# Cover page

# Tables of Content

[Cover page 1](#_Toc148300210)

[Tables of Content 2](#_Toc148300211)

[List of Tables and Figures 2](#_Toc148300212)

[1. Background & Introduction 3](#_Toc148300213)

[2. Project Outline 3](#_Toc148300214)

[2.1 Part 1: Data Processing 4](#_Toc148300215)

[2.2 Part 2: Data Visualization 4](#_Toc148300216)

[2.3 Part 3: Correlation Analysis 6](#_Toc148300217)

[2.4 Part 4: Classification Model Development/Engineering 7](#_Toc148300218)

[2.5 Part 5: Model Performance Analysis 7](#_Toc148300219)

[2.6 Part 6: Model Evaluation 8](#_Toc148300220)

[3. Conclusion 10](#_Toc148300221)

[Appendix 11](#_Toc148300222)

[GitHub Link 11](#_Toc148300223)

[Python Code 11](#_Toc148300224)

# List of Tables and Figures

[**Figure 2.0. 1: Inverter of the FlightMax Fill Motion Simulator** 3](#_Toc148298266)

[**Figure 2.1. 1: Python console output for DataFrame from .csv file** 4](#_Toc148298293)

[**Table 2.2. 1: Basic statistical analysis of raw data** 4](#_Toc148298331)

[**Figure 2.2. 1: Scatter plot of X, Y, Z Coordinate based on Step** 5](#_Toc148298396)

[**Figure 2.3. 1: Correlation plot between feature and target variables** 6](#_Toc148298409)

[**Table 2.5. 1: Performance Analysis of the 3 chosen ML models** 7](#_Toc148298431)

[**Figure 2.5. 1: Confusion Matrix of the Support Vector Classifier model** 8](#_Toc148298455)

[**Table 2.6. 1: Predicted Maintenance Step from provided data** 8](#_Toc148298469)

[**Figure 2.6. 1: Comparison of Predicted to Original Data** 9](#_Toc148298502)

# 1. Background & Introduction

Predictive Machine Learning algorithms together with a large numbers of data set can be used to develop Augmented Reality (AR) through instruction modules. The computational ability becoming more powerful than ever, machine learning models are becoming more sophisticated and popular. In the Aerospace industry, machine learning is mainly used in the manufacturing and maintenance sectors, there is a great potential to improve its efficiency and accuracy. At a micro-level approach, the coordinate system is incorporated into various machine learning models to enhance existing augmented reality animation methods. The Aircraft Maintenance Engineers (AME) nowadays are required to handle a big amount of workload in order to service airplanes. With such a complex mechanical structure, it is extremely complicated to perform such tasks. Therefore, they would benefit greatly from such Machine Learning models to help guide through stages of maintenance thus helping everyone to work safely and efficiently. Beyond just maintenance, the aerospace industry could enjoy such benefits at other stages like computer-aided design, assembly, electrical service and even space related applications.

# 2. Project Outline

This project utilizes various supervised classification Machine Learning models to predict the maintenance step/stage based on 3-D coordinates. There are many different approaches to solving this project as there are many Machine Learning models out there, each has its strengths and weaknesses. The purpose is the asses the few of the best ML to successfully use train and test data and predict new data. This project uses an inverter of the FlightMax Fill Motion Simulator as shown in Figure 1 below:

A machine with wires and a box

Description automatically generated

**Figure 2.0. 1: Inverter of the FlightMax Fill Motion Simulator**

There are a total of 13 unique maintenance steps required to correctly disassemble the inverter of the FlightMax Fill Motion Simulator. A 3-D coordinate system is defined for each step, therefore a datapoint is formulated based on the X, Y and Z magnitude. As this is a supervised classification machine learning problem, the X, Y, Z data points are the feature variables while their corresponding Steps are the target variable. These two variables are the basis of supervised machine learning and will be demonstrated in the next part of the report. They are split in to seven stages to process and develop various Machine Learning models useful for maintenance step prediction. The following results are developed from Python Object-Oriented Programing.

# 2.1 Part 1: Data Processing

Using Pandas package in Python, the given XYZ Coordinates and Steps are converted from a .csv file into a DataFrame variable.

A table with numbers and letters

Description automatically generated

**Figure 2.1. 1: Python console output for DataFrame from .csv file**

# 2.2 Part 2: Data Visualization

Initial statistical analysis was performed to roughly understand the behaviour of the given data. This was done by using the .describe() function and the results of the feature data is presented in the table below:

**Table 2.2. 1: Basic statistical analysis of raw data**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **X** | **Y** | **Z** |
| **Count** | **860** | **860** | **860** |
| Mean | 5.5871 | 4.8456 | 1.1975 |
| Standard Deviation | 3.7191 | 1.1423 | 0.5228 |
| Min | 0.0000 | 3.0625 | 0.0000 |
| 25% | 1.5625 | 3.0625 | 0.7838 |
| 50% | 7.7700 | 5.1250 | 1.2200 |
| 75% | 8.5750 | 5.8450 | 1.6170 |
| Max | 9.3750 | 5.8450 | 2.3500 |

Based on Table 2.2.1, all X, Y and Z have an equal amount of 860 data points, therefore no missing value handling is required. The magnitude of the coordinate ranges from 0 to approximately 10, that means the inverter freely moves in the space of 10x10x10 cubic unit.

To visualize the data further, each X, Y and Z is scatter plotted against the 13 maintenance steps. This will roughly help us understand the relative position of X, Y and Z throughout each stage of maintenance:

A graph with numbers and points

Description automatically generated with medium confidence

**Figure 2.2. 1: Scatter plot of X, Y, Z Coordinate based on Step**

The figure above consists of blue circles representing X datapoints, orange squares for Y and green diamonds for Z. Each maintenance step is a class on the horizontal axis therefore it ranges from 1 to 13. Visually, this scatter plot presents the following characteristics:

-For step 1 to step 6, X is stable at 9.375, Y is stable at 3.0625, however during these steps, Z increases linearly roughly from 0.5 to 2

-For step 7 to 9, X decreases massively from 9.375 to 0, Y fluctuates arounds 5 and 6 while Z varies roughly between 0 and 2.5

-For step 10 to 13, X is constant at 0, Y returns to being constant at 3.0625 while Z increases linearly again from roughly 1 to 2.

Visually, the human brain might be able to use this data magnitude pattern to predict future datapoints. However, we cannot do that to a lot of data point especially with any levels of accuracy. That is why a machine learning algorithm is developed to recognize this pattern and predict new data is an extremely high level of accuracy.

# 2.3 Part 3: Correlation Analysis

Correlation analysis is important to assessed between the feature and the target variable. The method used for this assessment is Pearson Correlation and it is showcased in the correlation plot below:

A graph with different colored squares

Description automatically generated

**Figure 2.3. 1: Correlation plot between feature and target variables**

According to correlation plot figure 2.3.1, X coordinate has the highest absolute correlation. X is negatively correlated to maintenance step at a Pearson coefficient of -0.75. Y and Z are extremely less correlated to maintenance step compared to X. The coordinates of Y only has a Pearson coefficient of 0.29 while Z has a Pearson coefficient of 0.20.

These correlations characteristic could have an impact on the predictions of future datapoints. The Y and Z coordinates by themselves have such a low correlation to Step, therefore they won't be of must use for machine learning. On the other hand, the coordinates of X by itself are relatively higher and can somewhat be a factor for prediction since it has a negative correlation of -0.75. This is by no means high enough by itself to predict maintenance with certainty, therefore more complex models must be developed beyond just correlation in order to predict maintenance step accurately.

# 2.4 Part 4: Classification Model Development/Engineering

The raw data are split into a train set and test at an 80%-20% ratio. Therefore, out of the 860 datapoints, 688 will be used to for training and 172 datapoints are used for testing. These train and test set are selected from the raw DataFrame by Stratified Random Sampling method. As this is supervised classification problem, the feature variables are normally scaled before being fitted into the 3 selected Machine Learning models: Support Vector Machine, K-Nearest Neighbor and Random Forest Classifier.

The Support Vector Machine Classifier model is chosen because this dataset is multi-dimensional. The SVM method has several hypermeters that can be fine-tuned to fit any types of data frame, especially those with high number of dimensions between their feature variables. The Grid Search Cross-Validation method is used to fine tune the most important hyperparameters to ensure accuracy, they are C, kernel and gamma. This characteristic also allows for linear and non-linear classification, which is extremely useful when X, Y, Z has low correlation with Step.

The K-Nearest Neighbor model is chosen because it considers the relative position of X, Y, Z. Figure 2.2.1 shows that there is a pattern emerges from step 1 to 6, 7 to 9 and 10 to 13 between the 3 dimensions. Therefore, if the k-size hyperparameter is tuned well to consider local datapoints, the K-NN method can be very effective and simple to calculate. The Grid Search CV method is also used to assess the best value for k-neighbor hyperparameter.

The Random Forest Classifier model is chosen for their ability to handle large data set since it is and combination of multiple decision trees operating in parallel. Since we have up to 860 data points, it is prone to have noise and the RFC can mitigate overfitting. The Grid Search CV method is utilized to determine to most important hyperparameters for RFC, they are: number of decision trees, max depth, min sample split, min sample leaf and max features.

# 2.5 Part 5: Model Performance Analysis

After developing the Machine Learning models in part 4, the train data sets are fitted to the train target variables. Each of the 3 models then access then test feature sets of 172 datapoints to yield 172 target prediction. These predictions are then measured against the test target set. This way, we can compare the performance between the 3 different Machine Learning models. The metrics used for this analysis are f1 score, precision and accuracy. The table below showcases the performance metrics of the 3 classification methods:

**Table 2.5. 1: Performance Analysis of the 3 chosen ML models**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Support Vector Machine** | **K-Nearest Neighbor** | **Random Forest Classifier** |
| **f1 score** | 1.000000 | 0.982018 | 0.982018 |
| **Precision** | 1.000000 | 0.987179 | 0.987179 |
| **Accuracy** | 1.000000 | 0.994186 | 0.994186 |

Based on table 2.5.1, the Support Vector Machine model predicted correctly for all of the test set, it is superior compared to the other 2 models. On the other hand, the K-Nearest Neighbor and Random Forest Classifier is less accurate, they predicted 2 incorrect datapoint in the test set. The superior performance of the SVM is visualized in the figure below:

A graph of a step

Description automatically generated with medium confidence

**Figure 2.5. 1: Confusion Matrix of the Support Vector Classifier model**

Figure 2.5.1 demonstrate the 100% accuracy of the SVM method with every square outside of the diagonal line having a value of 0. The matrix also shows that the data between step 6 to 10 is accurately predicted even though it would have been very difficult for human to predict from part 2 of the report. Based on this performance analysis, the Support Vector Machine model is selected to predict new data point in the next part of the report.

# 2.6 Part 6: Model Evaluation

The Support Vector Classifier is packaged and saved as a Joblib format. It is then loaded in this part of the project to predict a new set of provided data. The table below shows the predicted maintenance step based on the provided data with the use of Support Vector Machine model:

**Table 2.6. 1: Predicted Maintenance Step from provided data**

|  |  |  |  |
| --- | --- | --- | --- |
| **X** | **Y** | **Z** | **Predicted Step** |
| 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 order to roughly verify this prediction, feature and target variables in this part of the report will be plotted on the same plot as the original DataFrame from part 1 that consist of 860 datapoints. The figure below visualizes the data from part 6 and part 1 on the same scatter plot:

A diagram of a diagram

Description automatically generated with medium confidence

**Figure 2.6. 1: Comparison of Predicted to Original Data**

In Figure 2.6.1, the blue 'x', orange '+' and green '.' represent the X,Y,Z data from part 1. While the red circle, purple square and brown diamond represents the X, Y, Z data from part 6. Each of the datapoint in part 6 closely matches the original datapoint from part 1, especially the Step target variable. This again showcase the incredible accuracy of the Support Vector Classifier Machine Learning model developed in this project.

# 3. Conclusion

Machine Learning is extremely beneficial in the Aerospace industry, especially for Aircraft Maintenance Engineers and the Manufacturing sector. The FlightMax Fill Motion Simulator has 13 stages of disassembly, each with variables X, Y, Z coordinates in a 3-D space. There are 860 given datapoints roughly ranging from 0 to 10 in magnitude. These coordinates are the feature variables used for the supervised classification machine learning project. X is negatively correlated to the target step variable having a Pearson coefficient of -0.75 while Y and Z aren't very correlated at 0.20 and 0.29. The dataset is split at 80-20 percent ratio and then normalized for training and testing purposes. The 3 machine ML models chosen for this project is Support Vector Machine, K-Nearest Neighbor and Random Forest Classifier. This selection is due to their accuracy and ability to be fine-tuned by the Grid Search CV method to their respective hyperparameters. Comparing the 3 models, the SVM model yields the highest performance metrics compared to K-NN and RFC. Therefore, new X, Y, Z coordinate datapoints utilize the SVM model to predict maintenance step accurately.

# Appendix

## GitHub Link

<https://github.com/GetHydrogen/Project-1.git>

## Python Code

#%% import packages

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

import seaborn as sns

import joblib

#%% Part 1: Data Processing

df = pd.read\_csv("Project 1 Data.csv")

print('Part 1: The Given Dataframe is: \n{}'.format(df))

#%% Part 2: Data Visualization

dfa = df.to\_numpy()

scatter\_X = [point[0] for point in dfa]

scatter\_Y = [point[1] for point in dfa]

scatter\_Z = [point[2] for point in dfa]

scatter\_Step = [point[3] for point in dfa]

plt.scatter(scatter\_Step,scatter\_X, label='X',marker='o',s=10)

plt.scatter(scatter\_Step,scatter\_Y, label='Y',marker='s',s=10)

plt.scatter(scatter\_Step,scatter\_Z, label='Z',marker='D',s=10)

plt.title('Part 2: Scatter Plot of X,Y,Z based on Steps')

plt.xlabel('Maintenance Steps')

plt.ylabel('Coordinates')

plt.legend()

plt.savefig('Part 2 Scatter plot',dpi=200)

plt.show()

summary\_stats = df.describe().drop(columns=['Step'])

print('Part 2: Statistical analysis of the dataset: \n{}'.format(summary\_stats))

#%% Part 3: Correlation Analysis

corr\_matrix = df.corr()

corr\_target = corr\_matrix['Step']

corr\_features = corr\_matrix.index[0:3]

plt.figure()

sns.barplot(x=corr\_features, y=corr\_target[corr\_features])

plt.title('Part 3: X,Y,Z Correlations with Maintenance Steps')

plt.savefig('Part 3 Correlation plot',dpi=200)

plt.show()

print('\nPart 3: Correlation analysis of the dataset: \n{}'.format(corr\_matrix))

#%% Part 4: Classification Model Development/Engineering

# 4.1 Stratified Random Sampling

from sklearn.model\_selection import StratifiedShuffleSplit

split = StratifiedShuffleSplit(n\_splits=1, test\_size=0.2, random\_state=42)

for train\_index, test\_index in split.split(df, df["Step"]):

strat\_train\_set = df.loc[train\_index].reset\_index(drop=True)

strat\_test\_set = df.loc[test\_index].reset\_index(drop=True)

train\_y = strat\_train\_set['Step']

test\_y = strat\_test\_set['Step']

strat\_train\_set = strat\_train\_set.drop(columns=["Step"], axis = 1)

strat\_test\_set = strat\_test\_set.drop(columns=["Step"], axis = 1)

#%% 4.2 Feature Scaling

from sklearn.preprocessing import MinMaxScaler

my\_scaler = MinMaxScaler()

scaled\_data = my\_scaler.fit\_transform(strat\_train\_set)

scaled\_data\_df = pd.DataFrame(scaled\_data)

train\_X = strat\_train\_set

test\_X = strat\_test\_set

#%% 4.3 Classfication models creation - GridSearch CV

from sklearn.model\_selection import GridSearchCV

#%% 4.3.1 Support Vector Machine - Classifier

from sklearn.svm import SVC

model1 = SVC()

param\_grid1 = {

'C': [0.1, 1, 10],

'kernel': ['linear', 'rbf','poly'],

'gamma': [0.001, 0.01, 0.1, 'scale', 'auto'],

}

grid\_search1 = GridSearchCV(model1, param\_grid1, cv=5, scoring='accuracy')

grid\_search1.fit(train\_X, train\_y)

best\_hyperparams1 = grid\_search1.best\_params\_

best\_model1 = grid\_search1.best\_estimator\_

y\_pred1 = best\_model1.predict(test\_X) #Use test feature to predict target by SVC

#%% 4.3.2 K-Nearest Neighbors Classifier

from sklearn.neighbors import KNeighborsClassifier

param\_grid2 = {'n\_neighbors': [3, 4, 5, 6, 7, 8, 9],}

model2 = KNeighborsClassifier()

grid\_search2 = GridSearchCV(model2, param\_grid2, cv=5, scoring='accuracy')

grid\_search2.fit(train\_X, train\_y)

best\_hyperparameters2 = grid\_search2.best\_params\_

best\_model2 = grid\_search2.best\_estimator\_

y\_pred2 = best\_model2.predict(test\_X) #Use test feature to predict target by KNN

#%% 4.3.3 Random Forest Classifier

from sklearn.ensemble import RandomForestClassifier

param\_grid3 = {

'n\_estimators': [10, 30, 50],

'max\_depth': [None, 10, 20, 30],

'min\_samples\_split': [2, 5, 10],

'min\_samples\_leaf': [1, 2, 4],

'max\_features': ['sqrt', 'log2']

}

model3 = RandomForestClassifier()

grid\_search3 = GridSearchCV(model3, param\_grid3, cv=5, scoring='accuracy')

grid\_search3.fit(train\_X, train\_y)

best\_hyperparameters3 = grid\_search3.best\_params\_

best\_model3 = grid\_search3.best\_estimator\_

y\_pred3 = best\_model3.predict(test\_X) #Use test feature to predict target by RFC

#%% Part 5: Model Performance Analysis

from sklearn.metrics import confusion\_matrix, accuracy\_score, precision\_score, f1\_score

#The Following matrix shows the F1 Score, Precision and Accuracy of the 3 models

classifier\_perf = {

'SVC': [f1\_score(test\_y, y\_pred1, average='macro'), precision\_score(test\_y, y\_pred1, average='macro'), accuracy\_score(test\_y, y\_pred1)],

'K-NN': [f1\_score(test\_y, y\_pred2, average='macro'), precision\_score(test\_y, y\_pred2, average='macro'), accuracy\_score(test\_y, y\_pred2)],

'RFC': [f1\_score(test\_y, y\_pred3, average='macro'), precision\_score(test\_y, y\_pred3, average='macro'), accuracy\_score(test\_y, y\_pred3)]

}

classifier\_perf = pd.DataFrame(classifier\_perf,['F1 Score','Precision','Accuracy'])

print('\nThe performance of the 3 Machine Learning models: \n{}'.format(classifier\_perf))

print('=>Support Vector Classifier has the highest F1 Score, Precision and Accuracy\n')

cm1 = confusion\_matrix(test\_y, y\_pred3)

plt.figure()

sns.heatmap(cm1)

plt.xlabel('Predicted Step from SVM')

plt.ylabel('Test Set')

plt.title('Part 5: Confusion Matrix of Random Forest Classifier')

plt.savefig('Part 5 Confusion Matrix',dpi=200)

plt.show()

#%% Part 6: Model Evaluation

joblib.dump(grid\_search1, 'Duong\_SVM.joblib')

loaded\_model = joblib.load('Duong\_SVM.joblib')

p6\_data = [[9.375,3.0625,1.51],

[6.995,5.125,0.3875],

[0,3.0625,1.93],

[9.4,3,1.8],

[9.4,3,1.3]]

p6\_df = pd.DataFrame(p6\_data, columns=['X', 'Y', 'Z'])

p6\_pred = loaded\_model.predict(p6\_df)

p6\_pred = pd.DataFrame(p6\_pred, columns=['Step'])

p6\_join = p6\_df.join(p6\_pred)

print('Part 6: The SVC model prediction for provided data set is: \n{}'.format(p6\_join))

plt.scatter(df['Step'],df['X'], label='X\_p1',marker='x',s=10)

plt.scatter(df['Step'],df['Y'], label='X\_p1',marker='+',s=20)

plt.scatter(df['Step'],df['Z'], label='X\_p1',marker='.',s=10)

plt.scatter(p6\_join['Step'],p6\_join['X'], label='X\_p6',marker='o',s=60)

plt.scatter(p6\_join['Step'],p6\_join['Y'], label='Y\_p6',marker='s',s=60)

plt.scatter(p6\_join['Step'],p6\_join['Z'], label='Z\_p6',marker='D',s=60)

plt.title('Part 6: Comparison of Predicted to Original Data')

plt.xlabel('Maintenance Steps')

plt.ylabel('Coordinates')

plt.legend()

plt.savefig('Part 6 Scatter compare',dpi=200)

plt.show()