Toronto Metropolitan University

Department of Aerospace Engineering
Faculty of Engineering & Architectural Science

Semester (Term, Year)	Fall 2024
Course Code	AER 850
Course Section	01
Course Title	Machine Learning
Course Instructor	Dr. Reza Faieghi
Submission	Project 1
Submission No.	1
Submission Due Date	October 20, 2024
Title	Project 1
Submission Date	October 21, 2024

Name	**Student Number**	**Signature**
Shayaan Shaikh	501028734	[illegible]

Table of Contents

List of Figures

List of Tables

Objective

The objective of this project was to familiarize students with the end to end ML model pipeline and classification models. The maintenance stage of various parts of the inverter of the FlightMax Fill Motion Simulator (outcome measure) needed to be predicted based on their X, Y, and Z location (features) using classification methods. Data was provided for the inverter in terms of X,Y, and Z coordinates of specific parts, alongside the maintenance step that each part was at. The data was first split into training and test data, and visualized to determine its behavior. Next, a correlation analysis was done on the data to determine the relationship between features as well as the relationship between the features and the outcome measure. Using the Scikit-Learn Python package, various machine learning models were trained and tested on the data. Techniques such as grid searching, random grid searching, and model stacking were also used. The performance of each model was compared to determine the strengths and weaknesses of each. Models were also packaged in order to be callable in a function-like manner for ease of future use.

Part 1

In Part 1 of the project, the data was first read from the provided csv file and converted into a data frame using the Pandas package. Next, it was split into training and test data with an 80/20 ratio using a stratified shuffle splitter to ensure that, with respect to the original data set, the training data would have equal proportions of data points for each class. Stratified sampling is used because it reduces bias in the training data, and improves model performance on imbalanced data sets, especially when specific classes are rare [1]. Note that data splitting was done before visualization in order to avoid data snooping bias.

Part 2

In Part 2, the data was visualized using a scatter matrix. This can be seen in Figure 1 below.

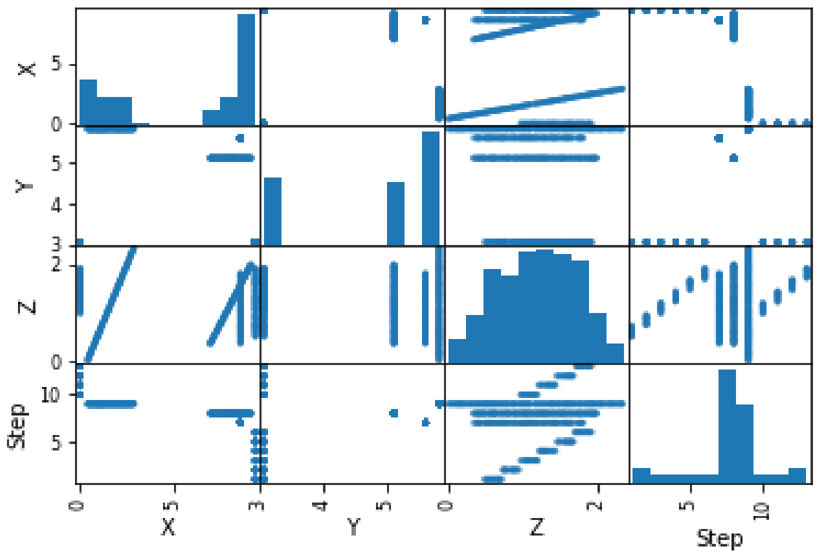

Figure 1: Scatter matrix created from training data

The relationship between the data can be seen in Figure 1. The scatter matrix was a good first way to visualize the data and see if different features and the maintenance step blatantly depended on each other. The histograms for each variable can be seen on the main diagonal. It can be seen that there is no clear relationship between each feature or between the features and the step. For example, looking at the Z vs. Y plot, the various vertical lines indicate that for various Y coordinates, various types of Z coordinates exist (no clear relationship). Additionally, looking at the Step vs. Z plot, various maintenance steps exist at all Z values. One interesting plot is the Step vs. X plot, where a tangent like behavior can be seen. That being said, there is a gap in the mid range of the X values, and thus, this relationship can not be confirmed.

The histograms indicate an imbalance in the data. For example, in the step histogram, a majority of the data points can be seen to be in the maintenance step 7-10 range, and data points at all other steps are sparse. For X, there are values at the low and high range, but little to none in the middle. For Y, there are values in the low and high range, but they are all similar. The Z values are more balanced, with many data points across the entire range.

Part 3

In Part 3, correlation analysis was done using a correlation matrix. The correlation matrix was computed for the training data frame using Pandas, and then visualized as a heatmap using Seaborn. This can be seen in Figure 2 below.

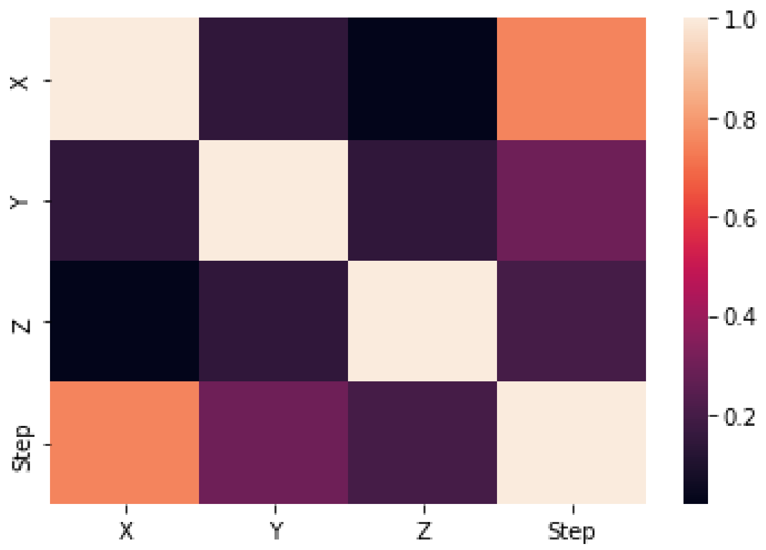

Figure 2: Correlation matrix created from training data

The linear correlations between variables can be seen in Figure 2. As expected from the scatter matrix, there is no clear linear relationship between the features. This is a good thing as features that have high correlation can reduce model performance and often need to be removed[2]. More interestingly, a high degree of correlation can be seen between the maintenance step and X. This is due to what was seen on the correlation matrix earlier, i.e., the tangent like relationship. Although not blatantly linear, the general trend that can be seen is that as the X coordinate increases, the maintenance step is more likely to be lower. This high correlation suggests that X may be a good predictor of the maintenance step and may be very important to model performance.

<u>Part 4</u>

In Part 4, 4 different models were trained and tested. Logistic Regression, Random Forest, and Support Vector Machine (SVM) models were developed using grid search cross validation (via the Scikit-Learn GridSearchCV method) to determine the best of a variety of hyperparameters that were tested for each of them (different models have different hyper parameters). This means that for each combination of parameters, the training data was split into 5 folds, and tested with 1 fold as the test data and 4 folds as the training data (done 5 times in total, with the test data being a different fold for each iteration, i.e., 5 fold cross-validation). The weighted f1 score (weighted average of f1 score for each class, weight based on the amount of the test data points that were actually in that class, called the support) was then averaged across all folds and taken as the score for the model with that specific combination of hyperparameters.

The models created with different combinations of parameters were compared using the weighted f1 score. For each model, this took the best combination of parameters as the ones which yielded a model with the highest weighted f1 score. The f1 score is a combination of precision and recall. Precision can be thought of as how many positive predictions are actually true positives [2]. Recall can be thought of as how many actual positives were correctly identified [2]. This is discussed more in Part 5. Both of these metrics are important in gaining a full picture in the assessment of any model, and thus the f1 score becomes important.

Additionally, a fourth Random Forest model was created using RandomizedSearchCV, where only a subset of the hyperparameters fed to it were tested.

Before the models were trained, the data was scaled using a standard scaler. Standard scaling ensures that the data for each variable follows a normal distribution with a mean of 0 and a standard deviation of 1. This is a way of balancing the data, ensuring that no one feature dominates the model behavior simply because it tends to be higher in magnitude. Note that the test data was scaled using the mean and standard deviation of the training data. This ensured that no data leak occurred, as if the test data was scaled with the test data mean and standard deviation, it would be a form of sharing data across the test set, falsifying any results.

Logistic Regression

Logistic regression was the first model trained and tested. It was selected because although it is ideal for binary classification, it can be extended to multiple classes through multinomial regression, tends to perform well on small to medium data sets, is computationally efficient, and is less likely to overfit data compared to other models [3]. These advantages are very important in this use case because the data set is so small, at less than 1000 data points. The parameters tested via grid search and the best model parameters can be seen in Table 1:

Table 1: Logistic regression grid search parameters used

Parameter	Function of Parameter	Options Used	Best Model
Solver	Type of solver used to minimize loss function	Lbfgs, newton-cg, sag	newton-cg
Penalty	Type of regularization applied	None, l2	None
C	Inverse of regularization strength. Smaller value tends to result in less overfitting	0.01, 0.1, 1, 10	0.01

Random Forest

A Random Forest model was the second model used. Although generally computationally complex, it was chosen because they are generally highly accurate, perform well on noisy data, and can natively handle multiclass classification problems (as opposed to Logistic Regression, which has to be expanded to multinomial regression to do so) [4]. The parameters given to the Random Forest model and the best model parameters are shown in Table 2:

Table 2: Random Forest grid search parameters used

Parameter	Function of Parameter	Options Used	Best Model
n_estimators	Number of decision trees used. Can improve model performance with increase computational cost	5, 10, 30, 50	10
max_depth	Maximum tree depth. If too high, can cause overfitting	None, 5, 15, 45	none
min_samples_split	Minimum number of samples required to split an internal node	2, 5, 10	5
min_samples_leaf	Minimum number of samples required in a leaf node	1, 2, 4, 6	1
max_features	Number of features considered to determine best fit	None, 0.1, sqrt, log2, 1, 2, 3	0.1
criterion	Function used to measure split quality	Gini, entropy	entropy

Support Vector Machine (SVM)

The third model used was an SVM classifier. It was used because it works well with small data sets, is computationally efficient, and performs well when the number of features does not exceed the number of training data samples (as in this case) [5]. The parameters tested via grid search and the best model parameters can be seen in Table 3:

Table 3: SVM grid search parameters used

Parameter	Function of Parameter	Options Used	Best Model
gamma	Affects the shape of the decision boundary and how flexible it is. Larger values correspond to more rigid boundaries.	Scale, auto, 10, 100	10
kernel	Function to transform the data into higher dimensional space	Linear, poly, rbf, sigmoid	poly
C	Inverse of regularization strength. Smaller value tends to result in less overfitting	0.001, 0.01, 0.1, 1, 10	0.01

Random Forest with RandomizedSearchCV

Unlike grid search, RandomizedSearchCV only tries some combinations of the parameters provided [6]. In this case, 10 combinations of parameters were tested by setting the "n_iter" parameter (of the RandomizedSearchCV object) to 10. Another Random Forest model was created to use with RandomizedSearchCV, with the parameters and best model shown in Table 4.

Table 4: Random Forest RandomizedSearchCV parameters used

Parameter	Function of Parameter	Options Used	Best Model
n_estimators	Number of decision trees used. Can improve model performance with increase computational cost	5, 10, 15, 20, 25, 30, 50	50
max_depth	Maximum tree depth. If too high, can cause overfitting	None, 5, 10, 15, 25, 45	10
min_samples_split	Minimum number of samples required to split an internal node	2, 5, 10, 15	2
min_samples_leaf	Minimum number of samples required in a	1, 2, 4, 6, 12	4

	leaf node		
max_features	Number of features considered to determine best fit	None, 0.1, sqrt, log2, 1, 2, 3	None
criterion	Function used to measure split quality	Gini, entropy	gini

Part 5

In Part 5, the overall performance of each model using f1 score, precision, and accuracy was determined. This was done by generating a classification report and accuracy scores (test and training accuracies) for each model.

In order to determine the best metric for this use case, the equations for accuracy, precision, recall, and the f1 score must be examined. These can be seen in Equation (1), Equation (2), Equation (3), and Equation (4), respectively.

$$Accuracy = \frac{Correct\ Predictions}{Total\ Predictions} \tag{1}$$

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \tag{2}$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{3}$$

$$f1 = 2 \times \frac{Precision \times Recall}{Precision \times Recall} \tag{4}$$

The accuracy can provide an idea of how many correct predictions are made, but is not a good metric to use in this case, since the data is imbalanced, which can make accuracy misleading [2]. For example, consider a data set where 1 datapoint of 100 was positive, and 99 were negative. If the model predicted 100/100 to be negative, it would have 99% accuracy, but clearly, it would have missed the 1 only positive data point.

The precision is a measure of how many of the positive predictions were actually positive [2]. In a multiclass situation like this one, the precision needs to be calculated for each class. For example, what percentage of parts predicted to be class 1 were actually class 1. This is more useful than accuracy in this case, since it can be calculated for each class and help some of the imbalance. That being said, for classes where data points are sparse, the precision can be artificially inflated, which must be kept in mind.

The recall, or sensitivity is a measure of how many of the true positives were correctly identified [2]. High recall means there are few false negatives. For example, how many of the actual parts that were at step 1 were correctly predicted to be at step 1? Since the data set is imbalanced, recall is useful since it will be low if positive cases are not being detected, even in the case of more sparse classes.

The f1 score is a combination of precision and recall [2]. It is most useful in this use case, because the data set is not balanced. Precision is useful, but can be misleadingly high [2], and recall can balance it. Similarly, recall can be high in the case of little false negatives, while precision may be low due to false positives. Thus, in assessing the models for this project, the f1 score can be used. In particular, the weighted f1 score is used, increasing its usefulness as it considers the weight fraction of each class in the test data, which is particularly important due to the sparsity of some classes in the data set.

Logistic Regression

The classification report and accuracies for the Logistic Regression model can be seen in Table 5 and Table 6 below.

Table 5: Logistic Regression classification report

	Precision	Recall	f1-score	Support
1	1	1	1	4
2	1	0.8	0.888888888	5
3	0.833333333	1	0.909090909	5
4	1	1	1	5
5	1	1	1	5
6	1	1	1	5
7	1	1	1	29
8	1	1	1	44
9	1	1	1	50

10	1	1	1	5
11	1	1	1	5
12	0.83333333	1	0.909090909	5
13	1	0.8	0.888888888	5
Accuracy	0.98837209			
Macro avg	0.97435897	0.969230769	0.968919968	172
Weighted avg	0.99031007	0.988372093	0.988254639	172

Table 6: Logistic Regression Accuracies

Metric	Score
Accuracy	0.988372093
Training Accuracy	1

The model performs well, with a weighted f1 score of about ~98.8%, weighted precision score of ~99.0%, and a test accuracy of ~98.8 percent. This model performance is good, but so high to the point where overfitting is likely occurring. This is further indicated by the high training accuracy of 100%, which means the model is learning the noise and fluctuation of the training data set. Although overfitting can be adjusted with hyperparameters, for example, by decreasing C, the data set is very small, and so tuning the parameters accordingly resulted in no change or extremely low performance. This is a common trend that occurs when using powerful models designed for data sets much larger than this use case, i.e. under 1000 points. Even though Logistic Regression works well for simpler data, this data set was still too small to avoid overfitting.

The confusion matrix can be seen in Figure 3 below:

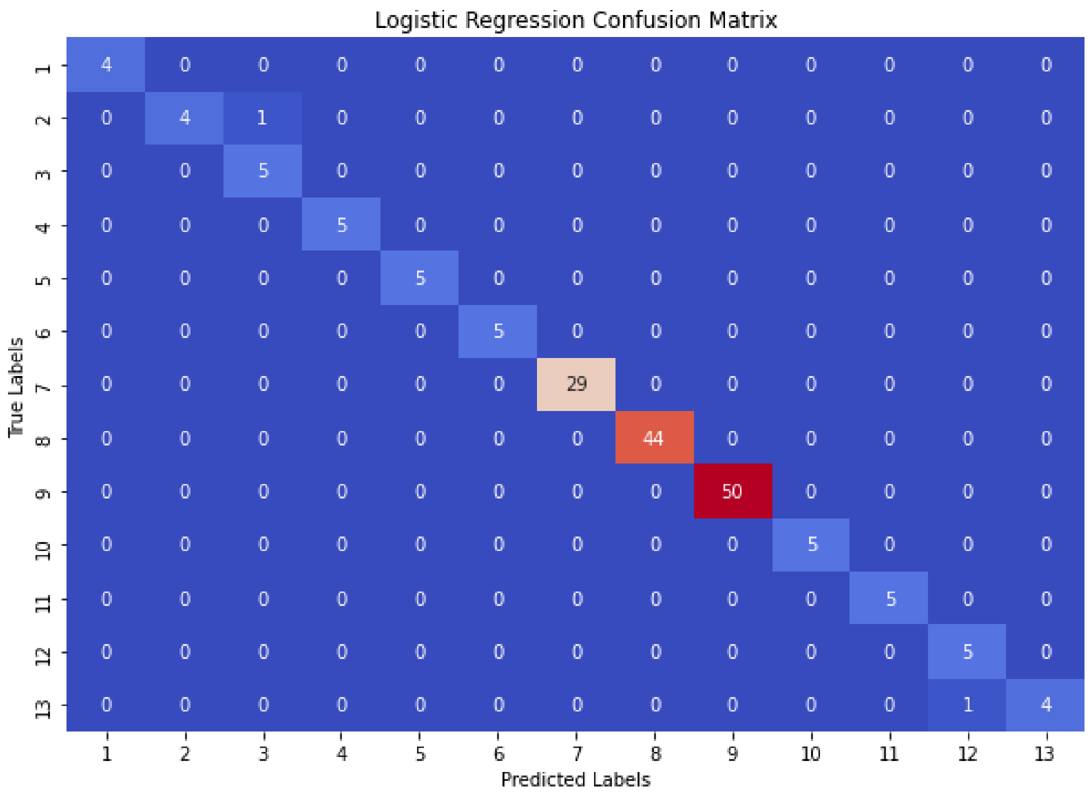

Figure 3: Logistic Regression confusion matrix

The high magnitude of the main diagonal of Figure 3 indicates a high number of true positives, and the low values in the off diagonal indicate a low number of false predictions. Although performance is high, it is almost perfect, and indicative of overfitting.

Random Forest

The classification report and accuracies for the Random Forest model can be seen in Table 7 and Table 8 below.

Table 7: Random Forest classification report

	Precision	Recall	f1-score	Support
1	1	1	1	4
2	1	0.8	0.888888889	5
3	0.833333333	1	0.909090909	5
4	1	1	1	5

5	1	1	1	5
6	1	1	1	5
7	1	1	1	29
8	1	1	1	44
9	1	1	1	50
10	1	1	1	5
11	1	1	1	5
12	0.833333333	1	0.909090909	5
13	1	0.8	0.888888889	5
Accuracy	0.988372093			
Macro avg	0.974358974	0.969230769	0.968919969	172
Weighted avg	0.990310078	0.988372093	0.988254639	172

Table 8: Random Forest accuracies

Metric	Score
Accuracy	0.9883721
Training Accuracy	1

Identical to the Logistic Regression model, The Random Forest model performs well, with a weighted f1 score of ~98.8%, weighted precision score of ~99.0%, and test accuracy of ~98.8%. The Random Forest model is likely overfitting the data. More complex than the Logistic Regression, it was more likely to overfit the small, simple data set, especially if a simpler model such as the Logistic Regression model did.

The confusion matrix can be seen in Figure 4 below:

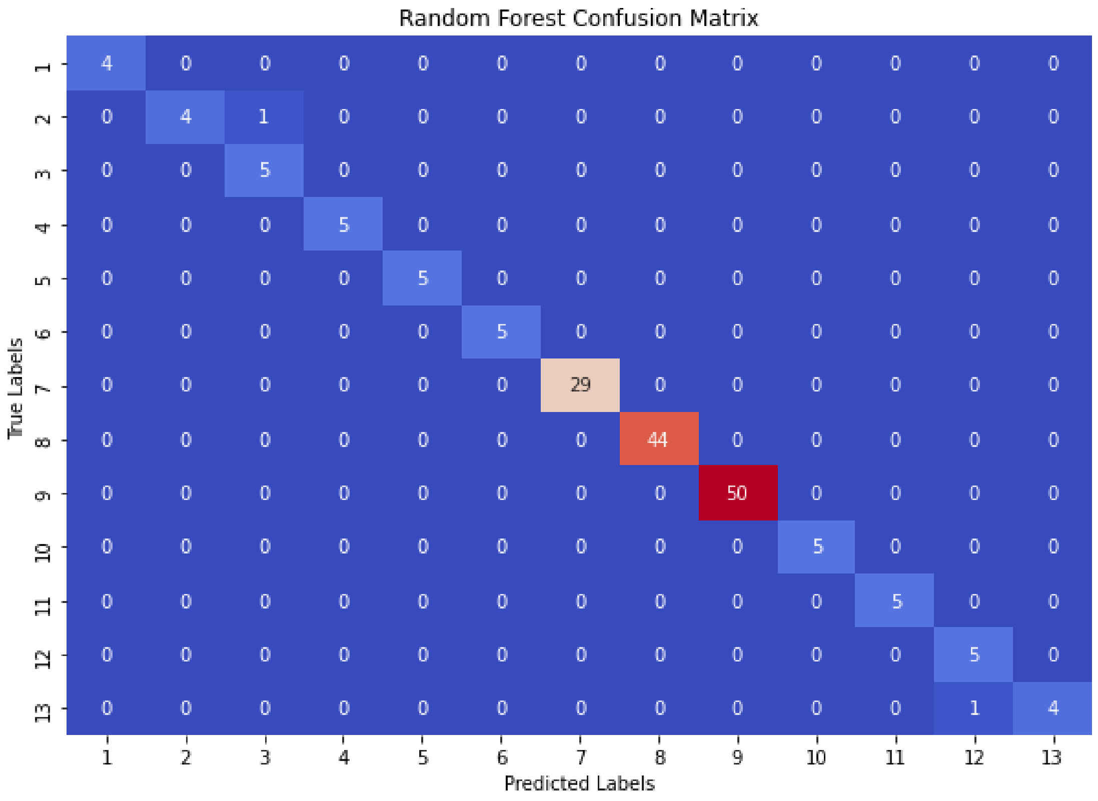

Figure 4: Random Forest confusion matrix

The high magnitude of the main diagonal of Figure 4 indicates a high number of true positives, and the low values in the off diagonal indicate a low number of false predictions. However, the data is likely being overfit due to the near-perfect performance.

Support Vector Machine (SVM)

The classification report and accuracies for the SVM model can be seen in Table 9 and Table 10 below.

Table 9: SVM classification report

	Precision	Recall	f1-score	Support
1	1	1	1	4
2	1	1	1	5
3	0.833333333	1	0.9090909	5
4	0.8	0.8	0.8	5
5	1	0.8	0.8888889	5

6	1	1	1	5
7	1	1	1	29
8	1	1	1	44
9	1	1	1	50
10	1	1	1	5
11	1	1	1	5
12	0.833333333	1	0.9090909	5
13	1	0.8	0.8888889	5
Accuracy	0.98255814			
Macro avg	0.958974359	0.953846154	0.9535354	172
Weighted avg	0.984496124	0.98255814	0.9824407	172

Table 10: SVM accuracies

Metric	Score
Accuracy	0.98255814
Training Accuracy	0.998546512

The SVM model has a high performance, with a weighted f1 score of ~98.2%, weighted precision score of ~98.4%, and a test accuracy ~98.3%. Although it performs slightly worse than the Random Forest and Logistic Regression models, the difference is negligible, and although performance is good, the data is likely being overfit.

The confusion matrix can be seen in Figure 5 below:

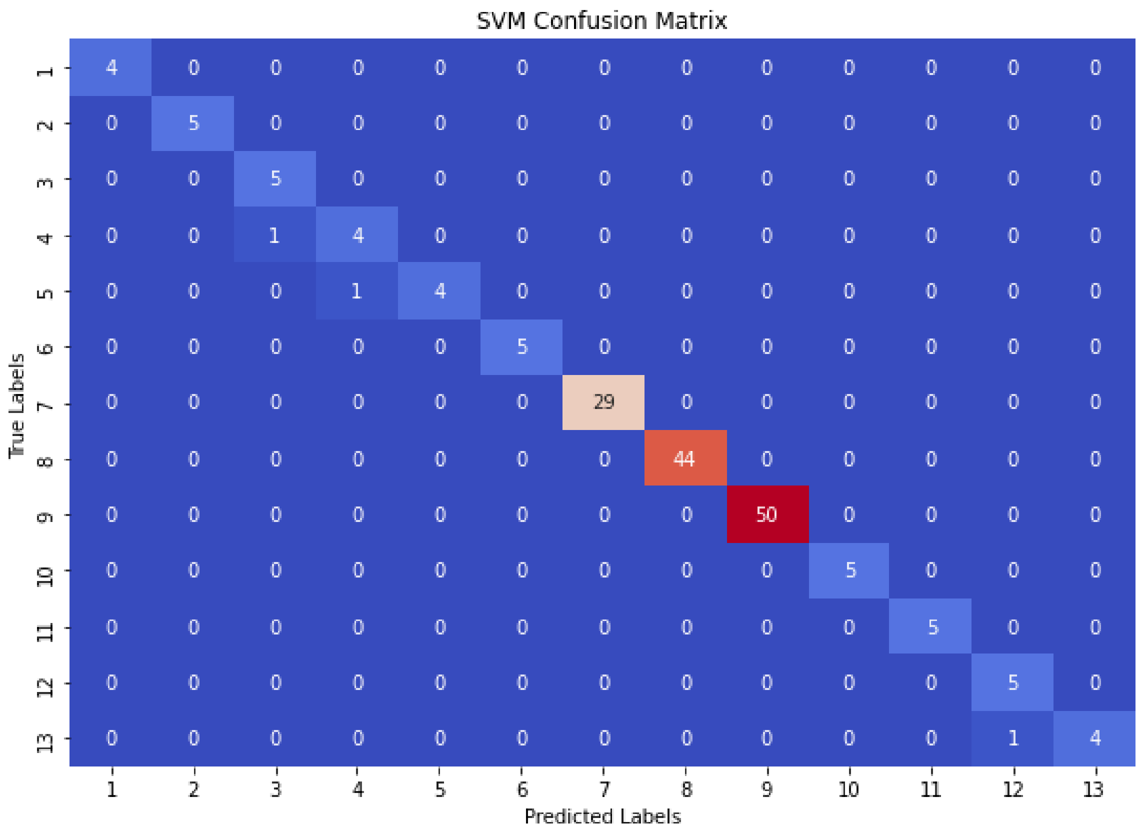

Figure 5: SVM confusion matrix

The high magnitude of the main diagonal of Figure 5 indicates a high number of true positives, and the low values in the off diagonal indicate a low number of false predictions. Although performance is high, it is near perfect, likely indicating overfitting.

Random Forest with RandomizedSearchCV

The classification report and accuracies for the RandomizedSearchCV Random Forest model can be seen in Table 11 and Table 12 below:

Table 11: SVM classification report

	Precision	Recall	f1-score	Support
1	1	1	1	4
2	1	0.8	0.888888889	5
3	0.714285714	1	0.833333333	5

4	1	0.8	0.888888889	5
5	1	1	1	5
6	1	1	1	5
7	1	1	1	29
8	1	1	1	44
9	1	1	1	50
10	1	1	1	5
11	1	1	1	5
12	0.833333333	1	0.909090909	5
13	1	0.8	0.888888889	5
Accuracy	0.98255814			
Macro avg	0.965201465	0.953846154	0.954545455	172
Weighted avg	0.986849391	0.98255814	0.98282241	172

Table 12: Random Forest with RandomizedSearchCV accuracies

Metric	Score
Accuracy	0.98255814
Training Accuracy	1

The second Random Forest model has a high performance, with a weighted f1 score of ~98.2%, weighted precision score of ~98.7%, and a test accuracy of ~98.3%. It performs similarly to the other models, and overfits the data. That being said, it is noteworthy that it performs significantly faster than the original Random Forest model, and produces similar results. This likely indicates that this model is too complex for this test case, and speaks to the potential diminishing returns of GridSearchCV.

The confusion matrix can be seen in Figure 6 below:

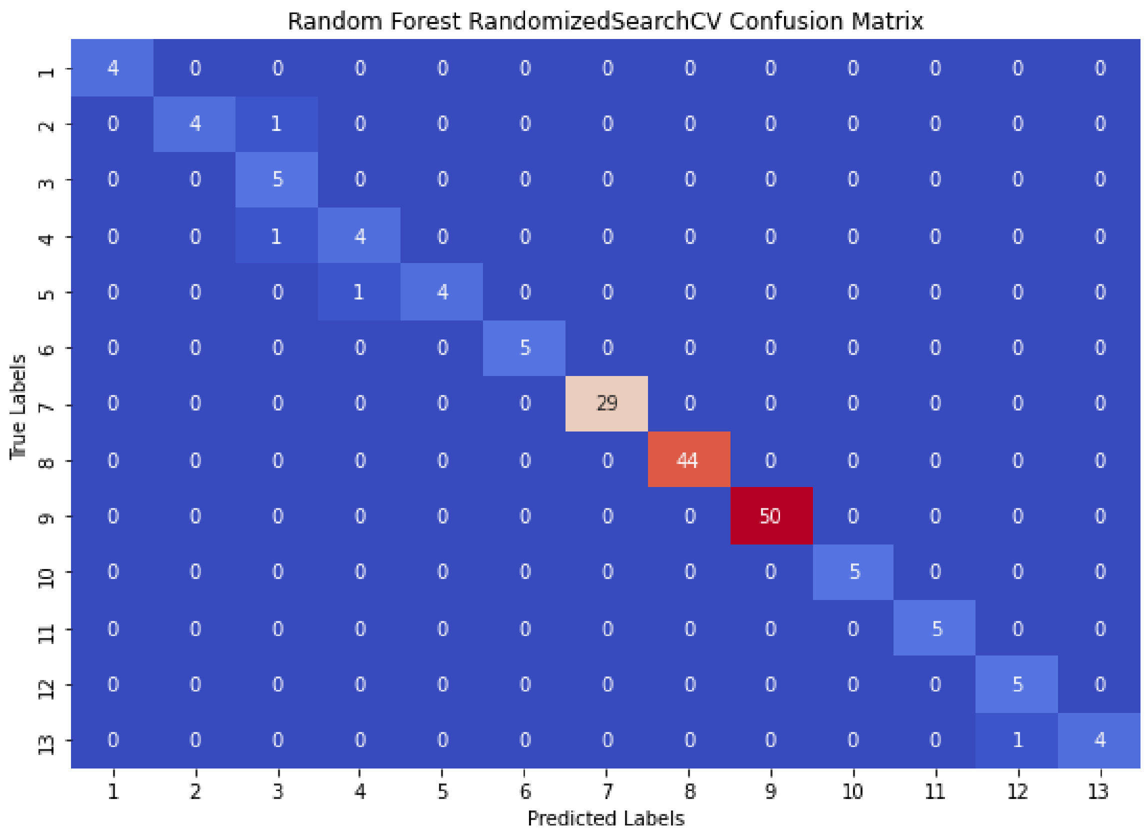

Figure 6: RandomizedSearchCV confusion matrix

The high magnitude of the main diagonal of Figure 6 indicates a high number of true positives, and the low values in the off diagonal indicate a low number of false predictions. The extremely low number of false predictions likely indicates overfitting.

Part 6

In Part 6, the Scikit-Learn StackingClassifier was used to combine two previously trained models to determine the impact of model stacking on performance. The StackingClassifier takes the two base estimators that are input by the user, and then for a given prediction, uses their output as the input for a third final classifier to determine the final prediction [7]. Cross validation can also be done. In this case the RandomizedSearchCV Random Forest and SVM models were passed to the stacking classifier and 5-fold cross validation was conducted. The classification report and accuracies for the stacked model can be seen in Table 13 and Table 14 below:

Table 13: Stacked model classification report

	Precision	recall	f1-score	support
1	1	1	1	4

2	1	1	1	5
3	0.833333333	1	0.909090909	5
4	0.8	0.8	0.8	5
5	1	0.8	0.888888888	5
6	1	1	1	5
7	1	1	1	29
8	1	1	1	44
9	1	1	1	50
10	1	1	1	5
11	1	1	1	5
12	0.833333333	1	0.909090909	5
13	1	0.8	0.888888888	5
Accuracy	0.982558139			
Macro avg	0.958974359	0.953846153	0.953535353	172
Weighted avg	0.984496124	0.982558139	0.982440686	172

Table 14: Stacked model accuracies

Metric	Score
Accuracy	0.982558139
Training Accuracy	0.998546512

The stacked model has a high performance, with a weighted f1 score of ~98.2%, weighted precision score of ~98.4%, and a test accuracy of ~98.3%. It performs nearly identically to every other model, including the SVM and RandomizedSearchCV models that serve as its base estimators. Thus, the stacking had little to no difference on model performance. This is likely due to the fact that each model was overfitting the sparse data set. Introducing another layer of complexity by stacking models would only exacerbate this effect or have no impact on performance, which was shown to be the case.

The confusion matrix can be seen in Figure 7:

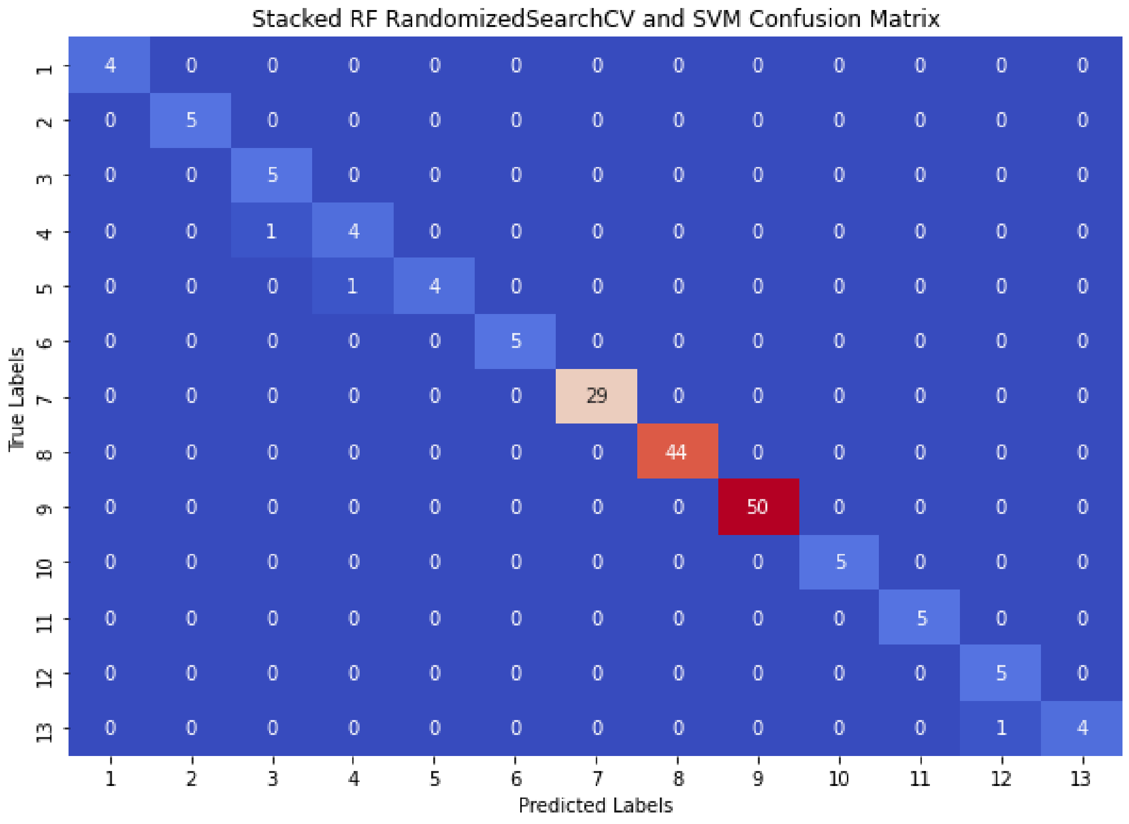

Figure 7: Stacked model confusion matrix

As in previous models, The high magnitude main diagonal and low magnitude off diagonal in the stacked model confusion matrix shown in Figure 7 is indicative of high performance, but overfitting.

Part 7

In Part 7, the stacked model was packaged as a .pkl file using the Joblib package, such that it could be called in a function-like manner to predict the maintenance step of any part given its X, Y, and Z coordinates. For the additional data provided, the stacked model predictions can be seen in Table 15

Table 15: Stacked model predictions on new data

X	Y	Z	Prediction Class
9.375	3.0625	1.51	8
6.995	5.125	0.3875	7
0	3.0625	1.93	9
9.4	3	1.8	7
9.4	3	1.3	7

Conclusion

In conclusion, this project provided exposure to multiclass classification using a simple, small data set of less than 1000 data points. Data was split into training and test data, and the training data was visualized. A correlation analysis was also done on the training data. Further, standard scaling was applied to the training data and test data, using only the mean and standard deviation of the training data to avoid a data leak. Three models were trained using GridSearchCV, one model was trained with RandomizedSearchCV, and a fifth model was created by stacking 2 models. The performance of all models was compared using their weighted f1 score, test accuracy, and precision score. It was found that all 3 metrics for all 5 models always exceeded 98%, a sure indicator that overfitting was occurring for each model. When model hyperparameters were tuned to a lower complexity to decrease overfitting, model performance either stayed the same, or the parameters were so low such that model performance completely disappeared. Thus, this provides the main takeaway for this project: in order to use complicated classification models, or even simpler ones such as Logistic Regression, a larger data set should be used. Additionally, although GridSearchCV is useful for assessing combinations of a variety of hyperparameters, it is more useful when constant overfitting is not occurring. In the future, a variety of smaller (i.e., those resulting in less complex models) hyperparameters can be used in GridSearchCV to reduce overfitting, or the model should be manually tuned to a state of overfitting and then tuned downwards such that it can approach a best fit. Additionally, larger data sets should be used to avoid overfitting, or if more data is not available, techniques such as data augmentation can be employed to artificially increase the amount of data.

References

[1] GeeksforGeeks, “Stratified Sampling in Machine Learning,” *GeeksforGeeks*, Apr. 02, 2024. https://www.geeksforgeeks.org/stratified-sampling-in-machine-learning/

[2] R. Faieghi, (2024). AER850 – Intro to Machine Learning (Classification Problems), lecture, Toronto Metropolitan University.

[3] A. R. Rout, “Advantages and Disadvantages of Logistic Regression,” *GeeksforGeeks*, Jan. 10, 2023. https://www.geeksforgeeks.org/advantages-and-disadvantages-of-logistic-regression/

[4] K. Dhiraj, “Top 4 advantages and disadvantages of Support Vector Machine or SVM,” *Medium*, Sep. 25, 2020. https://dhirajkumarblog.medium.com/top-4-advantages-and-disadvantages-of-support-vector-machine-or-svm-a3c06a2b107

[5] GeeksForGeeks, “What are the Advantages and Disadvantages of Random Forest?,” *GeeksforGeeks*, Feb. 15, 2024. https://www.geeksforgeeks.org/what-are-the-advantages-and-disadvantages-of-random-forest/

[6] Scikit-Learn, “RandomizedSearchCV,” *scikit-learn*, 2024. https://scikit-learn.org/1.5/modules/generated/sklearn.model_selection.RandomizedSearchCV.html

[7] Scikit-Learn, “StackingClassifier,” *scikit-learn*, 2024. https://scikit-learn.org/dev/modules/generated/sklearn.ensemble.StackingClassifier.html (accessed Oct. 21, 2024).

Appendix

GitHub Repository Link: https://github.com/kingy260/Project_1