

# SWINBURNE UNIVERSITY OF TECHNOLOGY

Start Date: September 16th 2024

Due Date: November 1st 2024

## COS40007: DESIGN PROJECT

### STUDIO 1-4

### GROUP 2 - THEME 3

---

*Submitted By:*

Arnob Gosh

*103494114*

Nguyen Tran

*103844463*

Alexandre Cluzet

*105318766*

Jacob F. Larsen

*105323472*

Nick Watson

*103600076*

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Background and motivation . . . . .	2
1.2	Project objectives . . . . .	2
1.3	Project schedule . . . . .	2
<b>2</b>	<b>Dataset</b>	<b>3</b>
2.1	Data Source . . . . .	3
2.2	Data Processing . . . . .	3
<b>3</b>	<b>Model Development</b>	<b>7</b>
3.1	Decision Tree . . . . .	7
3.2	Random Forest . . . . .	7
3.3	Support Vector Classifier (SVC) . . . . .	8
3.4	Stochastic Gradient Descent (SGD) . . . . .	8
3.5	Multilayer Perceptron (MLP) . . . . .	9
<b>4</b>	<b>User Interface</b>	<b>10</b>
<b>5</b>	<b>Conclusion and Recommendation</b>	<b>11</b>
5.1	Challenges Faced and Solutions Implemented . . . . .	11
5.2	Lessons Learned . . . . .	11
<b>6</b>	<b>Appendix</b>	<b>12</b>
6.1	Source code . . . . .	12

# 1 Introduction

## 1.1 Background and motivation

In the future, several members of our group aspire to become industrial engineers with a focus on optimizing production chains. They are particularly interested in how systems can be streamlined to improve efficiency and reduce waste. The project on Vegemite emerged as a perfect example of such optimization in practice. After some discussion, we unanimously agreed to pursue this project, partly because one group member is an avid fan of Vegemite, which added an extra layer of enthusiasm. This AI model is intended for companies looking to optimize their production chain, detect anomalies. By integrating this model into their systems, company can optimize their production chains but also detect potential disruptions before they escalate, leading to more consistent product quality and reduced downtime. Overall, the project aims to provide practical tools for companies seeking to improve performance and ensure long-term sustainability.

## 1.2 Project objectives

- Determine the recommended values of machine settings during production process for different class of product quality.
- Determine what anomalies can occur in production process for which a production run can fail.

We aim to develop an AI that can determine the optimal machine settings to achieve the desired product quality and analyze anomalies, identifying where errors in the machine settings occur.

For users, the AI model will provide actionable insights on machine settings adjustments in real time and alert operators to potential failures. This would improve efficiency, reduce waste, and increase the reliability of product quality during the manufacturing process.

Through this project, We would like to learn how to develop predictive machine learning model that can recommend optimal operational settings and also detect anomalies.

## 1.3 Project schedule

The Project Schedule will briefly be expected like below. As the project progress, more information and tasks might be added for better result:

Week	Weekly Achievement	Arnob	Alexandre	Jacob	Nick	Nguyen
7	Group Formation	Attend Tutorial, Forming Team and Divide Task				
8	Project Brief	Data	Introduction	Data	Requirements	Introduction
9	Project Training and Development	Data Pre-processing	Training and Validating Data			Data Labeling
10	Project Finalise / Presentation Preparation	AI Demonstration of Final Selected Model				Powerpoint Slide Preparation
11	Presentation	Powerpoint Presentation / Report Preparation				
12	Project Demo / Report Preparation	Demonstration of Final Selected Model / Receiving Feedback and Prepare Final Report				

Figure 1: Project Schedule

## 2 Dataset

### 2.1 Data Source

The data for this project was derived from a combination of machine sensor readings and machine settings during the processing and production of Vegemite. This data had been differentiated into multiple CSV files based on the output consistency quality of the product, labelled as "good", "low bad" and "high bad". Within these files were logs of all available data points, separated by the time of settings configuration. These files were used to provide an insight into which values are correlated with better or worse quality outcomes. The available data columns consisted of batch IDs, raw materials used, set points for controllable components and observed values from sensors, with each series possessing a prefix (TFE, FFTE or Extract Tank) to identify which system of the production line the data has been taken from. Along with the output quality data, this dataset also contained an information log of shutdowns across a two month period in another folder named "Downtime". Once a machine learning model had been developed, this data was intended to be used to predict shutdown events ahead of time based on readings gathered from the system.

### 2.2 Data Processing

The Vegemite dataset required significant cleaning and preprocessing due to factors such as a high volume of duplicate entries, poor class distribution and a number of series which were either irrelevant or required factorisation.

#### 2.2.1 Cleaning

The first step in cleaning was to combine the three CSV files and shuffle them to reduce sequence-based bias. Then the data was checked for any missing values and all 4230 duplicate entries were removed. The columns of 'Set Time', 'VYP Batch' and 'Part' were dropped from the dataset, as they were deemed irrelevant to the output quality. Next, outliers were identified as  $\pm 1.5 \times IQR$  from the upper and lower quartile bounds, with results shown before and after outlier removal.

#### 2.2.2 Preprocessing

The first step of preprocessing was to remove any constant value columns and encode those with 5 or less unique values. For this particular dataset, the four columns of 'FFTE Feed solids SP', 'TFE Steam pressure SP', 'TFE Motor speed' and 'TFE Product out temperature' all contained constant values and were removed. No columns required encoding. Following this, class distribution was checked to ensure balanced representation of each outcome. As is shown in the figure above, Classes 1 and 2 ('Good' and 'High Bad') had a relatively similar distribution, however Class 0 ('Low Bad') was severely underrepresented and thus the dataset required class balancing. Both undersampling and oversampling were employed to achieve the result shown in the figure below.

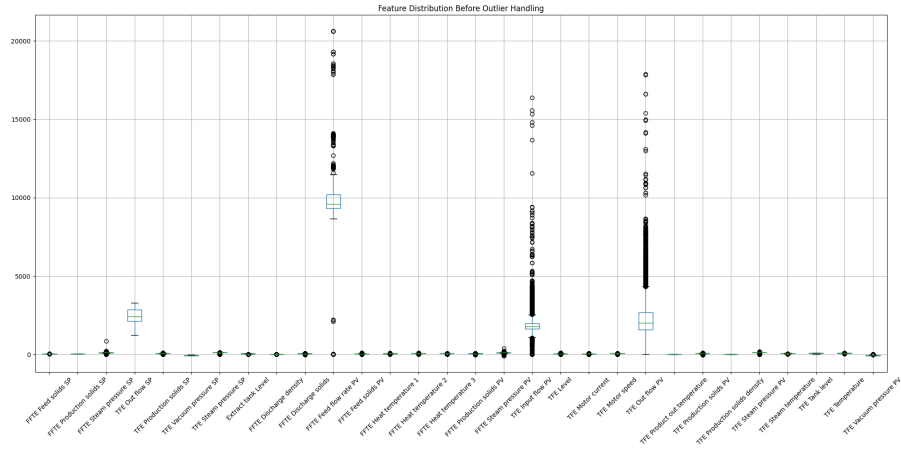


Figure 2: Distribution of values before outlier removal.

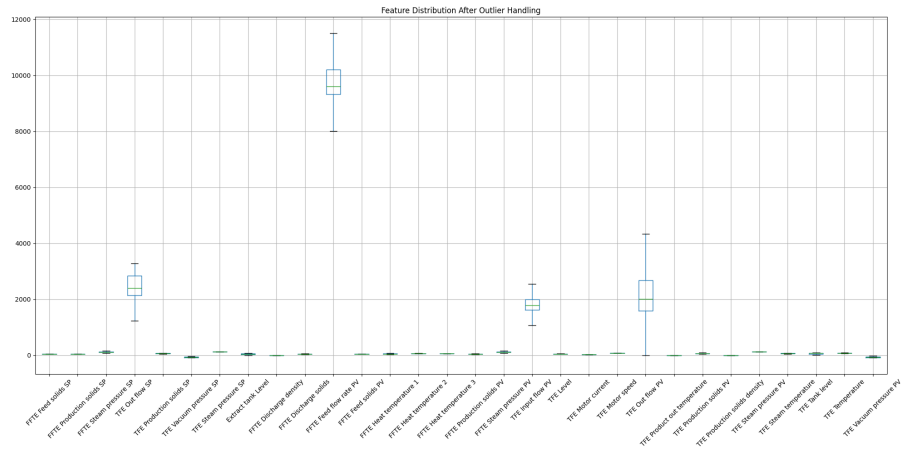


Figure 3: Distribution of values after outlier removal.

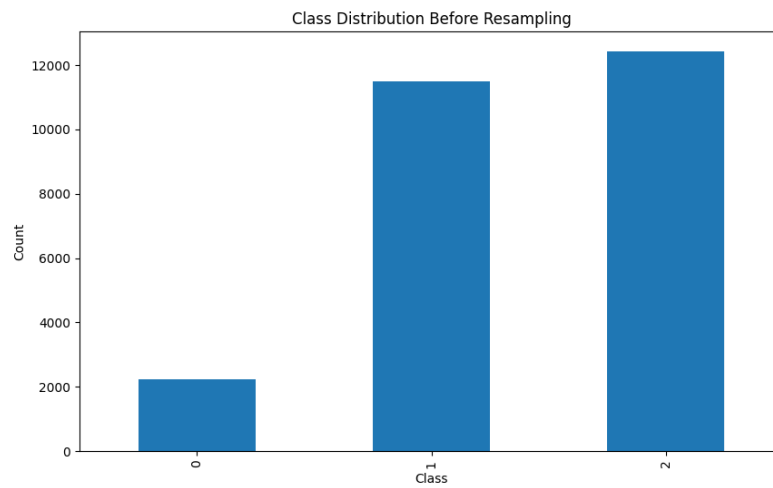


Figure 4: Histogram of class distribution before resampling.

After successfully balanced the class distribution, investigation could begin into the significance of

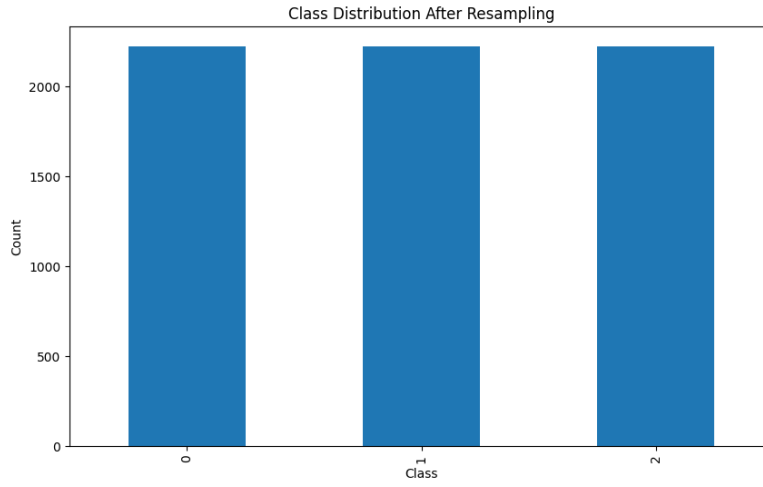


Figure 5: Distribution of classes after resampling techniques.

different series in the dataset. To visualise this, a correlation heat map was created, and features were kept or removed based on their mutual information scores.

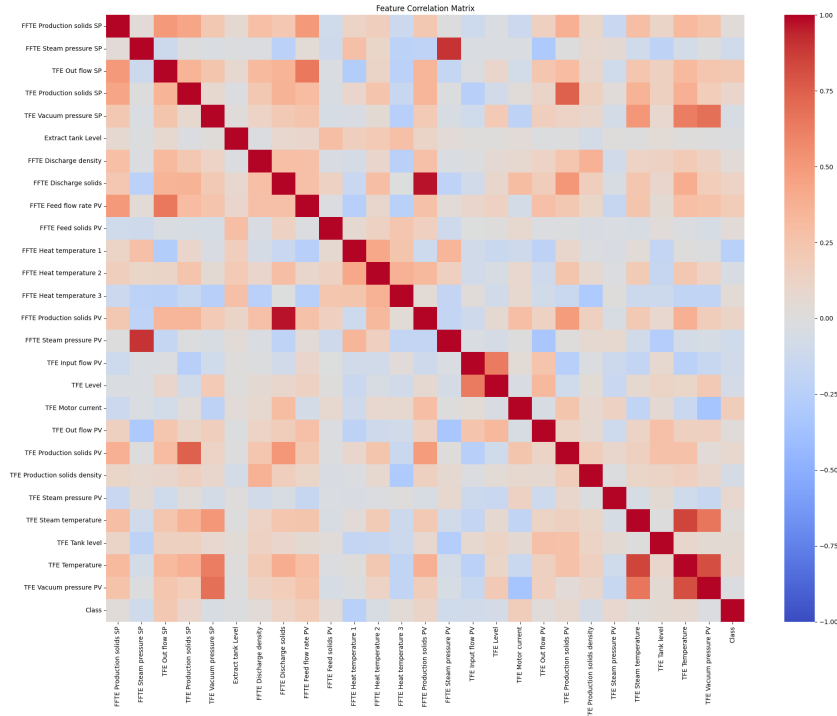


Figure 6: Correlation heat map of all features.

A threshold of  $0.5 \times \mu_{mi}$  was selected as the minimum mutual information score required for the feature to be retained for the machine learning process, where  $\mu_{mi}$  is the mean mutual information score of the dataset. This decision was based on the observation that many of the features had

a very low correlation with the target variable. From this, 17 of the remaining 26 features were retained, while 9 were removed. The relative importance of each feature was calculated by taking the highest mutual information score in the dataset and comparing each other feature's score against this, visualised in the histogram below. As is shown, the lowest relative importance score

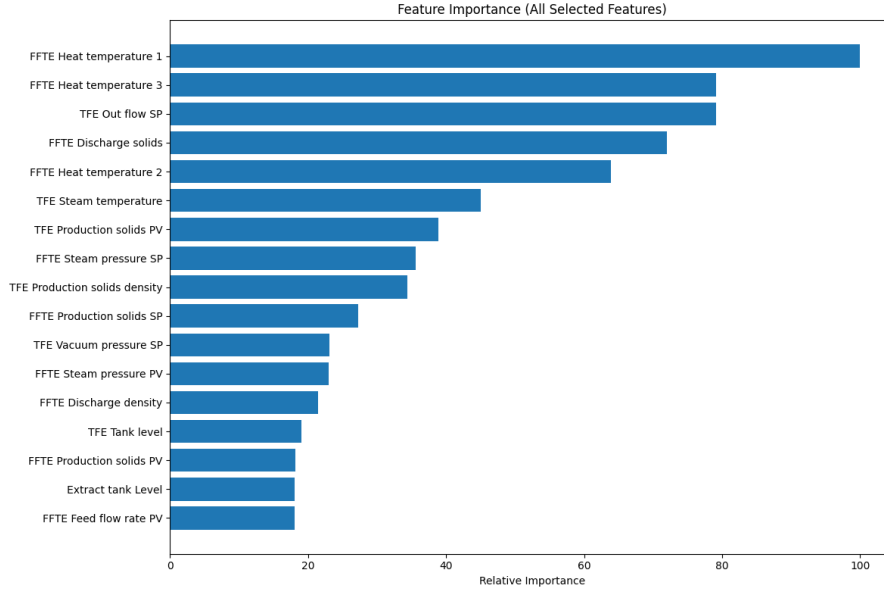


Figure 7: Feature importance ranking based on mutual information scores.

in the retained features was just below 0.2. For this reason, it was deemed that any features with a lower relative importance likely did not have a significant impact on output quality.

### 2.2.3 Avoiding Use of Composite Features and Unsuitability of Downtime Data

For the purpose of recommending machine settings, it was decided that the creation of composite features would convolute the dataset and reduce the ability of the machine learning model to recommend specific values, as it would be relying on pairs of settings rather than treating each as individual components. Therefore, no composite features were integrated into the dataset.

Additionally, exploration undertaken into the provided downtime dataset concluded that this file did not contain any meaningful features which could relate the system downtime back to the specific Vegemite batch that it was caused by/occurred during. Specifically, due to the timestamps given in the output quality files, it is believed that multiple batches were occurring simultaneously and the lack of batch identifiers in the downtime log meant that no correlations could be made with specific machine settings or sensor readings. Additionally, there were no consistencies identified in the temporal separation of downtime events which would allow for predictions to be made based on timing patterns.

### 3 Model Development

In this analysis, we evaluated several machine learning models to determine the best performer for our classification task. The models considered include **Decision Tree**, **Random Forest**, **Support Vector Classifier (SVC)**, **Stochastic Gradient Descent (SGD)**, and **Multilayer Perceptron (MLP)**. Each model was assessed based on its **accuracy**, **confusion matrix**, and **classification report**, which measures precision, recall, and F1-score. Below, we provide an in-depth review of the results obtained.

#### 3.1 Decision Tree

The Decision Tree model achieved an **accuracy of 89.6%**, indicating that it correctly classified most instances. The confusion matrix shows that the model performed well across all classes, with class 0 being the most accurately predicted. Precision and recall scores for class 0 were notably high (0.93 and 0.95, respectively), showcasing the model's strength in classifying this group. However, for classes 1 and 2, there was a slight drop in precision and recall, with class 1 reaching a recall of 0.86, suggesting some misclassification. Overall, the Decision Tree displayed a balanced performance across all metrics, making it a reliable choice for accurate but interpretable predictions.

```
Decision Tree Results:
Accuracy: 0.8958801498127341
Confusion Matrix:
[[436  17   5]
 [ 21 375  39]
 [ 11  46 385]]
Classification Report:

```

	precision	recall	f1-score	support
0	0.93	0.95	0.94	458
1	0.86	0.86	0.86	435
2	0.90	0.87	0.88	442
accuracy			0.90	1335
macro avg	0.90	0.90	0.89	1335
weighted avg	0.90	0.90	0.90	1335

Figure 8: Results Decision tree

#### 3.2 Random Forest

The Random Forest model outperformed other models with an **accuracy of 93.5%**, making it the best performer in this comparison. The confusion matrix revealed a high level of precision across all classes, with class 0 achieving an impressive **0.99 recall** rate. This result suggests that Random Forest is highly effective at minimizing false negatives for this class. Precision and recall values for classes 1 and 2 were also robust, both around **0.90 and 0.92**, respectively, demonstrating the model's consistent ability to generalize across multiple categories. The overall high F1-scores indicate that Random Forest effectively balances precision and recall, reinforcing it as the top



choice for this task.

```

 RandomForest Results:
 Accuracy: 0.9348314606741573
 Confusion Matrix:
 [[454  4  0]
 [ 17 387 31]
 [  4  31 407]]
 Classification Report:

```

	precision	recall	f1-score	support
0	0.96	0.99	0.97	458
1	0.92	0.89	0.90	435
2	0.93	0.92	0.93	442
accuracy			0.93	1335
macro avg	0.93	0.93	0.93	1335
weighted avg	0.93	0.93	0.93	1335

Figure 9: Results RandomForest

### 3.3 Support Vector Classifier (SVC)

The SVC model displayed a moderate performance with an **accuracy of 72.9%**, indicating it struggled more than the tree-based models. The confusion matrix highlights challenges with class 1, where recall was only **0.64**. The lower precision and recall scores across classes reflect the model's difficulty in correctly classifying all samples, especially for class 1, where misclassifications were common. While SVC achieved a reasonable precision of **0.77** for class 0, the overall F1-scores suggest the model may lack the robustness needed for more challenging classification tasks. As such, SVC may require further tuning or feature engineering to improve performance in this context.

```

 SVC Results:
 Accuracy: 0.7288389513108614
 Confusion Matrix:
 [[365  80 13]
 [ 71 280 84]
 [ 40  74 328]]
 Classification Report:

```

	precision	recall	f1-score	support
0	0.77	0.80	0.78	458
1	0.65	0.64	0.64	435
2	0.77	0.74	0.76	442
accuracy			0.73	1335
macro avg	0.73	0.73	0.73	1335
weighted avg	0.73	0.73	0.73	1335

Figure 10: Results SVC

### 3.4 Stochastic Gradient Descent (SGD)

The SGD model was the lowest-performing model, with an **accuracy of 53.2%**. The confusion matrix shows that it had considerable difficulty distinguishing between classes, particularly with class 1, where recall was only **0.23**. This performance is further evidenced by the low F1-score for

this class, indicating that the model frequently misclassified instances from class 1. While class 0 saw a better recall of **0.73**, the model's overall performance was inconsistent, suggesting that SGD may not be well-suited to the dataset without extensive tuning or additional preprocessing.

```
SGD Results:
Accuracy: 0.5318352059925093
Confusion Matrix:
[[336  40  82]
 [207  99 129]
 [110  57 275]]
Classification Report:
```

	precision	recall	f1-score	support
0	0.51	0.73	0.60	458
1	0.51	0.23	0.31	435
2	0.57	0.62	0.59	442
accuracy			0.53	1335
macro avg	0.53	0.53	0.50	1335
weighted avg	0.53	0.53	0.51	1335

Figure 11: Results SGD

### 3.5 Multilayer Perceptron (MLP)

The MLP model achieved a comparable result to the SVC with an **accuracy of 72.9%**. While it showed some improvements over SGD, MLP struggled with similar issues as the SVC, especially with class 1, where precision and recall were both around **0.65**. Class 0, however, saw a slightly better performance, with a recall of **0.84** and precision of **0.79**, reflecting the model's strength in identifying this class. Despite these strengths, the lower scores in other classes suggest that MLP may require further parameter tuning or network adjustments to optimize its classification power.

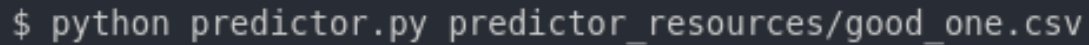
```
MLP Results:
Accuracy: 0.7295880149812735
Confusion Matrix:
[[383  50  25]
 [ 58 281  96]
 [ 41  91 310]]
Classification Report:
```

	precision	recall	f1-score	support
0	0.79	0.84	0.81	458
1	0.67	0.65	0.66	435
2	0.72	0.70	0.71	442
accuracy			0.73	1335
macro avg	0.73	0.73	0.73	1335
weighted avg	0.73	0.73	0.73	1335

Figure 12: Results MLP

## 4 User Interface

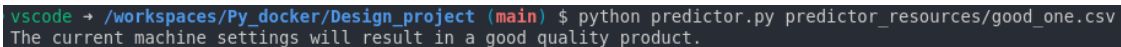
To facilitate the use of the model developed in this project, a simple user interface was created. The interface works like a command line tool, where you specify an input file as you call the program. The input file should be a CSV file containing all the current data points available from the manufacturing plant. See fig. 13 for an example of how to call the program.



```
$ python predictor.py predictor_resources/good_one.csv
```

Figure 13: How to call the prediction program. "predictor.py" is the name of the prediction program, "predictor\_resources/good\_one.csv" specifies the location of the file containing the current data points from the machine.

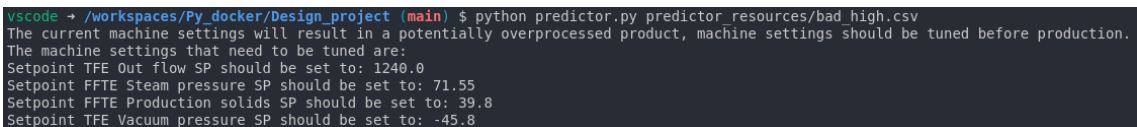
When called the prediction program will perform the necessary data processing and then make a prediction of whether the output product will be of acceptable quality or not. If the prediction indicates that the product will be of good quality, the program will print a success statement and then exit. An example of this can be seen in fig. 14.



```
vscode + /workspaces/Py_docker/Design_project (main) $ python predictor.py predictor_resources/good_one.csv
The current machine settings will result in a good quality product.
```

Figure 14: Example of output text when the prediction indicates a good product.

Conversely if the prediction indicates that the product will be of unacceptable quality, either over-processed or under-processed, the program will try to find a combination of machine set points that will result in a good quality product. It will then print these set points to the terminal, if it could find a combination. For an example of this, see fig. 15.



```
vscode + /workspaces/Py_docker/Design_project (main) $ python predictor.py predictor_resources/bad_high.csv
The current machine settings will result in a potentially overprocessed product, machine settings should be tuned before production.
The machine settings that need to be tuned are:
Setpoint TFE Out flow SP should be set to: 1240.0
Setpoint FFTE Steam pressure SP should be set to: 71.55
Setpoint FFTE Production solids SP should be set to: 39.8
Setpoint TFE Vacuum pressure SP should be set to: -45.8
```

Figure 15: Example of output text and suggested set points after a prediction of bad outcome.

By using this program it is now quick and easy to ensure that the product will be of good quality, and it is easy to know which settings need to be changed for the product to become good.

## 5 Conclusion and Recommendation

Through this project, we developed an AI model that successfully optimizes machine settings and identifies anomalies in the Vegemite production process. After evaluating several machine learning algorithms, the **Random Forest model** emerged as the most accurate, achieving a classification accuracy of **93.5%**. This model performed consistently well across all product quality categories, offering actionable insights into machine settings adjustments and anomaly detection. The solution not only supports improved production efficiency, but also improves product quality reliability by minimizing machine-related failures.

### 5.1 Challenges Faced and Solutions Implemented

We encountered several challenges during the project:

1. **Data Imbalance:** The dataset initially showed a significant class imbalance, with certain quality levels. To address this, we applied a combination of over-sampling and under-sampling techniques, achieving a balanced class distribution that improved the performance and reliability of the model.
2. **Data Quality Issues:** The dataset contained many duplicate and irrelevant entries, along with outliers that could negatively impact the accuracy of the model. We handled this by removing duplicates, filtering unnecessary columns, and using the IQR method to eliminate outliers. In addition, a thorough feature selection process allowed us to retain only the most relevant features, further enhancing the accuracy of the model.
3. **Model Tuning:** While some models, such as the Support Vector Classifier and Stochastic Gradient Descent, initially performed poorly, further tuning helped improve their results to some extent. However, after a detailed analysis, we determined that the Random Forest model provided the best balance between precision, recall, and F1 scores.

### 5.2 Lessons Learned

This project provided our team with valuable information on the application of machine learning in a real-world manufacturing context. We learned how to handle complex datasets, particularly those with data quality issues and class imbalances, to produce accurate predictions. In addition, we gained a deeper understanding of feature selection, model tuning, and evaluation, recognizing the importance of each step in achieving reliable results. The experience also underscored the critical role of machine learning in optimizing industrial processes, which aligns with our long-term professional goals in production optimization and industrial engineering.

## 6 Appendix

### 6.1 Source code

The source code for the project can be found here:

<https://github.com/lesoldatalex/COS40007---Design-Project>