Student name: Nguyen Anh Tuan
Student ID: 103805949

Swinburne University of Technology

COS40007-Artificial Intelligence for Engineering

Portfolio 4

Nguyen Anh Tuan - 103805949

Studio class: Studio 1-1

Hanoi, Vietnam

Student name: Nguyen Anh Tuan
Student ID: 103805949

A. Step 1: Data Preparation

- Shuffle dataset:

In [2]:
```
dataset = pd.read_csv('/kaggle/input/vegemite.csv')
```

In [3]:
```
shuffled_df = dataset.sample(n=len(dataset))

# Reset the index
shuffled_df = shuffled_df.reset_index(drop=True)
```

- Randomly take out 1000 data points (rows) such as way that each class in those 1000 samples has near equal distribution (e.g. at least 300 samples from each class)

Student name: Nguyen Anh Tuan
Student ID: 103805949

]:
```
# Sample 1000 data points, ensuring equal class distribution
class0 = shuffled_df[shuffled_df['Class'] == 0].sample(n=333, random_state= 1)
class1 = shuffled_df[shuffled_df['Class'] == 1].sample(n=334, random_state = 1)
class2 = shuffled_df[shuffled_df['Class'] == 2].sample(n=333, random_state = 1)

sample_df = pd.concat([class0, class1, class2])

remaining_data_points = shuffled_df.drop(sample_df.index)
```

In [5]:
```
shuffled_df.head()
```

Out[5]:

	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	TFE Steam pressure SP	TFE Steam temperature SP	FFTE Feed flow SP	FFTE Out steam temp SP	...
0	50.0	43.00	130.0	2609.30	65.0	-80.00	120.0	80.0	10000.0	50.40	...
1	25.0	43.00	130.0	2255.81	51.0	-40.74	120.0	80.0	9500.0	50.00	...
2	50.0	40.50	104.0	2296.30	63.0	-79.78	125.0	80.0	9300.0	40.71	...
3	50.0	41.11	125.0	2506.91	60.0	-75.90	125.0	80.0	9220.0	50.00	...
4	50.0	43.00	128.0	2846.51	69.0	-78.63	120.0	80.0	10200.0	50.00	...

5 rows × 47 columns

In [6]:
```
sample_df.to_csv('1000_data_points.csv')
```

1) Does the dataset have any constant value column? If yes, then remove them

Yes, the dataset has constant value columns. Here is how I remove these columns

Student name: Nguyen Anh Tuan
Student ID: 103805949

Remove constant value columns

In [7]:
```
from sklearn.model_selection import train_test_split

X = remaining_data_points.iloc[:, :-1]
y = remaining_data_points.iloc[:, -1]
```

In [8]:
```
constant_features = remaining_data_points.columns[remaining_data_points.nunique() ==
1]
print(constant_features)
```

```
Index(['TFE Steam temperature SP', 'TFE Product out temperature'], dtype='object')
```

In [9]:
```
# Remove constant value columns
remaining_data_points = remaining_data_points.loc[:, remaining_data_points.nunique() >
1]
```

2) Does the dataset have any column with few integer values? If yes, then convert them to categorial feature

Yes, it does have. Here is how I convert them to categorial feature

Convert columns with few integer values to categorial feature.

In [10]:
```
few_int_values_columns = []
for col in X.columns:
    if X[col].nunique() < 10:
        few_int_values_columns.append(col)
print(few_int_values_columns)
```

```
['FFTE Feed tank level SP', 'TFE Steam temperature SP', 'FFTE Pump 1', 'FFTE Pump 1
- 2', 'FFTE Pump 2', 'TFE Motor speed', 'TFE Product out temperature']
```

In [11]:
```
X[few_int_values_columns] = X[few_int_values_columns].astype('category')
print(X.dtypes)
```

```
FFTE Feed tank level SP          category
FFTE Production solids SP         float64
FFTE Steam pressure SP            float64
TFE Out flow SP                   float64
TFE Production solids SP          float64
TFE Vacuum pressure SP            float64
```

Student name: Nguyen Anh Tuan
Student ID: 103805949

3) Does the class have a balanced distribution? If not, perform the necessary undersampling and oversampling or adjust the class weights.

No, it does not have. To balance the class distribution, I use SMOTETomek method, which combines SMOTE (Synthetic Minority Over-sampling Technique) and Tomek Links (an undersampling method) to balance the class distribution by both oversampling the minority class and undersampling the majority class.

```
# Class balancing distribution using oversampling (SMOTE)
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import TomekLinks
from imblearn.combine import SMOTETomek

smote = SMOTETomek(smote=SMOTE(random_state=1), tomek=TomekLinks(sampling_strategy='ma
jority'))

# Fit the resampled data
X_resampled, y_resampled = smote.fit_resample(X, y)

# Split the resampled data into training
X_train_resampled, X_test, y_train_resampled, y_test = train_test_split(X_resampled, y
_resampled, test_size=0.2, stratify=y_resampled, random_state=1)
```

4) Do you find any composite features through exploration? If so, add them to the dataset.

I combine existing variables for these features to provide more insight into the system's performance. The Energy Efficiency Ratio measures the efficiency of solid production relative to steam pressure. The Flow Rate Balance shows the difference between the system’s input and output flows. The Temperature Gradient calculates the difference in steam temperatures between input and output, indicating heat retention or loss. The Vacuum Efficiency Index assesses how efficiently vacuum pressure is applied relative to the material flow. These composite features help identify potential inefficiencies or imbalances in the system.

In [16]:
```
# 1. Energy Efficiency Ratio
remaining_data_points['Energy_Efficiency'] = remaining_data_points['TFE Production solids PV'] / remaining_data_points['TFE Steam pressure PV']

# 2. Flow Rate Balance
remaining_data_points['Flow_Balance'] = remaining_data_points['TFE Out flow PV'] - remaining_data_points['FFTE Feed flow rate PV']

# 3. Temperature Gradient
remaining_data_points['Temp_Gradient'] = remaining_data_points['TFE Steam temperature'] - remaining_data_points['FFTE Out steam temp PV']

# 4. Vacuum Efficiency Index
remaining_data_points['Vacuum_Efficiency'] = remaining_data_points['TFE Vacuum pressure PV'] / remaining_data_points['TFE Out flow PV']
```

In [17]:
```
remaining_data_points.head()
```

Out[17]:

eam erature	TFE Tank level	TFE Temperature	TFE Vacuum pressure PV	Class	Energy_Efficiency	Flow_Balance	Temp_Gradient	Vacuum_Efficiency
	84.67	74.0	-72.11	1	0.528202	-7996.75	17.37	-0.045515
	81.78	72.0	-73.18	2	0.581436	-8272.37	14.06	-0.042728
	83.69	79.0	-77.39	1	0.348955	-8734.76	13.27	-0.156609
	80.29	76.0	-67.86	0	0.626195	-7292.47	25.35	-0.023316
	84.86	70.0	-79.13	2	0.489083	-8068.09	20.12	-0.051857

5 rows × 49 columns

5) Finally, how many features do you have in your final dataset?

There are 46 features in the dataset.

- Source code: https://www.kaggle.com/code/twananguyen/port4-submit
- Data: https://drive.google.com/drive/folders/1ikjQfoRPKgVlipk1HzDVEOVj8BZFCTLQ?usp=sharing

Student name: Nguyen Anh Tuan
Student ID: 103805949

B. Step 2: Feature selection, model training and evaluation

6) Does the training process need all features? If not, can you apply some feature selection techniques to remove some features? Justify your reason of feature selection.

No, the training process does not need all the features. Feature selection techniques such as removing constant value columns, correlation analysis, and feature importance ranking were applied to eliminate irrelevant and redundant features. This improves model performance, reduces overfitting, and enhances computational efficiency.

7) Train multiple ML models (at least 5 including DecisionTreeClassifier) with your selected features.

For the model training, I use these models: DecisionTreeClassifier, RandomForestClassifier, GradientBoostingClassifier, KNeighborsClassifier, SVM

[20]:
```
# Handle missing values manually
imputer = SimpleImputer(strategy='mean')
X_train_imputed = imputer.fit_transform(X_train_resampled)
X_test_imputed = imputer.transform(X_test)

# Models to train
models = {
    'DecisionTree': DecisionTreeClassifier(),
    'RandomForest': RandomForestClassifier(),
    'GradientBoosting': GradientBoostingClassifier(),
    'SVM': SVC(),
    'KNN': KNeighborsClassifier()
}
```

8) Evaluate each model with classification report and confusion matrix

Student name: Nguyen Anh Tuan
Student ID: 103805949

]:
```python
# Train and evaluate the models
results = {}
for model_name, model in models.items():
    model.fit(X_train_imputed, y_train_resampled)
    predictions = model.predict(X_test_imputed)

    # Get the classification report and confusion matrix
    report = classification_report(y_test, predictions)
    cm = confusion_matrix(y_test, predictions)

    # Store the results
    results[model_name] = {
        'report': report,
        'confusion_matrix': cm
    }

# Print the results in the desired format
for model_name, result in results.items():
    print(f"Classification test_report for {model_name}:")
    print(result['report'])
    print(f"Confusion Matrix for {model_name}:")
    print(result['confusion_matrix'])
    print("\n" + "="*50 + "\n")
```

```
Classification test_report for DecisionTree:
              precision    recall  f1-score   support

           0       0.97      0.97      0.97      1437
           1       0.96      0.95      0.96      1443
           2       0.97      0.97      0.97      1443

    accuracy                           0.97      4323
   macro avg       0.97      0.97      0.97      4323
weighted avg       0.97      0.97      0.97      4323

Confusion Matrix for DecisionTree:
[[1400   28    9]
 [  31 1378   34]
 [   9   30 1404]]

--------------------------------------------------
```

Student name: Nguyen Anh Tuan
Student ID: 103805949

```
Classification test_report for RandomForest:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1437
           1       1.00      0.99      1.00      1443
           2       1.00      1.00      1.00      1443

    accuracy                           1.00      4323
   macro avg       1.00      1.00      1.00      4323
weighted avg       1.00      1.00      1.00      4323

Confusion Matrix for RandomForest:
[[1435    1    1]
 [   5 1433    5]
 [   0    3 1440]]

==================================================

  Classification test_report for GradientBoosting:
              precision    recall  f1-score   support

           0       0.95      0.97      0.96      1437
           1       0.92      0.92      0.92      1443
           2       0.94      0.91      0.93      1443

    accuracy                           0.94      4323
   macro avg       0.94      0.94      0.94      4323
weighted avg       0.94      0.94      0.94      4323

  Confusion Matrix for GradientBoosting:
  [[1397   20   20]
   [  52 1329   62]
   [  26   99 1318]]
```

Student name: Nguyen Anh Tuan
Student ID: 103805949

```
Classification test_report for SVM:
              precision    recall  f1-score   support

           0       0.42      0.82      0.56      1437
           1       0.45      0.06      0.10      1443
           2       0.54      0.51      0.53      1443

    accuracy                           0.46      4323
   macro avg       0.47      0.46      0.39      4323
weighted avg       0.47      0.46      0.39      4323

Confusion Matrix for SVM:
[[1175   23  239]
 [ 975   80  388]
 [ 627   74  742]]
-------------------------------------------------

Classification test_report for KNN:
              precision    recall  f1-score   support

           0       0.85      0.95      0.90      1437
           1       0.86      0.87      0.86      1443
           2       0.90      0.78      0.83      1443

    accuracy                           0.87      4323
   macro avg       0.87      0.87      0.87      4323
weighted avg       0.87      0.87      0.87      4323

Confusion Matrix for KNN:
[[1371   31   35]
 [  96 1251   96]
 [ 137  181 1125]]
```

9) Compare all the models across different evaluation measures and generate a comparison table

Student name: Nguyen Anh Tuan
Student ID: 103805949

22]:

	Model	Accuracy	Precision	Recall	F1 Score
0	DecisionTree	0.970622	0.970612	0.970622	0.970613
1	RandomForest	0.996068	0.996067	0.996068	0.996066
2	GradientBoosting	0.935461	0.935439	0.935461	0.935324
3	SVM	0.461948	0.472434	0.461948	0.394497
4	KNN	0.866759	0.868529	0.866759	0.865382

10) Now select your best-performing model to use as AI. Justify the reason of your selection

Based on the comparison table, the RandomForest model is the best-performing model since it has the highest accuracy, precision, recall, and F1 scores.

11) Now save your selected model

In [24]:
```
import joblib
# Save the trained best model to a file (Kaggle uses `joblib` for model saving)
model_filename = "best_model.pkl"
joblib.dump(best_model, model_filename)
```

Out[24]:
```
['best_model.pkl']
```

C. Step 3: ML to AI

12) Now take the 1000 rows that you have not used (we put aside at the beginning)

:
```
df = pd.read_csv('/kaggle/working/1000_data_points.csv')
# sample_df is saved as '1000_data_points.csv' file in step 1
```

13) Load the model

]:
```
best_model = joblib.load('/kaggle/working/best_model.pkl')
```

14) Iteratively convert columns in each row in the format of your training feature set

Student name: Nguyen Anh Tuan
Student ID: 103805949

```
X_test = sample_df.drop(columns='Class', axis=1)
y_test = sample_df['Class']
```

15) Find class prediction using the loaded model and compare with the original label

+ Code + Markdown

```
y_pred = best_model.predict(X_test)
```

16) Measure the performance of your best model for 1000 unseen data points.

In [31]:

```
print(f'Accuracy: {accuracy}')
print(f'Confusion Matrix: {confusion_matrix}')
print(f'Classification Report: {classification_report}')
```

```
Accuracy: 0.8609440074039797
Confusion Matrix: <function confusion_matrix at 0x7f13aed0cee0>
Classification Report: <function classification_report at 0x7f13aed0d900>
```

17) Now measure the performance of other model using these 1000 data points. Have you observed same result of model selection that you identified through evaluation?

The result did not change that much. The Random Forest model was assessed based on the 1000 unseen data points. It was decided that RandomForest was the best model because the performance metrics matched those shown on the test dataset throughout the evaluation. It further supports the choice of model if I run the same evaluation with other models and see that RandomForest continues to perform better than them.

Student name: Nguyen Anh Tuan
Student ID: 103805949

```
Evaluating DecisionTree...

Performance of DecisionTree on 1000 Unseen Data Points:
Accuracy: 0.967
Confusion Matrix:
 [[320   8   5]
 [  6 320   8]
 [  0   6 327]]
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.96      0.97       333
           1       0.96      0.96      0.96       334
           2       0.96      0.98      0.97       333

    accuracy                           0.97      1000
   macro avg       0.97      0.97      0.97      1000
weighted avg       0.97      0.97      0.97      1000


Evaluating RandomForest...
```

```
Evaluating RandomForest...

Performance of RandomForest on 1000 Unseen Data Points:
Accuracy: 0.991
Confusion Matrix:
 [[330   2   1]
 [  1 329   4]
 [  0   1 332]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.99      0.99       333
           1       0.99      0.99      0.99       334
           2       0.99      1.00      0.99       333

    accuracy                           0.99      1000
   macro avg       0.99      0.99      0.99      1000
weighted avg       0.99      0.99      0.99      1000
```

Student name: Nguyen Anh Tuan
Student ID: 103805949

```
Evaluating GradientBoosting...

Performance of GradientBoosting on 1000 Unseen Data Points:
Accuracy: 0.937
Confusion Matrix:
 [[325   7   1]
 [  7 307  20]
 [  8  20 305]]
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.98      0.97       333
           1       0.92      0.92      0.92       334
           2       0.94      0.92      0.93       333

    accuracy                           0.94      1000
   macro avg       0.94      0.94      0.94      1000
weighted avg       0.94      0.94      0.94      1000
```

```
Evaluating SVM...

Performance of SVM on 1000 Unseen Data Points:
Accuracy: 0.453
Confusion Matrix:
 [[262  14  57]
 [221  22  91]
 [135  29 169]]
Classification Report:
              precision    recall  f1-score   support

           0       0.42      0.79      0.55       333
           1       0.34      0.07      0.11       334
           2       0.53      0.51      0.52       333

    accuracy                           0.45      1000
   macro avg       0.43      0.45      0.39      1000
weighted avg       0.43      0.45      0.39      1000
```

```
Evaluating KNN...

Performance of KNN on 1000 Unseen Data Points:
Accuracy: 0.814
Confusion Matrix:
 [[285  30  18]
 [ 40 267  27]
 [ 39  32 262]]
Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.86      0.82       333
           1       0.81      0.80      0.81       334
           2       0.85      0.79      0.82       333

    accuracy                           0.81      1000
   macro avg       0.82      0.81      0.81      1000
weighted avg       0.82      0.81      0.81      1000
```

D. Step 4: Develop rules from ML model

1. Using only SP feature, generate a decision tree model

In [34]:

```
from sklearn.tree import export_text
sp_features = [col for col in remaining_data_points.columns if col.endswith('SP')]
X_sp = remaining_data_points[sp_features]
y_sp= remaining_data_points['Class']

decisiontree_sp = DecisionTreeClassifier(random_state = 1)
decisiontree_sp.fit(X_sp, y_sp)

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

2. Define some rules

- For Class 0:
  - If TFE Steam pressure SP > 122.50 and TFE Production solids SP > 65.50, then classify as Class 0.
  - If FFTE Steam pressure SP > 104.50 and TFE Production solids SP ≤ 66.50 and FFTE Feed flow SP > 11000.00, then classify as Class 0.
  - If TFE Vacuum pressure SP > -79.95 and FFTE Steam pressure SP > 112.50, then classify as Class 0.

- For Class 1:
  - If TFE Production solids SP ≤ 52.75 and FFTE Steam pressure SP ≤ 94.00, then classify as Class 1.
  - If TFE Production solids SP > 60.00 and TFE Vacuum pressure SP ≤ -76.47, then classify as Class 1.
  - If FFTE Feed flow SP > 9550.00 and TFE Vacuum pressure SP > -66.43 and TFE Production solids SP > 28.25 and FFTE Steam pressure SP > 112.50, then classify as Class 1.
- For Class 2:
  - If TFE Production solids SP > 52.75 and TFE Production solids SP ≤ 60.00 and FFTE Steam pressure SP ≤ 103.50, then classify as Class 2.
  - If FFTE Feed tank level SP ≤ 37.50 and FFTE Steam pressure SP > 111.00 and FFTE Feed flow SP > 9350.00 and TFE Production solids SP > 64.00, then classify as Class 2.
  - If FFTE Out steam temp SP ≤ 50.06 and TFE Production solids SP ≤ 66.00 and FFTE Feed flow SP ≤ 9350.00, then classify as Class 2.

E. Appendix:

- Source code: https://www.kaggle.com/code/twananguyen/port4-submit
- Data: https://drive.google.com/drive/folders/1ikjQfoRPKgVlipk1HzDVEOVj8BZFCTLQ?usp=sharing