Portfolio Week 4

COS40007 - Artificial Intelligence for Engineering
Studio 1-1 (12:30-2:30)

Hanh Nguyen Nguyen
103532674

Step 1: Data Preparation

1) The dataset has 2 constant value columns of TFE Product out temperature and TFE Steam temperature SP.
TFE Product out temperature has a constant value of 0 and TFE Steam temperature SP of 80.

```
train_df = pd.read_csv('vegemite_train.csv')
train_drop_df = train_df

train_drop_df = train_drop_df.drop(columns= ["TFE Product out temperature", "TFE Steam temperature SP"])
# print out # of unique values
print(train_df.nunique())
```

Figure 1. Source code for dropping constant value columns.

The unique values are checked using nunique() and double-checked the CSV file uses a sorting filter in Excel. After the columns are dropped, we checked again to make sure the program ran smoothly.

2) The dataset has a few columns with few unique integer values. Those are FFTE Feed tank level SP (25, 45, 50), TFE Motor speed (0, 20, 80), FFTE Pump 1 (0, 50, 70, 75, 80), FFTE Pump 1 - 2 (0, 85, 95, 100), and FFTE Pump 2 (0, 45, 65, 70, 81)
The values are converted into categorical values using 0, 1, 2, etc.

```
feedlevel_mapping = {25: 0, 45: 1, 50: 2}
motorspeed_mapping = {0: 0, 20: 1, 80: 2}
pump1_mapping = {0: 0, 50: 1, 70: 2, 75: 3, 80: 4}
pump12_mapping = {0: 0, 85: 1, 95: 2, 100: 3}
pump2_mapping = {0: 0, 45: 1, 65: 2, 70: 3, 81: 4}

train_drop_df['FFTE Feed tank level SP'] = train_drop_df['FFTE Feed tank level SP'].map(feedlevel_mapping)
train_drop_df['TFE Motor speed'] = train_drop_df['TFE Motor speed'].map(motorspeed_mapping)
train_drop_df['FFTE Pump 1'] = train_drop_df['FFTE Pump 1'].map(pump1_mapping)
train_drop_df['FFTE Pump 1 - 2'] = train_drop_df['FFTE Pump 1 - 2'].map(pump12_mapping)
train_drop_df['FFTE Pump 2'] = train_drop_df['FFTE Pump 2'].map(pump2_mapping)

print(train_drop_df)
print(train_drop_df.nunique())
print(train_drop_df['Class'].value_counts())
train_drop_df.to_csv('vegemite_converted_train.csv', index = False)
print(train_drop_df)
```

Figure 2. Source code for converting few-integer columns to categorical values.

3) The classes in the original data frame didn't have a balanced distribution. The original distribution was 2459 data points from class 0, 4724 from class 1, and 7054 from class 2. SMOTE was used to oversampling the minority classes and after implementing SMOTE, the dataset ended up with 7054 data points from each class.

```
df = pd.read_csv('vegemite_converted_train.csv')

feature_cols = df.loc[:, df.columns != 'Class']
X = feature_cols
Y = df['Class']

smote = SMOTE(random_state=42)
X_resampled, Y_resampled = smote.fit_resample(X,Y)
resampled_df = pd.concat([pd.DataFrame(X_resampled, columns =X.columns), pd.Series(Y_resampled, name= 'Class')], axis=1)

resampled_df.to_csv("vegemite_resampled.csv", index = 0)
print(resampled_df)
print(resampled_df['Class'].value_counts())
# distribution is 0: 2459, 1: 4724, 2: 7054
```

Figure 3. Source code for oversampling minority classes using SMOTE

4) Class is chosen as the Target variable since the project's outcome is to assess the quality (or consistency level) of vegemite, which is what the class values resemble in this data frame. To examine the correlation, heat map is used to visualise the relationship of every variables to another.

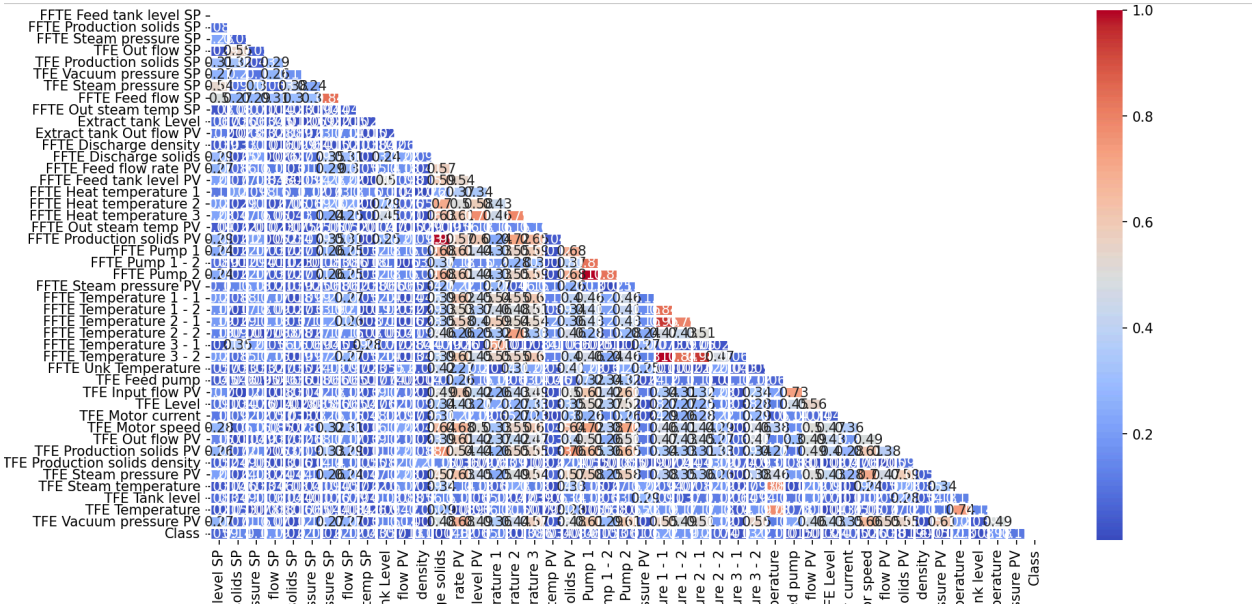

Figure 4. Overall heatmap demonstrates the correlation of class with other features

Through the heatmap exploration (without threshold filter), it is noticeable that the correlation of the Class target variable with different features is relatively low, the maximum value is 0.22 of FFTE Temperature 1-1, where the desired threshold is around 0.4-0.5. This explains the need to add composite features in the dataset to demonstrate a better correlation between class and other features.

Composite features are generated by examining the relationship of other features to one another by applying a filter so the heat map will only show a correlation of 0.5 and above.

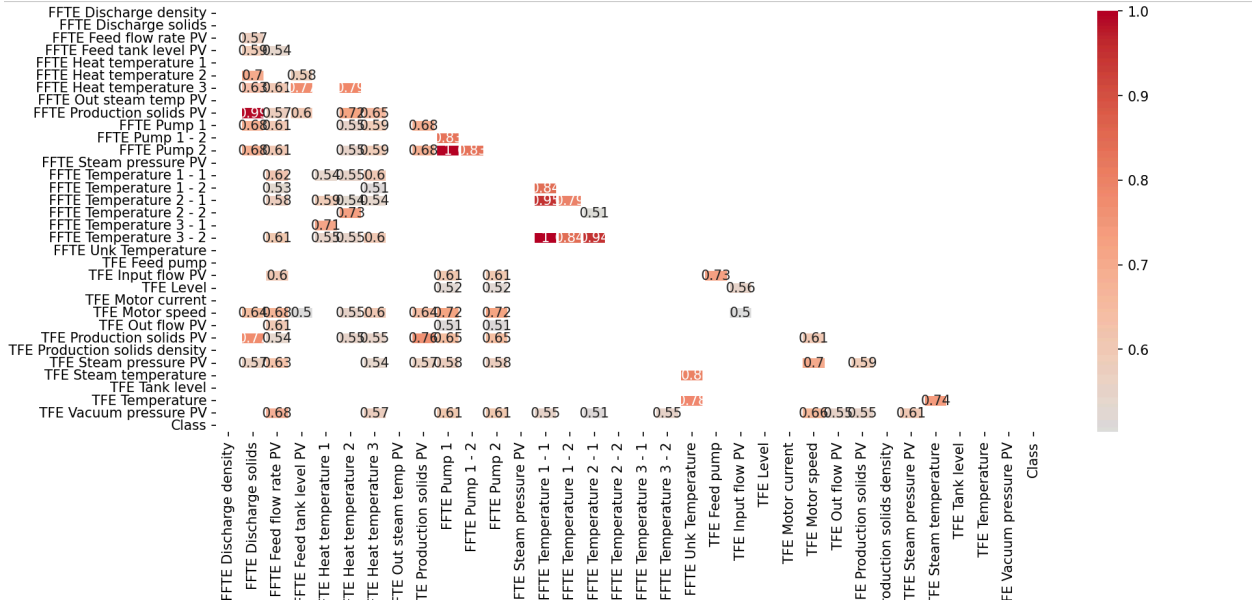

Figure 5. Overall heatmap demonstrates the correlation of class with other features with a filter so that only the correlation of 0.5 and above will be shown.

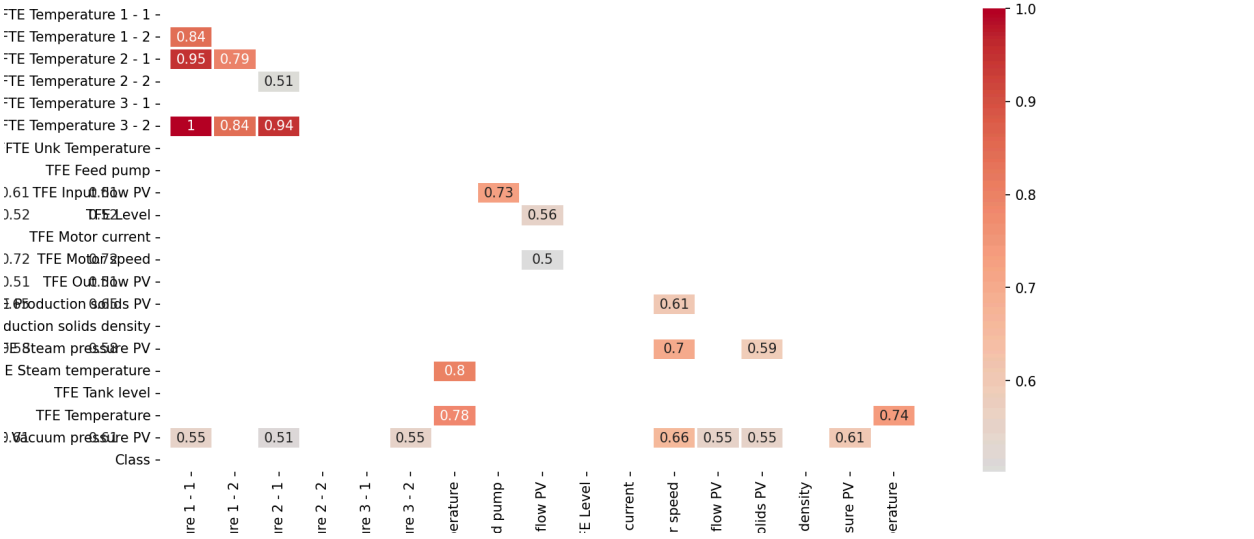

Figure 6. Heatmap zoom in (1)

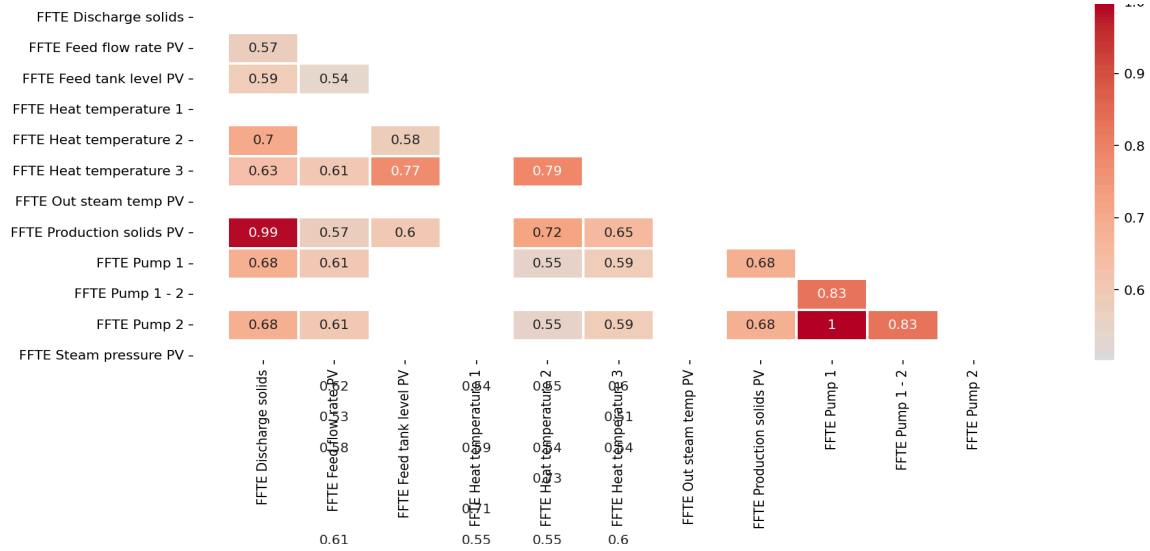

Figure 7. Heatmap zoom in (2)

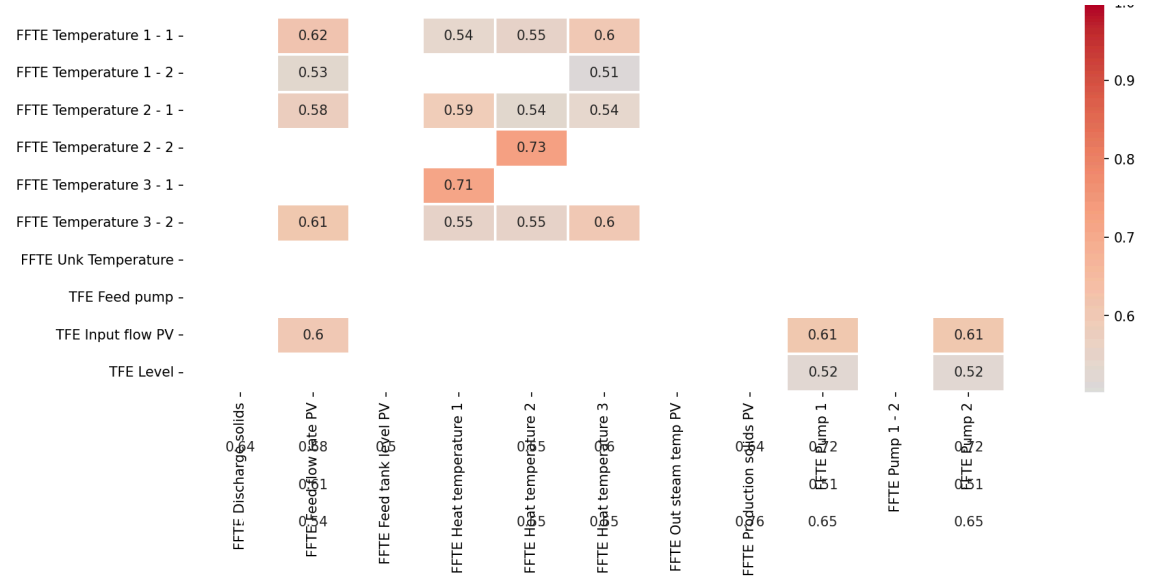

Figure 8. Heatmap zoom in (3)

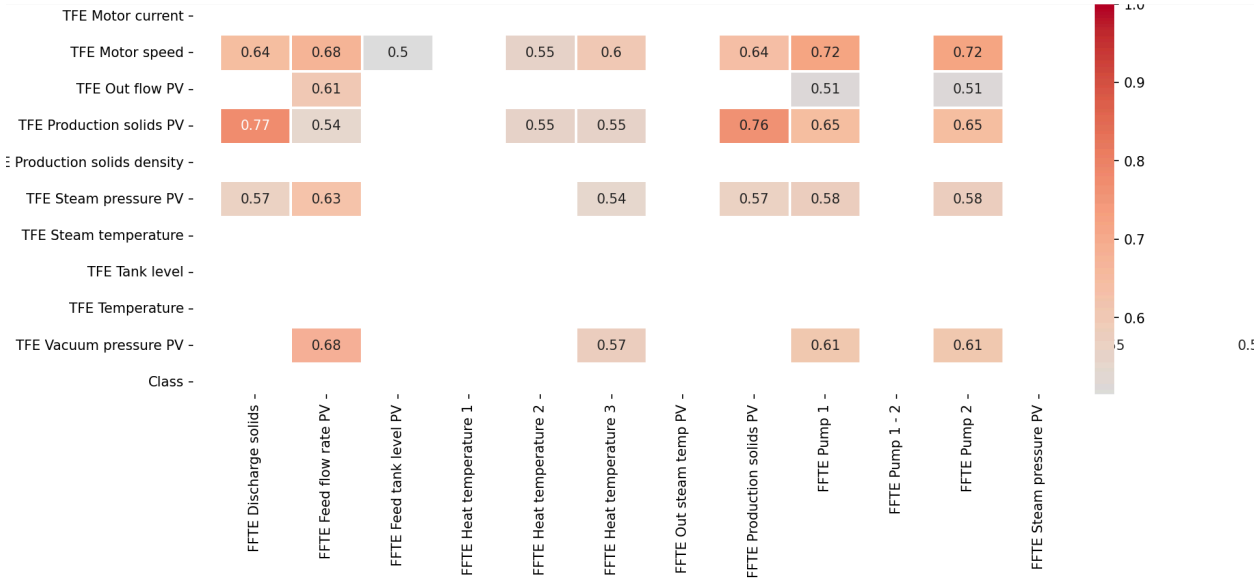

Figure 9. Heatmap zoom in (4)

After examining the heatmap and variables' relationships with one another, the composite features are decided to be: Pump Mean, Solid and Density Mean, Temperature Mean, Pressure Mean, and Pressure and Motor Speed covariance.
Meaning that the final data set has a total of 50 features.

```
resampled_df = pd.read_csv("vegemite_resampled.csv")

solid_density_columns = ['FFTE Production solids SP','FFTE Discharge density', 'FFTE Production solids PV',
                         'TFE Production solids PV', 'TFE Production solids SP', 'FFTE Discharge solids',
                         'TFE Production solids density']

resampled_df['Solid and Density Mean'] = resampled_df[solid_density_columns].mean(axis=1)

# mean pump 1 + pump 2 + pump 12

resampled_df['Pump Mean'] = resampled_df[['FFTE Pump 1', 'FFTE Pump 1 - 2', 'FFTE Pump 2']].mean(axis=1)

# pressure / motor speed
pressure_column = ['FFTE Steam pressure SP', 'TFE Vacuum pressure SP', 'TFE Steam pressure SP',
                   'FFTE Steam pressure PV','TFE Steam pressure PV', 'TFE Vacuum pressure PV']
resampled_df['Pressure Mean'] = resampled_df[pressure_column].mean(axis=1)

pressure_motor_cov = resampled_df[['Pressure Mean', 'TFE Motor speed']].cov().iloc[0, 1]
resampled_df['Pressure and Motor Speed covariance'] = pressure_motor_cov

# temp
temperature_column = ['FFTE Heat temperature 1', 'FFTE Heat temperature 2', 'FFTE Heat temperature 3',
                      'FFTE Temperature 1 - 1', 'FFTE Temperature 1 - 2',
                      'FFTE Temperature 2 - 1', 'FFTE Temperature 2 - 2',
                      'FFTE Temperature 3 - 1', 'FFTE Temperature 3 - 2']

resampled_df['Temperature Mean'] = resampled_df[temperature_column].mean(axis=1)
```

Figure 10. Source code for composite features generation.

Step 2: Feature selection, Model Training and Evaluation

6) Selected features are chosen using SelectKBest for the 25 best features.

```
Index(['FFTE Feed tank level SP', 'FFTE Production solids SP',
       'FFTE Steam pressure SP', 'TFE Out flow SP', 'TFE Production solids SP',
       'TFE Vacuum pressure SP', 'FFTE Feed flow SP', 'FFTE Out steam temp SP',
       'FFTE Discharge density', 'FFTE Feed flow rate PV',
       'FFTE Feed tank level PV', 'FFTE Heat temperature 1',
       'FFTE Heat temperature 3', 'FFTE Temperature 1 - 1',
       'FFTE Temperature 1 - 2', 'FFTE Temperature 2 - 1',
       'FFTE Temperature 2 - 2', 'FFTE Temperature 3 - 2',
       'FFTE Unk Temperature', 'TFE Motor current', 'TFE Steam temperature',
       'TFE Temperature', 'TFE Vacuum pressure PV', 'Pressure Mean',
       'Temperature Mean'],
```

Figure 11. List of 25 chosen features using SelectKBest.

```python
resampled_df = pd.read_csv('vegemite_features.csv')

resampled_df = resampled_df.astype(float)
feature_cols = resampled_df.loc[:, resampled_df.columns != 'Class']
X = feature_cols
Y = resampled_df['Class']

selector = SelectKBest(f_classif, k=25)
X_feature = selector.fit_transform(X, Y)

selected_features = selector.get_support()
selected_features = feature_cols.columns[selected_features]

selected_features_df = resampled_df[selected_features]
selected_features_df.to_csv('vegemite_selected_features.csv', index=False)
# print(selected_features_df)
```

Figure 12. Source code for choosing best features using SelectKBest.

7) Train with multiple models, generate a classification report and confusion matrix

```
Decision Tree:
              precision    recall  f1-score   support

           0       0.96      0.97      0.97      2073
           1       0.95      0.95      0.95      2114
           2       0.96      0.96      0.96      2162

    accuracy                           0.96      6349
   macro avg       0.96      0.96      0.96      6349
weighted avg       0.96      0.96      0.96      6349

Decision Tree confusion matrix:
 [[2011   38   24]
 [  50 2001   63]
 [  23   73 2066]]
```

```
SVM:
              precision    recall  f1-score   support

           0       0.40      0.79      0.53      2073
           1       0.64      0.03      0.06      2114
           2       0.51      0.51      0.51      2162

    accuracy                           0.44      6349
   macro avg       0.52      0.44      0.37      6349
weighted avg       0.52      0.44      0.37      6349

SVM confusion matrix:
 [[1632    0  441]
 [1411   62  641]
 [1020   35 1107]]
```

```
SGD:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00      2073
           1       0.00      0.00      0.00      2114
           2       0.34      1.00      0.51      2162

    accuracy                           0.34      6349
   macro avg       0.11      0.33      0.17      6349
weighted avg       0.12      0.34      0.17      6349

SGD confusion matrix:
 [[   0    0 2073]
 [   0    0 2114]
 [   0    0 2162]]
```

```
RandomForest:
              precision    recall  f1-score   support

           0       0.99      1.00      0.99      2073
           1       0.99      0.99      0.99      2114
           2       1.00      0.99      0.99      2162

    accuracy                           0.99      6349
   macro avg       0.99      0.99      0.99      6349
weighted avg       0.99      0.99      0.99      6349

Random Forest confusion matrix:
 [[2067    6    0]
 [  18 2087    9]
 [   3   10 2149]]
```

```
MLP:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00      2073
           1       0.67      0.04      0.08      2114
           2       0.34      0.99      0.51      2162

    accuracy                           0.35      6349
   macro avg       0.34      0.34      0.20      6349
weighted avg       0.34      0.35      0.20      6349

MLP confusion matrix:
 [[   0   20 2053]
 [   0   93 2021]
 [   0   25 2137]]
```

Figure 13. Confusion matrix and classification report

Table 1. Different evaluation measures of different models summary comparison table

Model	Accuracy	F1-score	Precision	Recall
Decision Tree	96%	96%	96%	96%
SVM	44%	37%	52%	44%
SGD	34%	17%	12%	34%
RandomForest	99%	99%	99%	99%
MLP	35%	20%	34%	35%

The chosen model for the next steps is RandomForest. There could be a slight chance that the model is overfitting but overall, considering all the pros and cons of RandomForest compared to Decision Tree (the second most accurate model), RandomForest is a better option for machine learning and AI development [1].

11) Saving model

The chosen model is performed on the original dataset that has 14000+ datapoints with SMOTE implemented to oversample the minority classes to balance the class distribution, and conversion of few-integer columns to categorical. After performing with that data set, the model is saved as follows:

```
df = pd.read_csv('vegemite_resampled.csv')
X = df.loc[:, df.columns != 'Class']
Y = df['Class']

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=1)
# random forest
rf_clf = RandomForestClassifier(random_state=1)
rf_clf = rf_clf.fit(X_train, Y_train)
rf_Y_pred = rf_clf.predict(X_test)

rf_cm = confusion_matrix(Y_test, rf_Y_pred, labels = rf_clf.classes_)
rf_disp = ConfusionMatrixDisplay(confusion_matrix = rf_cm,
                                 display_labels = rf_clf.classes_)

rf_filename = 'vegemite_randomforest_model.pkl'
pickle.dump(rf_clf, open(rf_filename, 'wb'))
```

Figure 14. Source code for saving the model.

[1] S. Talari, "Random forest® vs decision tree: Key differences - kdnuggets," KDnuggets, 2022. Available: https://www.kdnuggets.com/2022/02/random-forest-decision-tree-key-differences.html

```
Random Forest accuracy: 0.9938573003622617
RandomForest:
              precision    recall  f1-score   support

           0       0.99      1.00      0.99      2073
           1       0.99      0.99      0.99      2114
           2       1.00      0.99      1.00      2162

    accuracy                           0.99      6349
   macro avg       0.99      0.99      0.99      6349
weighted avg       0.99      0.99      0.99      6349

Random Forest confusion matrix:
 [[2067    6    0]
 [  14 2092    8]
 [   1   10 2151]]
```

Figure 15. Chosen model (Random Forest) result.

Step 3: ML to AI

The same result is observed when performing other models using these 1000 data points

Table 2. Testing accuracy of different models summary comparison table

Model	Accuracy
Decision Tree	98.7%
SVM	42.7%
SGD	35.5%
RandomForest	99.6%
MLP	35.7%

Other models are tested using the following source code:

```
df = pd.read_csv('vegemite_test_resampled.csv')

X = df.loc[:, df.columns != 'Class']
Y = df['Class']

rf_counter = 0
for i in range(len(X)):
    x_i = X.iloc[i, :].values.reshape(1, -1)
    y_i = Y.iloc[i]

    # Random Forest
    y_pred_rf = rf_model.predict(x_i)[0]
    if y_pred_rf == y_i:
        rf_counter += 1

rf_accuracy = rf_counter / len(Y)
print('Random Forest accuracy with 1000 testing data points:', rf_accuracy)
```

Figure 16. Source code for only Random Forest (chosen) model testing.

Step 4: Developing rules from ML model

```python
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.tree import export_text

df = pd.read_csv('vegemite_selected_features.csv')

sp_features = [col for col in df.columns if col.endswith('SP')]
sp_df = df[sp_features]

X_train, X_test, y_train, y_test = train_test_split(sp_df, df['Class'], test_size=0.3, random_state=42)

sp_tree_model = DecisionTreeClassifier(random_state=42)
sp_tree_model.fit(X_train, y_train)

tree_rules = export_text(sp_tree_model, feature_names=sp_features)
print(tree_rules)
```

Figure 17. Source code for selecting only SP features.

Decision Tree structure is then generated and the rules are listed as follows:

Table 3. Decision Tree SP value rules for each class.

Class	Rule	Condition
0	1	TFE Vacuum pressure SP > -76.93
	2	TFE Production solids SP > 37.69
1	1	TFE Production solids SP > 84.00 FFTE Steam pressure SP > 109.50
	2	TFE Production solids SP > 84.00 TFE Vacuum pressure SP > -58.38 FFTE Steam pressure SP > 73.50 FFTE Steam pressure SP > 110.96
	3	TFE Production solids SP > 84.00 TFE Vacuum pressure SP > -58.38 FFTE Feed flow SP <= 10030.00
	4	TFE Out flow SP <= 2068.53 FFTE Out steam temp SP <= 50.11 FFTE Steam pressure SP <= 114.35

2	1	TFE Vacuum pressure SP > -57.26
	2	TFE Production solids SP <= 48.52 FFTE Feed flow SP <= 9460.69 TFE Out flow SP <= 1991.59
	3	FFTE Steam pressure SP <= 110.96
	4	TFE Vacuum pressure SP > -42.68

Source Code

Guide:

- PORTFOLIO4.py is for tasks related to Step 1
- Portfolio4_STEP2.py is for tasks in Step 2 (from part 6-10)
- Portfolio4_SavingMLmodel.py is for Step 2 (part 11)
- Portfolio4_TestingAIModel.py is for Step 3
- Portfolio4_step4.py is for tasks in Step 4

Resource:

- Source code
- Data set
- Decision Tree structure

Appendix

Model results on the original dataset

Decision Tree:

	precision	recall	f1-score	support
0	0.97	0.98	0.97	2073
1	0.96	0.95	0.96	2114
2	0.97	0.96	0.97	2162
accuracy			0.96	6349
macro avg	0.96	0.96	0.96	6349
weighted avg	0.96	0.96	0.96	6349

Decision Tree confusion matrix:

```
[[2028   25   20]
 [  51 2017   46]
 [  19   63 2080]]
```

SVM:

	precision	recall	f1-score	support
0	0.41	0.85	0.55	2073
1	0.66	0.03	0.06	2114
2	0.57	0.51	0.54	2162
accuracy			0.46	6349
macro avg	0.55	0.46	0.38	6349
weighted avg	0.55	0.46	0.38	6349

SVM confusion matrix:

```
[[1764    1  308]
 [1516   65  533]
 [1028   32 1102]]
```

SGD:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	2073
1	0.33	1.00	0.50	2114
2	0.85	0.01	0.02	2162
accuracy			0.34	6349
macro avg	0.39	0.34	0.17	6349
weighted avg	0.40	0.34	0.17	6349

SGD confusion matrix:

```
[[   0 2069    4]
[   0 2114    0]
[   4 2136   22]]
```

RandomForest:

	precision	recall	f1-score	support
0	0.99	0.99	0.99	2073
1	0.99	0.99	0.99	2114
2	0.99	0.99	0.99	2162
accuracy			0.99	6349
macro avg	0.99	0.99	0.99	6349
weighted avg	0.99	0.99	0.99	6349

Random Forest confusion matrix:

```
[[2060    9    4]
[  14 2088   12]
[   4   10 2148]]
```

MLP:

	precision	recall	f1-score	support
0	0.56	0.16	0.25	2073
1	0.35	0.92	0.51	2114
2	0.68	0.06	0.11	2162
accuracy			0.38	6349

macro avg	0.53	0.38	0.29	6349
weighted avg	0.53	0.38	0.29	6349

MLP confusion matrix:
[[331 1711 31]
[132 1954 28]
[130 1908 124]]

Random Forest accuracy with 1000 testing data points: 0.994
Decision Tree accuracy with 1000 testing data points: 0.991
SVM accuracy with 1000 testing data points: 0.427
SGD accuracy with 1000 testing data points: 0.414
MLP accuracy with 1000 testing data points: 0.393