Table of Contents

List of Figures

List of Tables

AER850 Project 1

Introduction

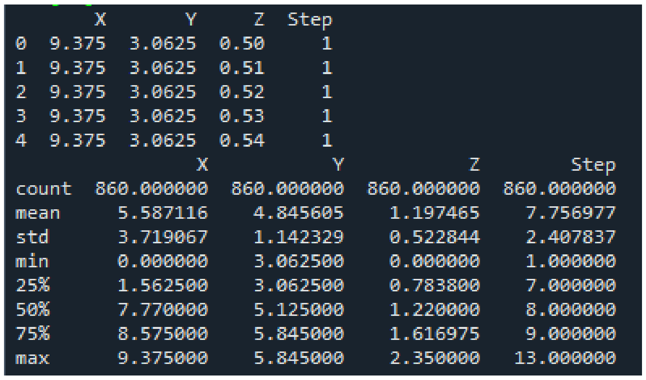
The objective of this project is to develop a classification machine learning model to predict maintenance steps during the disassembly of the FlightMax Fill Motion Simulator Inverter. The coordinate data for the inverter is provided, which displays coordinate data (x, y, z) alongside its 13 maintenance steps. The project workflow will include data preprocessing, data visualization, correlation analysis, classification model development/engineering, model performance analysis, stacked model performance analysis and model evaluation. The development of this model will be achieved through designing, training, and evaluating multiple supervised machine learning models, including Logistic Regression, Support Vector Machine (SVM), and Random Forest. These models will be tuned with the help of tools like GridSearchCV and RandomizedSearchCV to identify the most optimal configurations. Furthermore, model performance will be tested using accuracy, precision and F1 score metrics in order to select the best model for this application.

Part 1: Data processing

In part 1, the CSV file containing the data was imported into Spyder and put into a dataframe for analysis. The features x, y, and z were selected as the input variables, and step was defined as the output. The data was then split into a conventional 80-20 split, with 80% as the training data, while the other 20% was the test data. Furthermore, stratified sampling was used while splitting the data to ensure the same class distribution as the original dataset.

The data had four columns (x, y, z, step). Where the x values ranged from 0 to 9.375, the y values ranged from 3.0625 to 5.845, the z values ranged from 0 to 2.35 and the step values from from 1 to 13.

*Note: The data was split before visualization to avoid data snooping bias.



```
       X       Y     Z  Step
0  9.375  3.0625  0.50     1
1  9.375  3.0625  0.51     1
2  9.375  3.0625  0.52     1
3  9.375  3.0625  0.53     1
4  9.375  3.0625  0.54     1
                X           Y           Z        Step
count  860.000000  860.000000  860.000000  860.000000
mean     5.587116    4.845605    1.197465    7.756977
std      3.719067    1.142329    0.522844    2.407837
min      0.000000    3.062500    0.000000    1.000000
25%      1.562500    3.062500    0.783800    7.000000
50%      7.770000    5.125000    1.220000    8.000000
75%      8.575000    5.845000    1.616975    9.000000
max      9.375000    5.845000    2.350000   13.000000
```

Figure 1: Dataframe Info

Part 2: Data Visualization

In part 2, the data was plotted to better visualize the underlying patterns and gain an overall deeper understanding of the data. The 2 plots used for this were a 3D scatter plot and a Scatter matrix. The Scatter matrix contains all the pairwise relationships between the features, which allows for multiple plots using only one function (pd.plotting.scatter_matrix()).

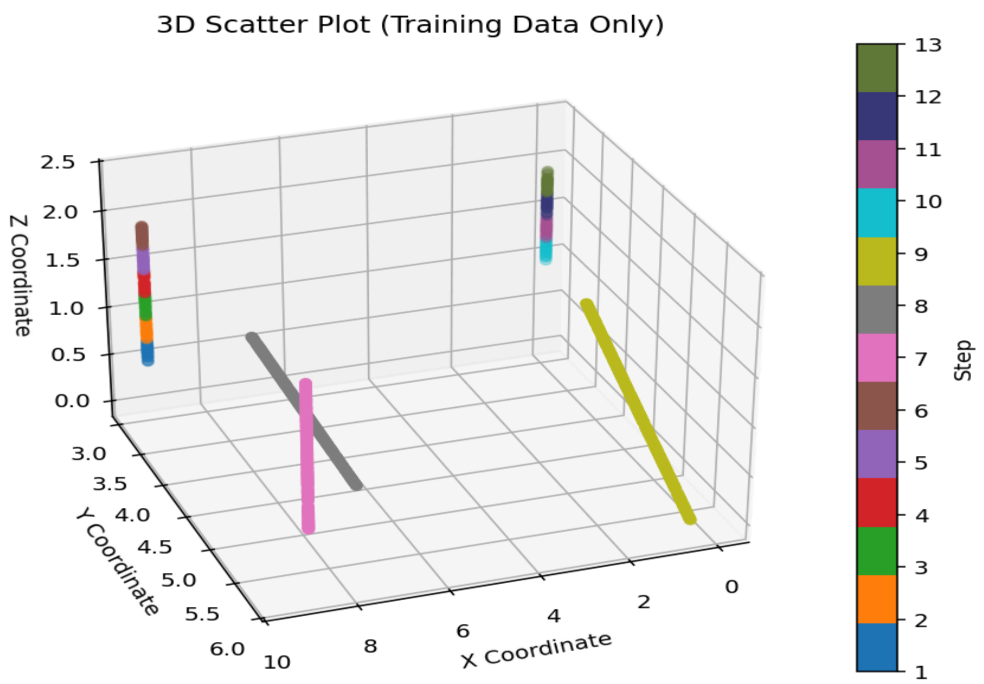


Figure 2: 3D Scatter Plot

Figure 2 shows the 3D Scatter plot. This plot gives an intuitive representation of how each maintenance step is distributed along the coordinates (x,y,z). The colours on the plot represent the maintenance step, where 1 is blue and dark green is 13. From the plot, it can be seen that the maintenance steps vary mostly in the z direction with minimal change in the x and y. This implies that the z values may play a dominant role in step differentiation. Furthermore, it can be seen that the steps are clustered and spatially distinct. This usually means the data is suitable for classification machine learning models. Also, the column-like formation of the clusters may correlate to tool/component movement along a specific axis, which may suggest that there is some structured workflow for maintenance. Lastly, there seems to be a pattern where lower z values have early maintenance steps, which indicates that the maintenance steps start off closer to the base of the component.

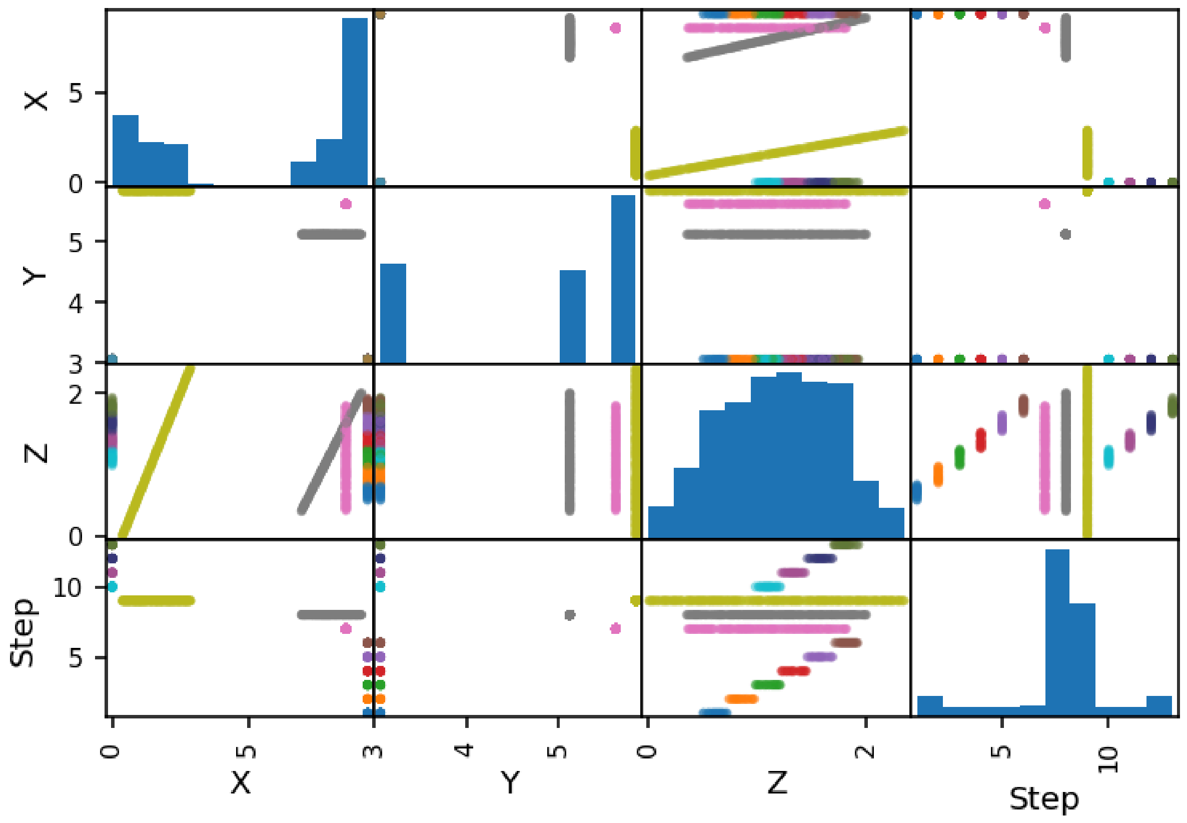


Figure 3: Scatter plots for all data

Figure 3 exhibits a scatter plot matrix showing all the relationships between all the features and their histograms along its diagonal. Each subplot is a 2D projection of two features showing a detailed view of how the features (coordinates) and target (step) are distributed in relation to each other. The histograms along the diagonal show that the z values are distributed more evenly throughout their range, whereas the x and y values are only present at the low and high ends. The step histogram also shows that most of the data points fall under the 7-9 range, whereas the other steps are scarce. In the z-step plot, it can be seen that all maintenance steps are present for each z-value; this is especially true for step 9, which is present at all z-values. This is indicated with a horizontal line across the entire plot. Furthermore, analyzing the x-step and y-step plots, the findings from the 3D scatter plot in Figure 2 can be confirmed. These 2 plots reaffirm that x and y have less variation with shorter flat segments where whereas the z-step plot has large variation.

Part 3: Correlation Analysis

Part 3 requires a correction matrix to be constructed. This was done using seaborn for the heatmap and pandas for the dataframe.

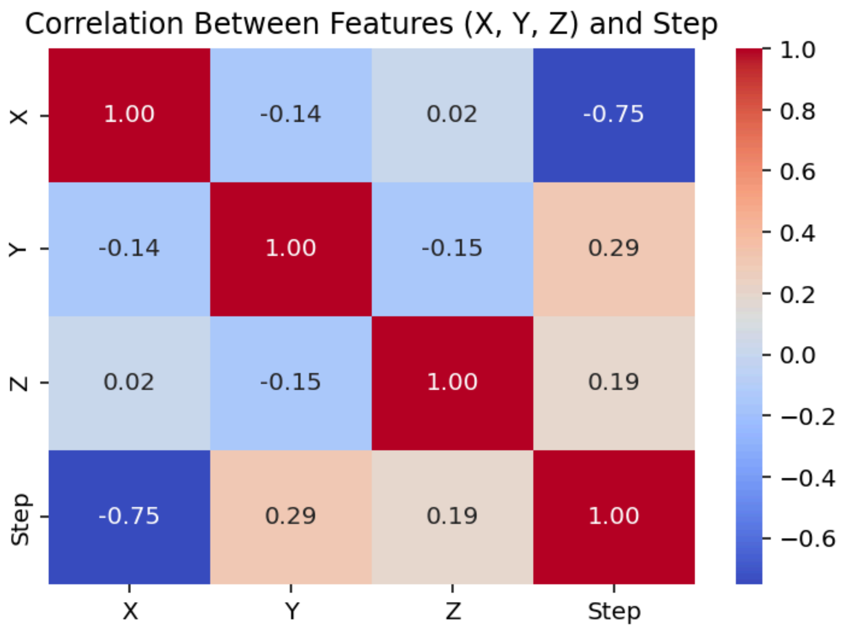


Figure 4: Correlation Matrix

The correlation matrix in Figure 4 displays the correlation of the features and target variables in a visual display of a matrix. The diagonal values are 1s, which shows perfect correlation of a feature/target variable with itself. This is not useful or relevant information, as the feature/target variable will obviously have 100% correlation with itself. The other boxes reveal useful correlation metrics. The x value shows a strong negative correlation with Step (-0.75). This means that as x increases, the step value decreases, which shows that the maintenance step sequence progresses in the opposite direction to the x-axis. Unlike the x value, the z (0.19) and y (0.29) values show weak correlation with step, showing that these coordinates are less variant across maintenance steps. This finding disagrees with what was said in step 2. Step 2 showed that the z coordinates varied fairly significantly across maintenance steps. Now the correlation matrix shows the opposite effect. This may be because the maintenance step process has non-linear movements, which means the correlation matrix will not pick up on it only measures linear relationships. From this, it can be concluded that both X and Z may be good predictors of the maintenance steps which can help in ML model performance.

Part 4: Classification Model Development/Engineering

In part 4, multiple machine learning models were developed and optimized to predict the maintenance steps of the inverter based on the coordinate data (x,y,z). The four models selected were Logistic Regression, Support Vector Machine (SVM), Random Forest, and Decision Tree. For all the models, grid search cross-validation was used through Scikit-learn GridSearchCV and RandomizedSearchCV. This was done to identify the best hyperparameters to optimize the F1 score of the model. For cross-validation, the data was split into 5 folds, which include 4 training folds and 1 testing fold. For each iteration, the test and train folds would change, which would allow for training and testing with all the data. Models such as Logistic Regression and SVM are sensitive to feature magnitudes. To combat this, a pipeline was implemented that scaled the data using StandardScaler, which ensured that all the input features were comparable in magnitude. Scaling in this case improves numerical stability and fairness. The other models did not require scaling as they split data based on thresholds rather than numerical distances.

Table 1: Model Hyperparameter Optimization

Logistic Regression		
Parameter	**Values Used**	**Best Value**
C	0.01, 0.1, 1, 10, 100	0.01
Solver	"lbfgs", "newton-cg", "saga"	"newton-cg"
Penalty	None, "l2"	None
Support Vector Machine (SVM)		
C	0.1, 1, 10, 100	1
Kernel	"rbf", "sigmoid", "poly"	poly
Gamma	'scale', 'auto'	scale
Random Forest		
n_estimators	50, 100, 200	50
max_depth	None, 10, 20, 30	None
min_samples_split	2, 5, 10	2
min_samples_leaf	2, 4, 6	2
max_features	'sqrt', 'log2'	‘sqrt’

criterion	'gini', 'entropy'	'entropy'
Decision Tree		
max_depth	None, 10, 20, 30	None
min_samples_split	2, 5, 10	2
min_samples_leaf	2, 4, 6	2
max_features	'sqrt', 'log2'	‘sqrt’
criterion	'gini', 'entropy'	'entropy'

Each of these 4 models was strategically chosen for their unique learning behaviours:

- Logistic regression was chosen as a baseline linear classifier, as it is capable of separating maintenance steps that follow near-linear spatial boundaries. Also, it can be extended to multiple classes and can work well with small data sets. Furthermore, it is less likely to overfit data compared to other models. Another advantage is that logistic regression is easy to interpret, as the model shows how strongly each coordinate affects the predictions. This makes it easier to compare with other models.

- SVM was chosen to handle non-linear data relationships, which was shown to be the case in step 3. SVM can handle non-linearity through the kernel (poly or RBF). It is not much affected by outliers and is well-suited for medium-sized datasets.

- Random Forest was also chosen to handle complex nonlinear data relationships. This model has high accuracy and robustness on noisy datasets. It combines many decision trees, where each is trained on random subsets of data and features. This allows it to handle a large variety of spatial relationships without overfitting.

- Decision Tree was chosen for nonlinearity, simplicity and interpretability. It makes decisions through splitting the data into smaller groups based on threshold values of coordinate features. This type of prediction closely resembles human decision-making, which makes it easy to understand how the model works.

Part 5: Model Performance Analysis

The models were evaluated using four metrics, these being accuracy, precision, recall and F1-score. Alongside this, a confusion matrix was also created for each model to compare overall classification performance and understand where correct and incorrect predictions were made.

$$Accuracy = \frac{\#\ of\ correct\ predictions}{\#\ of\ all\ data\ points}$$

$$Precision = \frac{\#\ of\ true\ positives}{\#\ of\ true\ positives\ +\ \#\ of\ false\ positives}$$

$$Recall = \frac{\#\ of\ true\ positives}{\#\ of\ true\ positives\ +\#\ of\ false\ negatives}$$

$$F_1 = 2 \times \frac{precision\ \times recall}{precision + recall}$$

- Accuracy measures the percentage of correct predications out of all the predications made. It serves to give a general measure of how correct the model is. However, in this case, it is not a good metric as the dataset is imbalanced. From the accuracy equation above, it can be seen that if the model only gets true negatives correctly, then it would still have a high accuracy even though it's getting the true positives wrong. This is why this metric can be misleading.

- Precision is a measure of how many of the model's predicted positives were actually true positives. It gives a measure of the model's ability to avoid false alarms. Precision is often more useful than accuracy when dealing with imbalanced datasets, because it focuses on the quality of positive predictions rather than overall correctness. However, precision used alone can be misleading if the model fails to identify many actual positives. A high precision value indicates that the model makes few false positive errors.

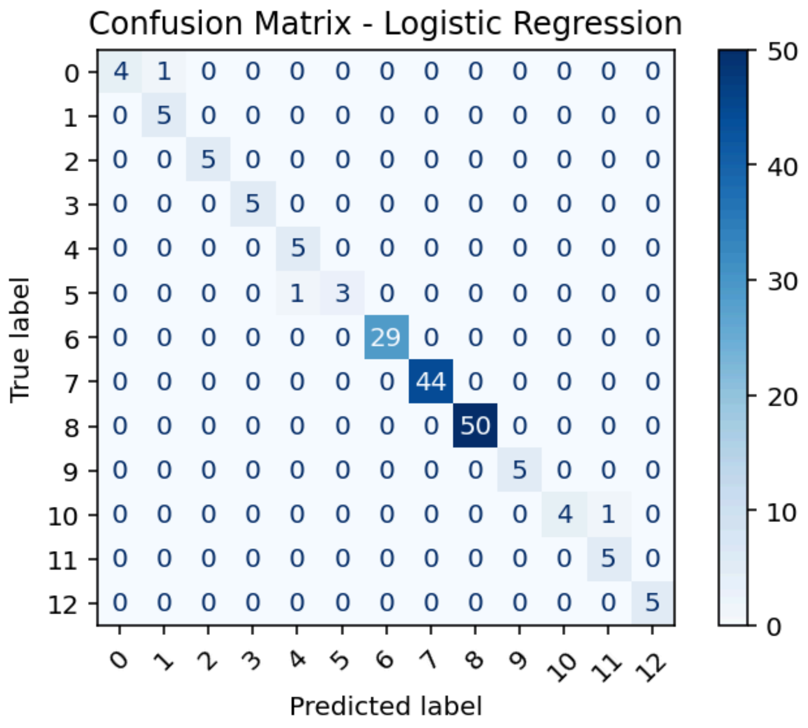
- Recall measures how many of the actual positives were correctly identified as positive. It's the model's ability to capture all relevant cases. However, recall alone does not account for false positives, so it’s often used together with precision to provide a more balanced assessment. A high recall value indicates that the model rarely misses true positives, meaning there will be fewer false negatives.

- The F1 score is the combination of precision and recall. The F1 score has a balance between avoiding false positives (precision) and lowering false negatives (recall). Because of this F1 score is useful when working with imbalanced datasets. In this project, class frequencies are not balanced, which makes the F1 score the most reliable and useful metric for this case, as it captures the trade-off between precision and recall in a single metric.

Table 2: Model Performance Metrics

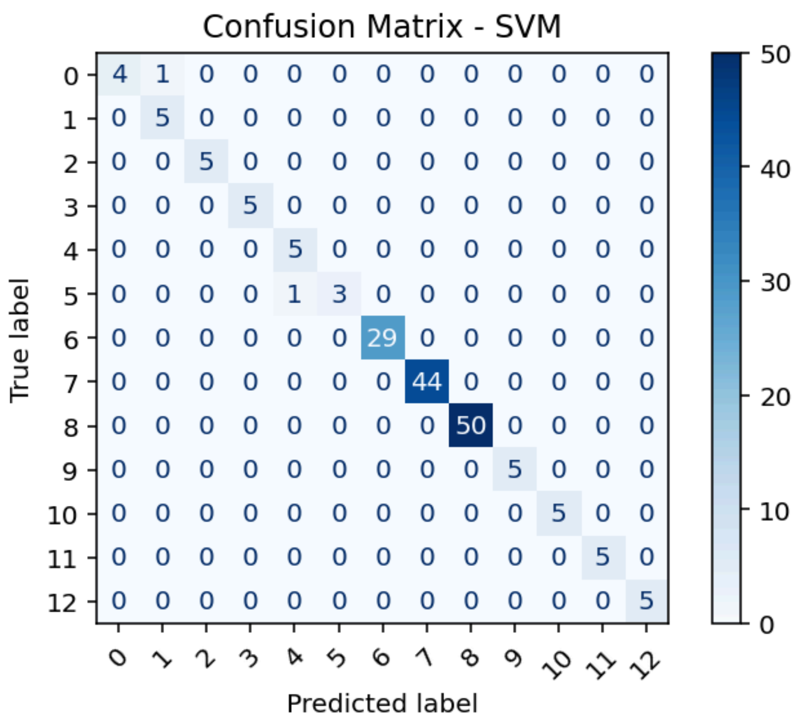
Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.9826	0.9855	0.9826	0.9823
SVM	0.9884	0.9903	0.9884	0.9882
Random Forest	0.9842	0.9852	0.9842	0.9840
Decision Tree (RandomizedSearchCV)	0.9826	0.9840	0.9826	0.9811

From Table 2, it can be seen that all the models showed high accuracy, precision and F1 score, with all being above 0.98. As all the models performed exceptionally well, it can be concluded that there is likely some overfitting occurring, which is why the performance metrics are so high. This was attempted to be fixed through adjusting the hyperparameters; however, since the dataset is very small, there was not much of an effect. Furthermore, the models used on this dataset were very powerful, and since the dataset is very small, naturally, the model would overfit the data. Overall, it can be seen that SVM has the highest F1 score of 0.9882; however, it doesn't mean much since all the models have very similar scores.



True \ Predicted	0	1	2	3	4	5	6	7	8	9	10	11	12
0	4	1	0	0	0	0	0	0	0	0	0	0	0
1	0	5	0	0	0	0	0	0	0	0	0	0	0
2	0	0	5	0	0	0	0	0	0	0	0	0	0
3	0	0	0	5	0	0	0	0	0	0	0	0	0
4	0	0	0	0	5	0	0	0	0	0	0	0	0
5	0	0	0	0	1	3	0	0	0	0	0	0	0
6	0	0	0	0	0	0	29	0	0	0	0	0	0
7	0	0	0	0	0	0	0	44	0	0	0	0	0
8	0	0	0	0	0	0	0	0	50	0	0	0	0
9	0	0	0	0	0	0	0	0	0	5	0	0	0
10	0	0	0	0	0	0	0	0	0	0	4	1	0
11	0	0	0	0	0	0	0	0	0	0	0	5	0
12	0	0	0	0	0	0	0	0	0	0	0	0	5

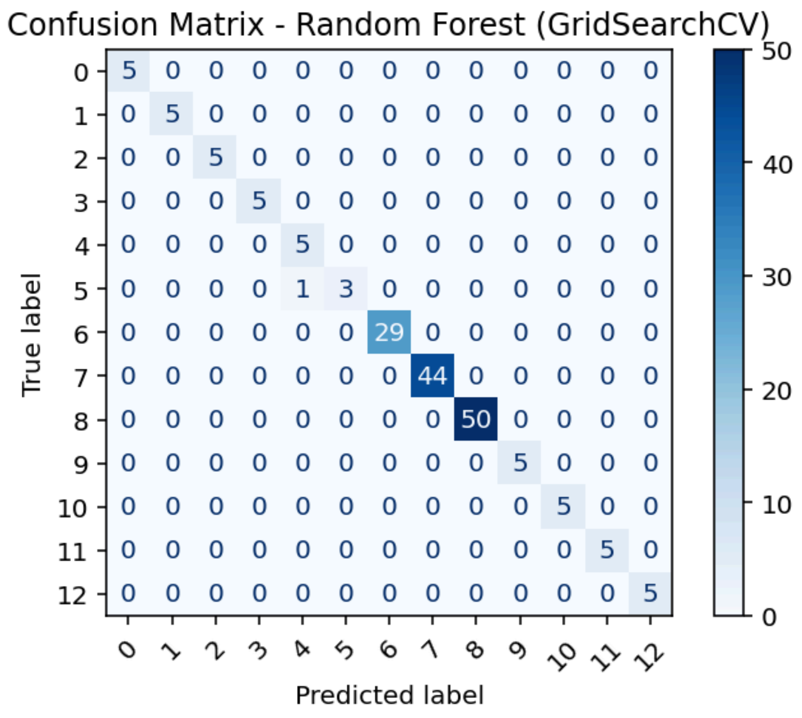
Figure 5: Logistic Regression Confusion Matrix



True \ Predicted	0	1	2	3	4	5	6	7	8	9	10	11	12
0	4	1	0	0	0	0	0	0	0	0	0	0	0
1	0	5	0	0	0	0	0	0	0	0	0	0	0
2	0	0	5	0	0	0	0	0	0	0	0	0	0
3	0	0	0	5	0	0	0	0	0	0	0	0	0
4	0	0	0	0	5	0	0	0	0	0	0	0	0
5	0	0	0	0	1	3	0	0	0	0	0	0	0
6	0	0	0	0	0	0	29	0	0	0	0	0	0
7	0	0	0	0	0	0	0	44	0	0	0	0	0
8	0	0	0	0	0	0	0	0	50	0	0	0	0
9	0	0	0	0	0	0	0	0	0	5	0	0	0
10	0	0	0	0	0	0	0	0	0	0	5	0	0
11	0	0	0	0	0	0	0	0	0	0	0	5	0
12	0	0	0	0	0	0	0	0	0	0	0	0	5

,

Figure 6: SVM Confusion Matrix



True \ Predicted	0	1	2	3	4	5	6	7	8	9	10	11	12
0	5	0	0	0	0	0	0	0	0	0	0	0	0
1	0	5	0	0	0	0	0	0	0	0	0	0	0
2	0	0	5	0	0	0	0	0	0	0	0	0	0
3	0	0	0	5	0	0	0	0	0	0	0	0	0
4	0	0	0	0	5	0	0	0	0	0	0	0	0
5	0	0	0	0	1	3	0	0	0	0	0	0	0
6	0	0	0	0	0	0	29	0	0	0	0	0	0
7	0	0	0	0	0	0	0	44	0	0	0	0	0
8	0	0	0	0	0	0	0	0	50	0	0	0	0
9	0	0	0	0	0	0	0	0	0	5	0	0	0
10	0	0	0	0	0	0	0	0	0	0	5	0	0
11	0	0	0	0	0	0	0	0	0	0	0	5	0
12	0	0	0	0	0	0	0	0	0	0	0	0	5

Figure 7: Random Forest Confusion Matrix

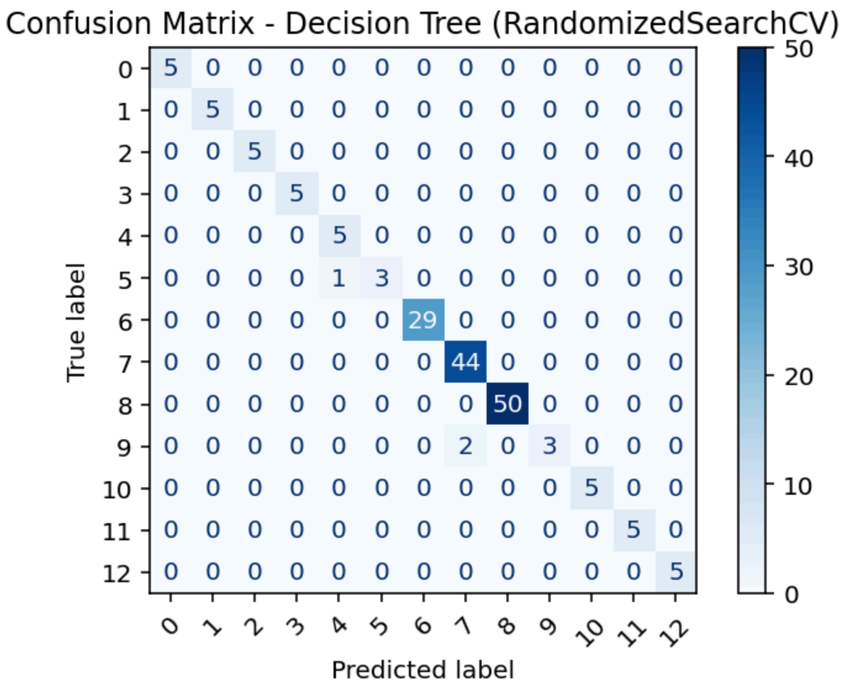


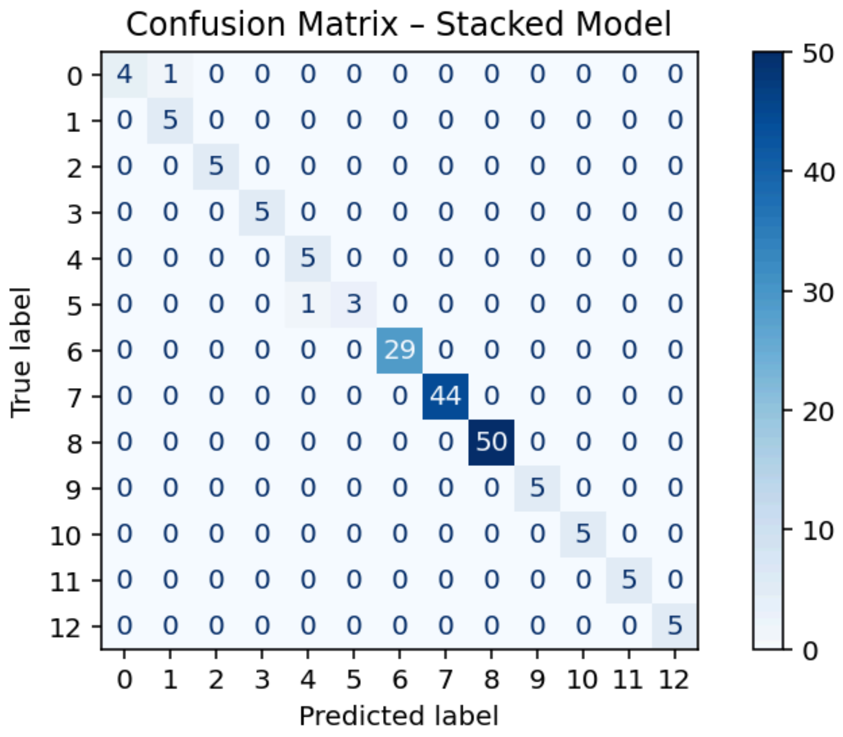
Figure 8: Decision Tree Confusion Matrix (RandomizedSearchCV)

In all the confusion matrices from Figures 5-8, it can be seen that almost all the samples are on the diagonal. This means that the true positives dominate the predicted labels, and false positives and false negatives are almost non-existent. The elements not on the diagonal are essentially all zero, except for 1 or 2 cases. This means that there is very little misclassification and high performance. This further reaffirms that there may be overfitting occurring in the data, as the performance for all the models is almost perfect.

Part 6: Stacked Model Performance Analysis

Table 3: Stacked Model Performance Metrics

Model	Accuracy	Precision	Recall	F1-Score
SVM + Random Forest	0.9884	0.9903	0.9884	0.9882



True \ Predicted	0	1	2	3	4	5	6	7	8	9	10	11	12
0	4	1	0	0	0	0	0	0	0	0	0	0	0
1	0	5	0	0	0	0	0	0	0	0	0	0	0
2	0	0	5	0	0	0	0	0	0	0	0	0	0
3	0	0	0	5	0	0	0	0	0	0	0	0	0
4	0	0	0	0	5	0	0	0	0	0	0	0	0
5	0	0	0	0	1	3	0	0	0	0	0	0	0
6	0	0	0	0	0	0	29	0	0	0	0	0	0
7	0	0	0	0	0	0	0	44	0	0	0	0	0
8	0	0	0	0	0	0	0	0	50	0	0	0	0
9	0	0	0	0	0	0	0	0	0	5	0	0	0
10	0	0	0	0	0	0	0	0	0	0	5	0	0
11	0	0	0	0	0	0	0	0	0	0	0	5	0
12	0	0	0	0	0	0	0	0	0	0	0	0	5

Figure 9: Stacked Model Confusion Matrix (Random Forest and SVM)

In order to try and improve model performance, StackingClassifier was used to stack SVM and Random Forest through a technique that combines the predictions of both models into one. These 2 models were chosen to be stacked together because they displayed the highest F1 score. From the results of Table 3, it can be seen that the F1 score of the stacked model is 0.9882, which is higher than SVM and Random forest, but only by a very small margin, making it a very minimal increase. This is what was expected, as both SVM and Random Forest already had exceptionally high scores individually, it makes sense that when combining them, the increase would be very small. Also, when looking at the confusion matrix in Figure 5, it is identical to the confusion matrices of the other individual models, further proving that the minimal increase in F1 score is negligible.

Part 7: Model Evaluation

Table 4: Stacked Model Predictions

X	Y	Z	Maintenance Step Prediction
9.375	3.0625	1.51	**5**
6.995	5.125	0.3875	**8**
0	3.0625	1.93	**13**
9.4	3	1.8	**6**
9.4	3	1.3	**4**

In part 7, the practical performance of the final stacked model was assessed. In Table 4, it can be seen that 5 different coordinates were given to the model for it to try and predict the maintenance step of each. It was found that the maintenance steps were 5,8,13,6, and 4, respectively. Furthermore, the stacked model was saved in a joblib format file, allowing it to be called upon as a function for use.

Conclusion

In conclusion, the purpose of this project was to create a machine learning model to predict and classify maintenance steps of a FlightMax Fill Motion Simulator Invester using coordinate data. This was accomplished through the development of four models, which included Logistic Regression, SVM, Random Forest, and Decision Tree. It was found that all the models displayed very similar performance metrics, meaning they perform the same. This was attributed to the fact that the data was likely overfit. When stacking SVM and Random Forest models, the performance only increased by a very marginal amount. The stacking model was then used to predict the maintenance steps for 5 coordinate points, which gave the results of 5,8,13,6, and 4, respectively, as seen in Table 4. Overall, the model was effective; however, in the future, a larger dataset would likely help in fixing any overfitting.

References

[1] R. Faieghi, (2025). AER850 – Intro to Machine Learning (L5 Classification), lecture, Toronto Metropolitan University.

[2] Scikit-Learn, "Classification," scikit-learn, 2025. https://scikit-learn.org/stable/supervised_learning.html