**: This is the dataset that contains selected features performed in the Feature Selection process with normalisation and with composite features.

Using the provided Decision Tree Classifier from `sklearn`, I have developed a Machine Learning model to train and validate the result from the given datasets.

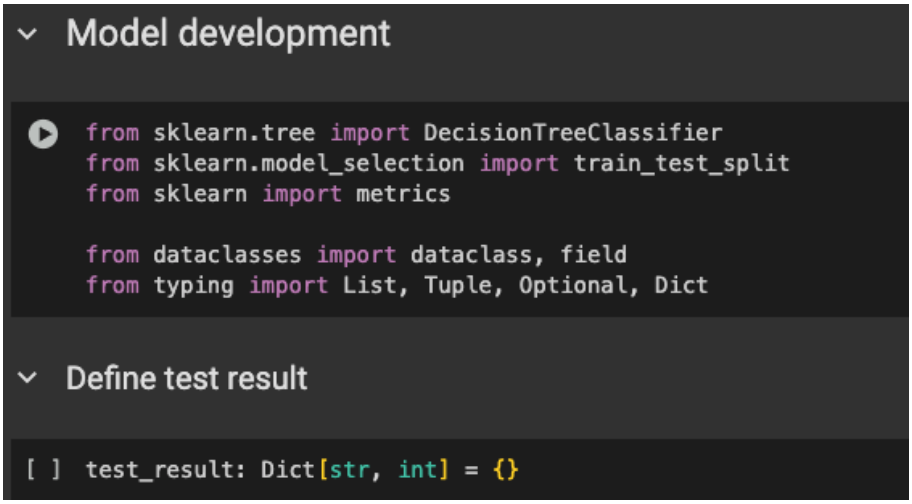

Figure 9: Import dependencies and create a dictionary to hold the prediction result

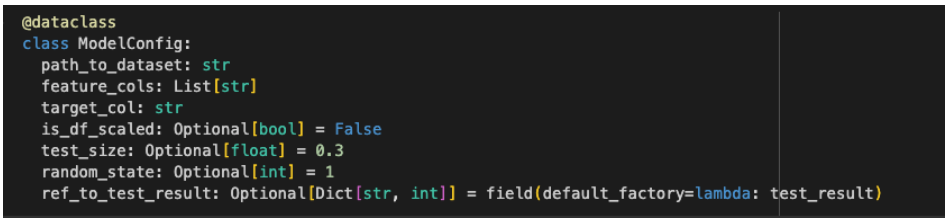

Figure 10: Define parameters for the `create_model()` function using `dataclass` annotation

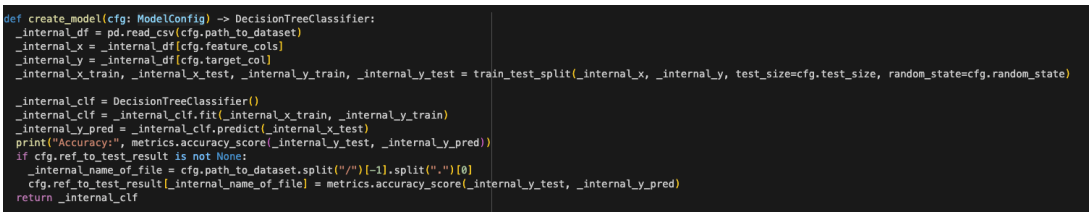

Figure 11: Reusable `create_model()` function to train and validate the `DecisionTreeClassifier` result

This is how the `create_model()` function works:

- First, it reads the csv from the provided `path_to_dataset` parameter and extracts values from the feature columns and target columns via `feature_cols` and `target_col` parameters.
- Secondly, it splits the data to train and test data using `sklearn.model_selection.train_test_split()` method with the extracted feature and target columns with the test size and random state from `test_state` and `random_state` parameters, by default, `test_state` will be 0.3 and `random_state` is 1.

- Next, the function will use the Decision Tree Classifier model from `sklearn.tree.DecisionTreeClassifier` to fit and predict data. Once the fit process is finished, it will predict and show the accuracy score by using the built in `sklearn.metrics.accuracy_score()` function.
- It then checks for where to store the accuracy score in `ref_to_test_result`, which in this case, our created `test_result` dictionary. If there is a place to store the accuracy store then it will store the accuracy score where the key is the name of the file and the value is the accuracy score. Finally, it returns the model for further usages.

Summarisation

1. *Test results*

For testing purposes, the given table illustrates the result of 5 different models with 5 datasets, of which one dataset is the unnormalised and no features selection / composition and the other fours are the one listed above.

```
1. non_outliers_water_potability: 59.93377483443708
2. normalised_water_potability: 60.76158940397352
3. features_water_potability: 61.258278145695364
4. selected_features_normalised_water_potability: 59.93377483443708
5. selected_features_water_potability: 59.27152317880795
```

Figure 12: Accuracy score of each model

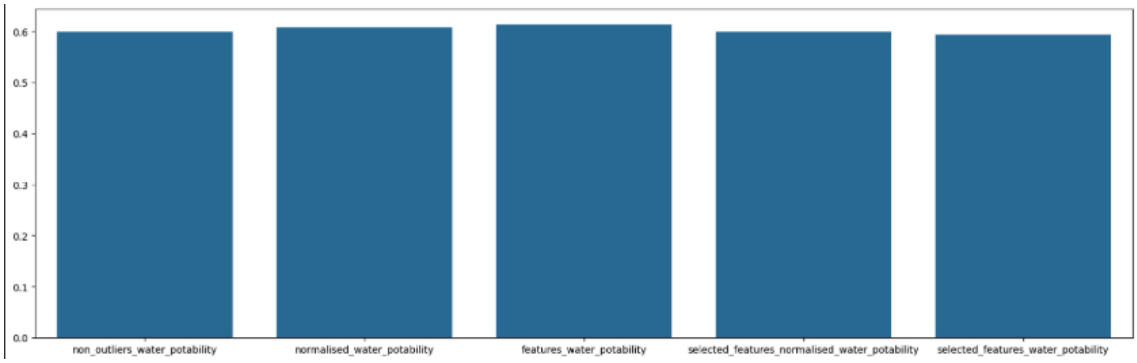

Figure 13: Visualising the accuracy score of each model using bar plot

Models	Accuracy Score
normalised_water_potability.csv	60.76%

selected_features_normalised_water_potability	59.93%
selected_features_water_potability	59.27%
features_water_potability	61.25%
non_outliers_water_potability	59.93%

Table 1: Comparison between model's accuracy score

2. *Observations*

Here are some of my observations based on the given accuracy score:

- **Best accuracy**: The two datasets that achieved the best accuracy are "normalised_water_potability.csv" and "features_water_potability.csv"
- **Impact of Compositions:** The dataset that included composite features, which are "selected_features_normalised_water_potability.csv" and "features_water_potability.csv" has achieved the score of 59.93% and 61.25%. Thus has shown the slight impact of creating composite features in this dataset.
- **Feature selection:** On the other hand, the two feature selection datasets, which are "selected_features_water_potability.csv" and "selected_features_normalised_water_potability.csv", has got a slightly less accuracy then other, which respectively achieved 59.27% and 59.93%. Therefore, this result shows that feature selection is not needed for this dataset.

Appendix

Source code for the Portfolio:

- **Week 1 studio:** https://github.com/cobeo2004/COS40007/tree/main/Studio%201
- **Week 2 studio:** https://github.com/cobeo2004/COS40007/tree/main/Studio%202
- **Portfolio 1:** https://github.com/cobeo2004/COS40007/tree/main/Assignment%201