




**Department of  
Aerospace Engineering**  
Faculty of Engineering  
& Architectural Science

|                              |                                  |
|------------------------------|----------------------------------|
| <b>Semester (Term, Year)</b> | F2025                            |
| <b>Course Code</b>           | AER850                           |
| <b>Course Section</b>        | 02                               |
| <b>Course Title</b>          | Introduction to Machine Learning |
| <b>Course Instructor</b>     | Dr. Reza Faieghi                 |
| <b>Submission</b>            | 1                                |
| <b>Submission No.</b>        | 1                                |
| <b>Submission Due Date</b>   | October 6, 2025                  |
| <b>Title</b>                 | Project 1 - Kai Stewart          |
| <b>Submission Date</b>       | October 6, 2025                  |

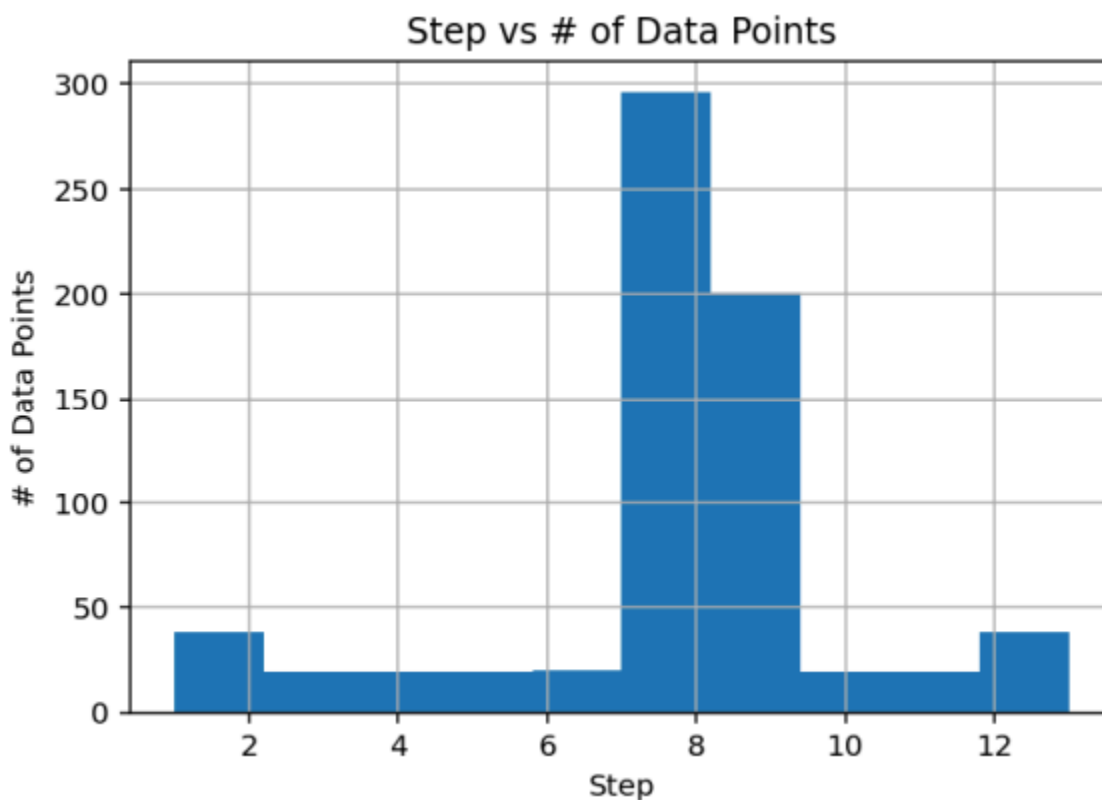
|                              |                                  |   |
|------------------------------|----------------------------------|---|
| <b>Submission by (Name):</b> | <b>Student ID<br/>(XXXX1234)</b> | <b>Signature</b>  |
| Kai Stewart                  | 29849                            |  |

*By signing the above you attest that you have contributed to this submission and confirm that all work you contributed to this submission is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a "0" on the work, and "F" in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Academic Integrity Policy 60, which can be found at [www.torontomu.ca/senate/policies/](http://www.torontomu.ca/senate/policies/)*

*Aerospace Assignment Cover as of May 2022*

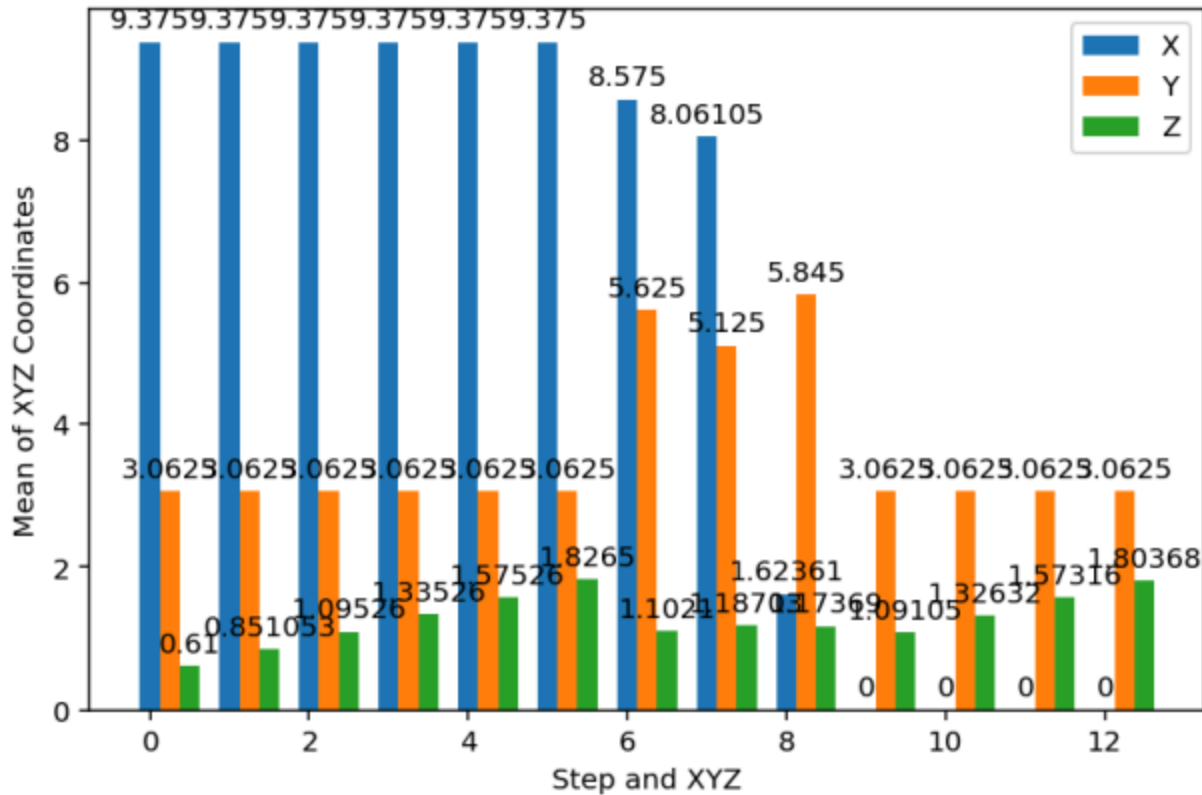
## Step 2: Data Visualization

Figure 2.1 below shows a quick high-level overview of the data, and we can see here that our classes are not balanced; there are a lot more samples in steps 7, 8 and 9 than there are in any of the other classes. This is important to know for using metrics later on in the project, as some may be negatively biased by the introduction of a massive imbalance such as this.



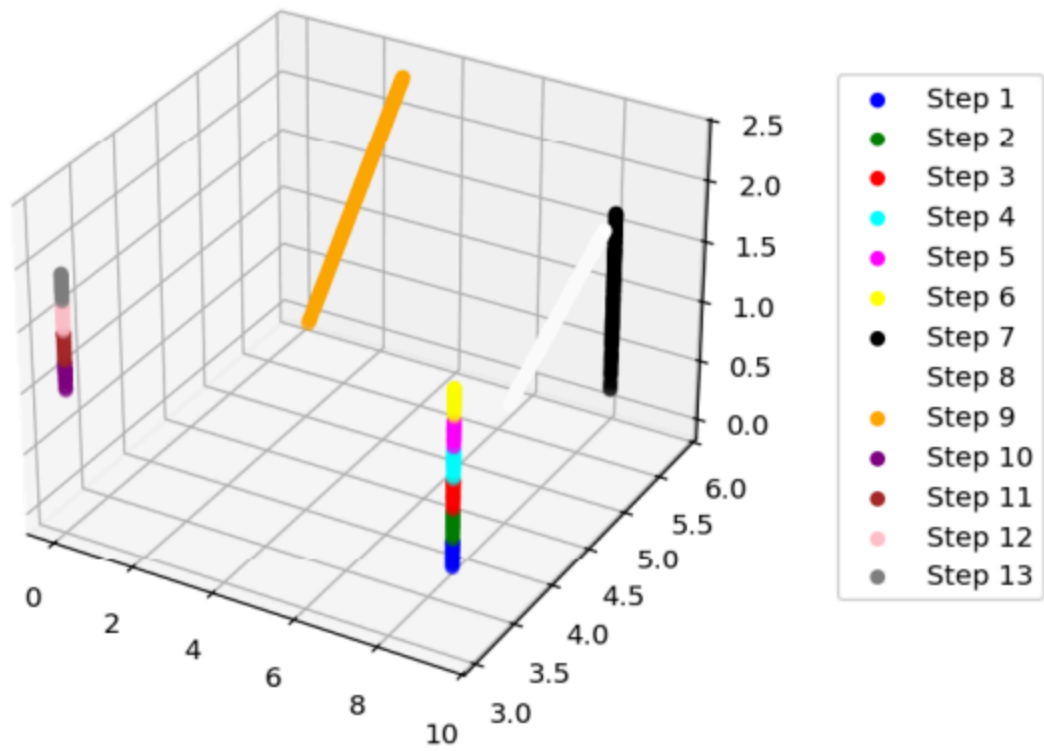
**Figure 2.1:** Histogram of Step vs # of Data Points per Step

Figure 2.2 below takes it a step lower by calculating the mean X, Y and Z coordinate for each step and plotting it on a multi-bar plot, such that we can see all of their relation to one another. Here, we can see some very interesting things that are good to know about how the data is laid out. We can see that for steps 0 to 5, the X and Y coordinates do not change, and it is only the Z coordinate that changes. Same thing for steps 10 to 13. I am sure this is not a very common occurrence in machine learning, but I would think that if I can notice the patterns immediately from this graph, then ML algorithms will have no problem being able to classify these steps.



**Figure 2.2:** Mean of X, Y and Z Coordinates for Each Step

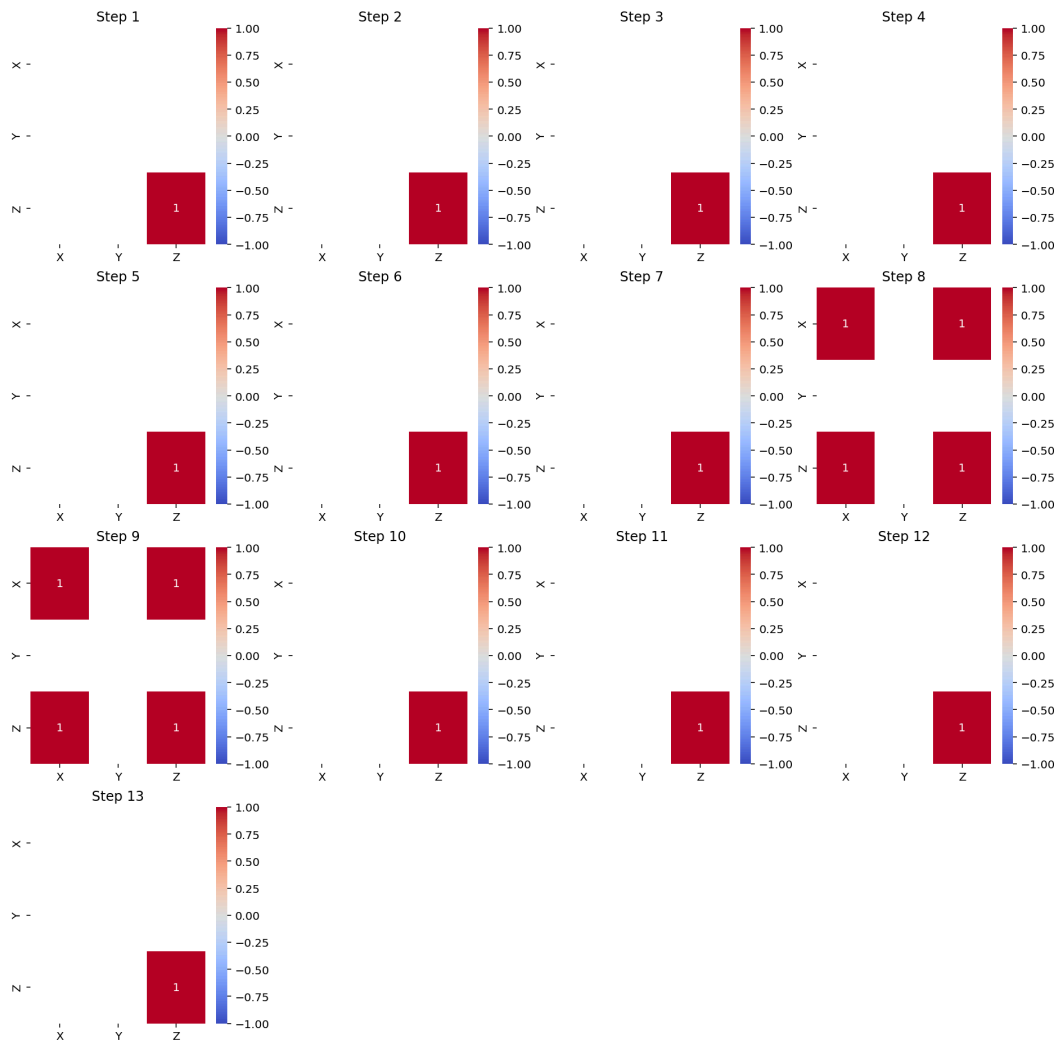
Figure 2.3 below takes it a next step further, by allowing us to be able to visualize these maintenance steps in a 3D space. In Figure 2.3, we can see the same behaviour that we discovered in Figure 2.2, except it is much more pronounced due to the 3D space. I can actually imagine the actions being made in real life. While this may have introduced data snooping bias, I am sure that there are no particularly worrisome artifacts that we will have to account for throughout our model creation process.



**Figure 2.3:** 3D Scatterplot of Each Step in 3D Space

### Step 3: Correlation Analysis

Figure 3.1 below shows the correlation matrix, which backs up the discoveries made in Figures 2.2 and 2.3. A lot of the coordinates do not change in certain steps, and we can see that above as they are represented as blank white squares. However, we do see only perfectly positive linear correlations between variables, which means that everything in figure 2.3 can be imagined to be moving in their associated positive directions. It is also understandable that steps 8 and 9 have 4 linear blocks on their correlation matrices due to the fact that these are the two linear relationships shown in figure 2.3.



**Figure 3.1: Correlation Matrix**

I believe that these correlations shown above in figure 3.1 will have a beneficial impact on my predictions, as they seem to be quite defined with respect to one another as well as in 3D space. I also wanted to note that this correlation matrix was created on a step basis, as you can see there are 13 of them. This is because I thought it may be a bit useless to correlate the coordinates to one another across all steps, as they may have little to no relationship with one another. Or if they did for one particular step, there would be no way of knowing unless I separated them all.

## Step 4: Correlation Analysis

The 4 machine learning algorithms that were chosen are as follows:

1. Support Vector Machine (SVM)
2. Random Forest
3. Decision Tree
4. Logistic Regression

For obvious reasons, these were all chosen because they are able to be used as classification models, and quite honestly, they were selected because I knew the most about how they actually worked, and had already looked at their documentation and hyperparameters prior to starting the project through the lecture notes.

For the SVM, RandomForest and DecisionTree models, I had no doubts about their applicability to this project/data set due to their multi-classification capabilities and the nature of how their algorithms functioned. The only one I was skeptical about was the Logistic Regression, which, from the lecture notes, I only thought I would be able to apply to data which could be classified on a binary basis. However, after reading, I understood that you are able to use multinomial logistic regression, which can extend logistic regression to all available classes simultaneously.

## Step 5: Model Performance Analysis

Before we look into the performance metrics of the 4 trained models, it is important to understand the meaning of the metrics that we are using to evaluate them. The following parameters are the main ones used:

- F1 Score (Macro)
- Precision (Macro)
- Accuracy

**Accuracy** measures the proportion of correctly predicted instances out of the total instances. It is a straightforward metric that gives an overall sense of model performance. Using a target board example, high accuracy would be hits on the bullseye of the target. While accuracy is intuitive and easy to interpret, it can be misleading in cases where the dataset is imbalanced, as our dataset is. It can be calculated by:

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

**Precision** focuses on the quality of positive predictions made by the model. It measures the proportion of true positive predictions among all positive predictions. High precision indicates that when the model predicts a positive class, it is usually correct. Think of closely grouped groupings on a target board (even if they are not on the bullseye). This metric is especially important in applications where false positives are costly or undesirable, which could apply to this dataset given its application in aircraft maintenance work, leading to lost time or possibly damaged parts.

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

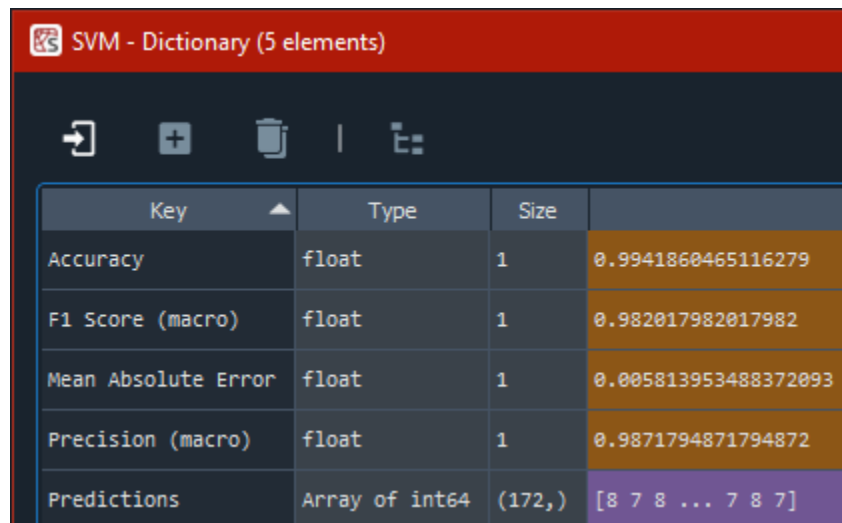
The **F1 score** is the harmonic mean of precision and recall, providing a balance between the two. It is particularly useful when dealing with imbalanced datasets or when both false positives and false negatives carry significant consequences. By combining precision and recall, the F1 score gives a single metric that reflects the model's overall ability to correctly identify positive cases. However, **the F1 Macro score** accounts for class imbalance by calculating the F1 score for each class and then averaging them together.

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

Here, the F1 (macro) score will be used as accuracy can be misleading in cases of imbalanced data sets, and the F1 score takes into account precision already, and even accounts for another metric we did not talk about, recall. The f1 macro score is best suited for scoring each of these models due to its ability to account for class imbalances.

```
--SVM--
Fitting 5 folds for each of 80 candidates, totalling 400 fits
SVM Done!
Best SVM params: {'C': 100, 'degree': 3, 'gamma': 'scale', 'kernel': 'poly'}
```

**Figure 5.1: SVM Best Parameters**



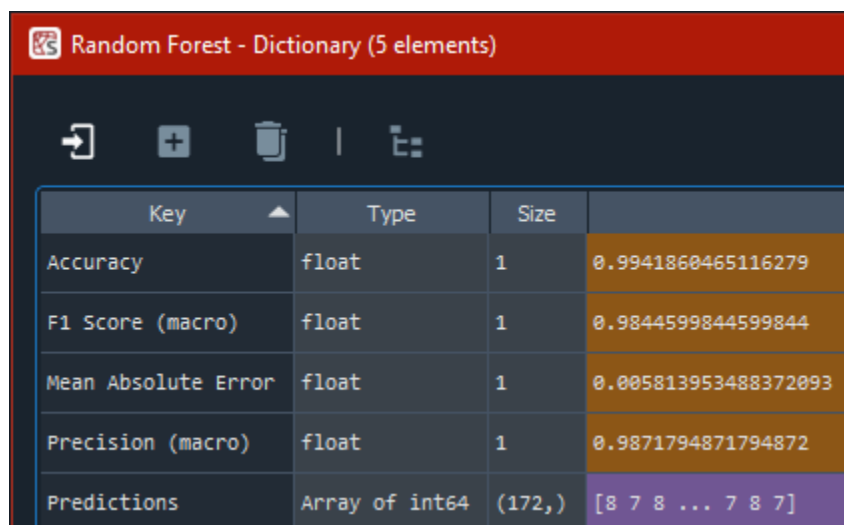
| Key                 | Type           | Size   |                      |
|---------------------|----------------|--------|----------------------|
| Accuracy            | float          | 1      | 0.9941860465116279   |
| F1 Score (macro)    | float          | 1      | 0.982017982017982    |
| Mean Absolute Error | float          | 1      | 0.005813953488372093 |
| Precision (macro)   | float          | 1      | 0.9871794871794872   |
| Predictions         | Array of int64 | (172,) | [8 7 8 ... 7 8 7]    |

**Figure 5.2: SVM Performance Metrics**



```
--RF--
Fitting 5 folds for each of 864 candidates, totalling 4320 fits
RandomForest Done!
Best RF params: {'bootstrap': False, 'criterion': 'entropy', 'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 1,
'min_samples_split': 2, 'n_estimators': 100}
```

**Figure 5.3:** RF Best Parameters



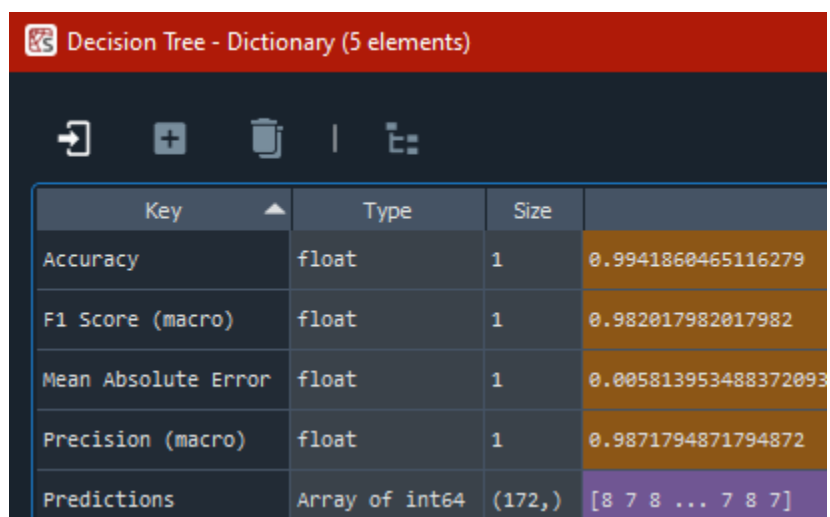
Random Forest - Dictionary (5 elements)

| Key                 | Type           | Size   |                      |
|---------------------|----------------|--------|----------------------|
| Accuracy            | float          | 1      | 0.9941860465116279   |
| F1 Score (macro)    | float          | 1      | 0.9844599844599844   |
| Mean Absolute Error | float          | 1      | 0.005813953488372093 |
| Precision (macro)   | float          | 1      | 0.9871794871794872   |
| Predictions         | Array of int64 | (172,) | [8 7 8 ... 7 8 7]    |

**Figure 5.4:** RF Performance Metrics

```
DecisionTree Done!  
Best DT params: {'criterion': 'gini', 'max_depth': None, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_split': 2,  
'splitter': 'random'}
```

**Figure 5.5:** DT Best Parameters



| Key                 | Type           | Size   |                      |
|---------------------|----------------|--------|----------------------|
| Accuracy            | float          | 1      | 0.9941860465116279   |
| F1 Score (macro)    | float          | 1      | 0.982017982017982    |
| Mean Absolute Error | float          | 1      | 0.005813953488372093 |
| Precision (macro)   | float          | 1      | 0.9871794871794872   |
| Predictions         | Array of int64 | (172,) | [8 7 8 ... 7 8 7]    |

**Figure 5.6:** DT Performance Metrics

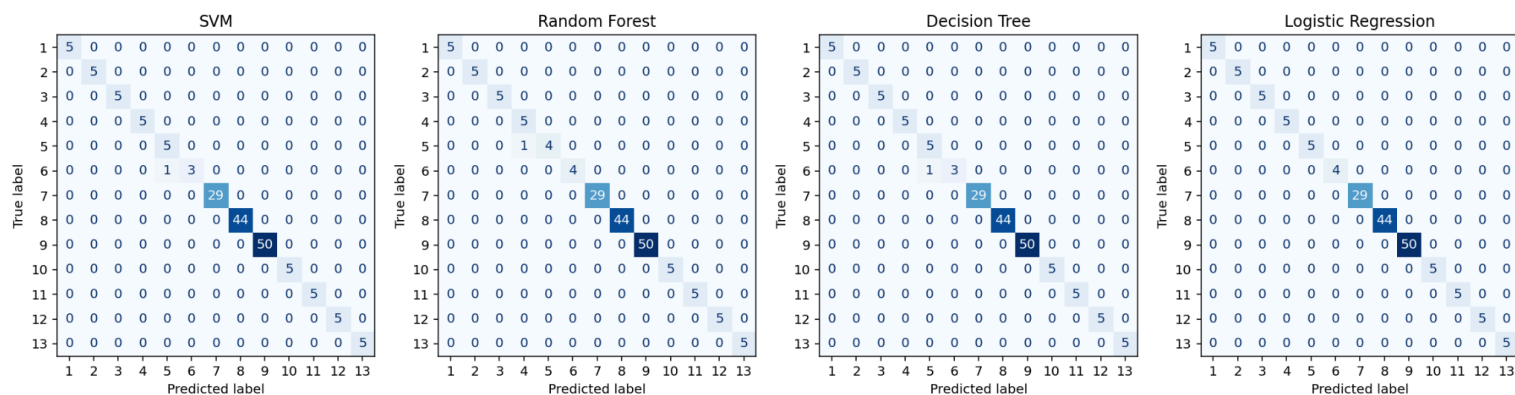
```
LogisticRegression Done!  
Best LReg params: {'solver': 'saga', 'penalty': 'l1', 'l1_ratio': np.float64(0.3333333333333333), 'C':  
np.float64(3792.690190732246)}
```

**Figure 5.7: LReg Best Parameters**

Logistic Regression - Dictionary (5 elements)

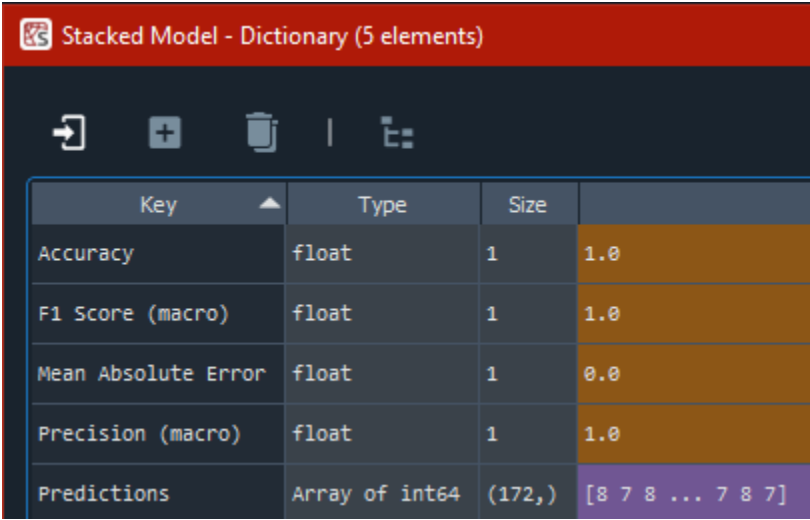
| Key                 | Type           | Size   |                   |
|---------------------|----------------|--------|-------------------|
| Accuracy            | float          | 1      | 1.0               |
| F1 Score (macro)    | float          | 1      | 1.0               |
| Mean Absolute Error | float          | 1      | 0.0               |
| Precision (macro)   | float          | 1      | 1.0               |
| Predictions         | Array of int64 | (172,) | [8 7 8 ... 7 8 7] |

**Figure 5.8: LReg Performance Metrics**



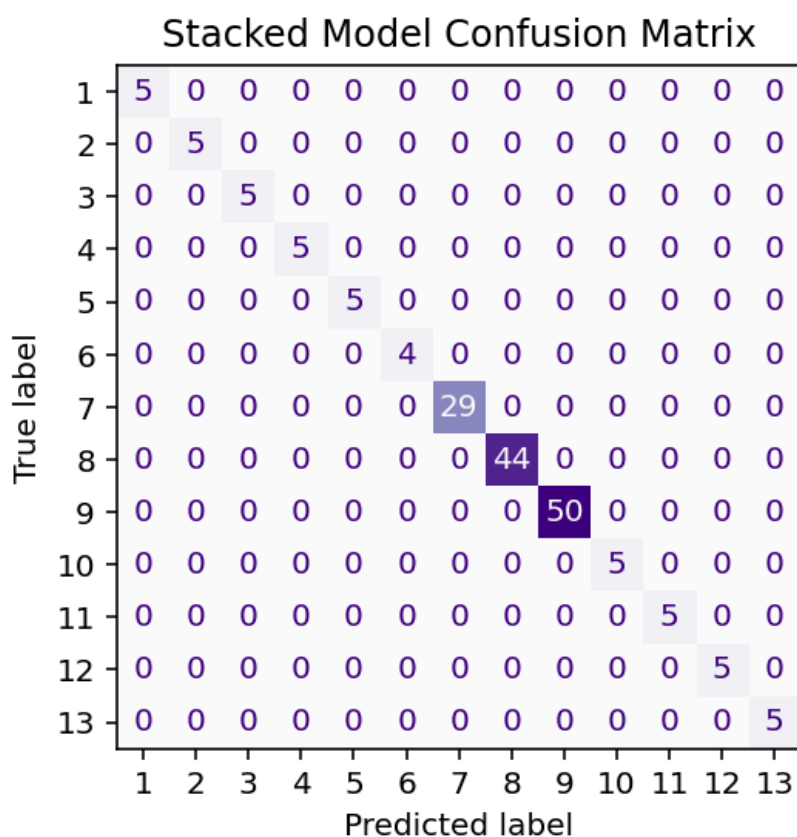
**Figure 5.19:** Confusion Matrix of Each of the 4 Models

## Step 6: Stacked Model Performance Analysis



| Key                 | Type           | Size   | Value             |
|---------------------|----------------|--------|-------------------|
| Accuracy            | float          | 1      | 1.0               |
| F1 Score (macro)    | float          | 1      | 1.0               |
| Mean Absolute Error | float          | 1      | 0.0               |
| Precision (macro)   | float          | 1      | 1.0               |
| Predictions         | Array of int64 | (172,) | [8 7 8 ... 7 8 7] |

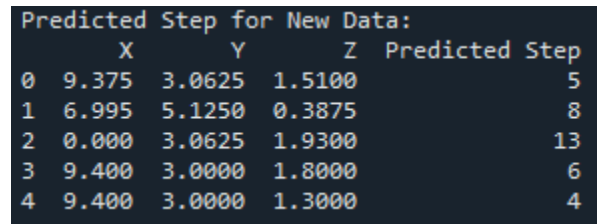
**Figure 6.1:** Stacked Model (SVM-RF) Performance Metrics



**Figure 6.2:** Confusion Matrix of SVM-RF Stacked Model

A stacked model combines multiple base models to improve overall predictive performance. Instead of relying on a single model, stacking leverages the strengths of different models to reduce their individual weaknesses. It is almost as if I had a math problem, and instead of 1 opinion, I had 2 opinions. I believe the combined strengths of the SVM and RF models have a beneficial effect due to this very reason. There is simply just another layer of judgment.

## Step 7: Model Evaluation



| Predicted Step for New Data: |       |        |        |                |
|------------------------------|-------|--------|--------|----------------|
|                              | X     | Y      | Z      | Predicted Step |
| 0                            | 9.375 | 3.0625 | 1.5100 | 5              |
| 1                            | 6.995 | 5.1250 | 0.3875 | 8              |
| 2                            | 0.000 | 3.0625 | 1.9300 | 13             |
| 3                            | 9.400 | 3.0000 | 1.8000 | 6              |
| 4                            | 9.400 | 3.0000 | 1.3000 | 4              |

Figure 7.1: Final Predictions Based on .Joblib