


Semester (Term, Year)	
Course Code	
Course Section	
Course Title	
Course Instructor	
Submission	
Submission No.	
Submission Due Date	
Title	
Submission Date	

Submission by (Name):	Student ID (XXXX1234)	Signature
		

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/

Tables of Content

COVER PAGE	1
TABLES OF CONTENT	2
LIST OF TABLES AND FIGURES.....	2
1. BACKGROUND & INTRODUCTION.....	3
2. PROJECT OUTLINE	3
2.1 PART 1: DATA PROCESSING	4
2.2 PART 2: DATA VISUALIZATION.....	4
2.3 PART 3: CORRELATION ANALYSIS.....	6
2.4 PART 4: CLASSIFICATION MODEL DEVELOPMENT/ENGINEERING	7
2.5 PART 5: MODEL PERFORMANCE ANALYSIS	7
2.6 PART 6: MODEL EVALUATION	8
3. CONCLUSION	10
APPENDIX.....	11
GITHUB LINK.....	11
PYTHON CODE	11

List of Tables and Figures

Figure 2.0. 1: Inverter of the FlightMax Fill Motion Simulator.....	3
Figure 2.1. 1: Python console output for DataFrame from .csv file	4
Table 2.2. 1: Basic statistical analysis of raw data	4
Figure 2.2. 1: Scatter plot of X, Y, Z Coordinate based on Step	5
Figure 2.3. 1: Correlation plot between feature and target variables	6
Table 2.5. 1: Performance Analysis of the 3 chosen ML models	7
Figure 2.5. 1: Confusion Matrix of the Support Vector Classifier model	8
Table 2.6. 1: Predicted Maintenance Step from provided data	8
Figure 2.6. 1: Comparison of Predicted to Original Data	9

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:



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.

```
Part 1: The Given Dataframe is:
      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
..    ...    ...    ...    ...
855  0.000  3.0625  1.89    13
856  0.000  3.0625  1.90    13
857  0.000  3.0625  1.91    13
858  0.000  3.0625  1.92    13
859  0.000  3.0625  1.93    13

[860 rows x 4 columns]
```

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:

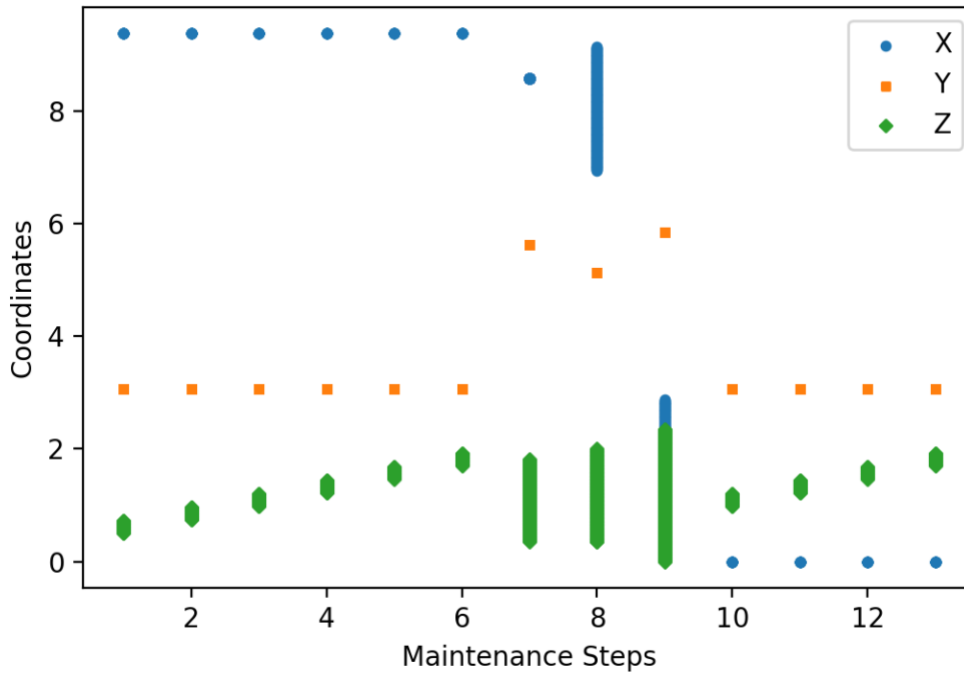


Figure 2.2. 1: Scatter plot of X, Y, Z Coordinate based on Step

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 around 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:

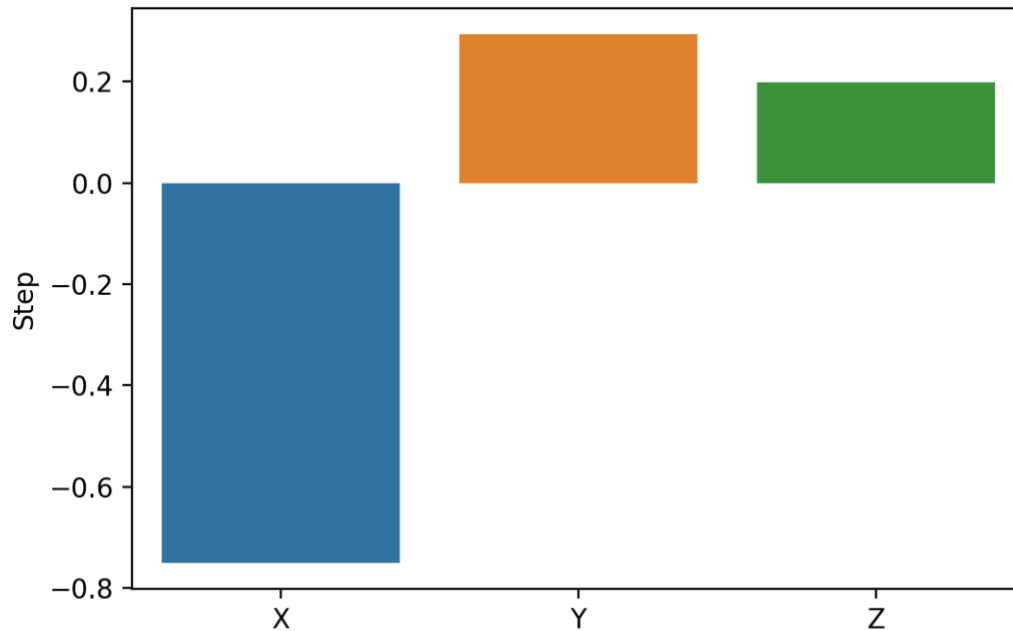


Figure 2.3. 1: Correlation plot between feature and target variables

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:

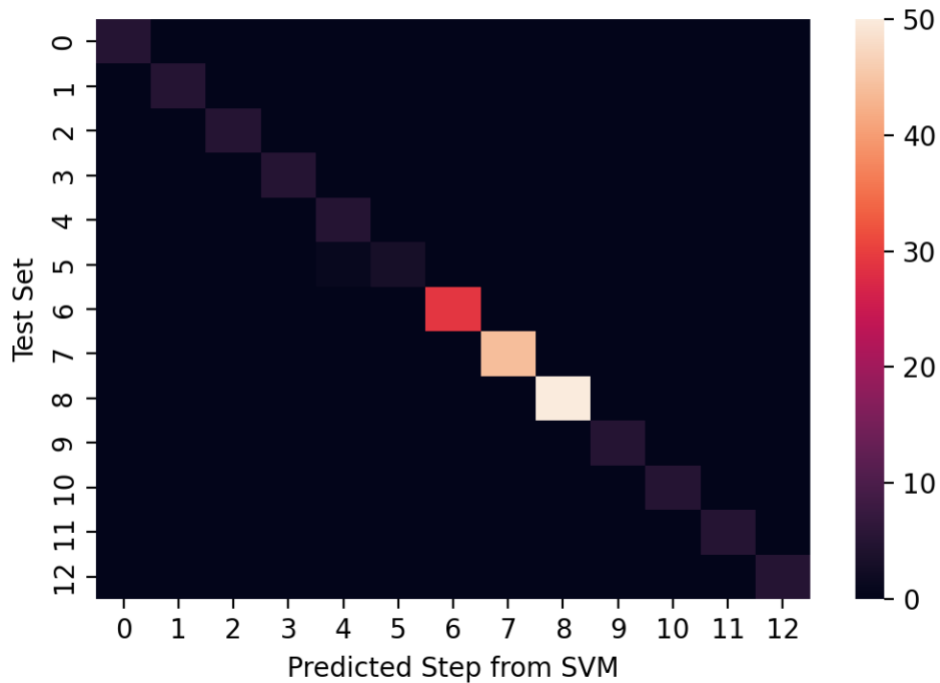


Figure 2.5. 1: Confusion Matrix of the Support Vector Classifier model

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:

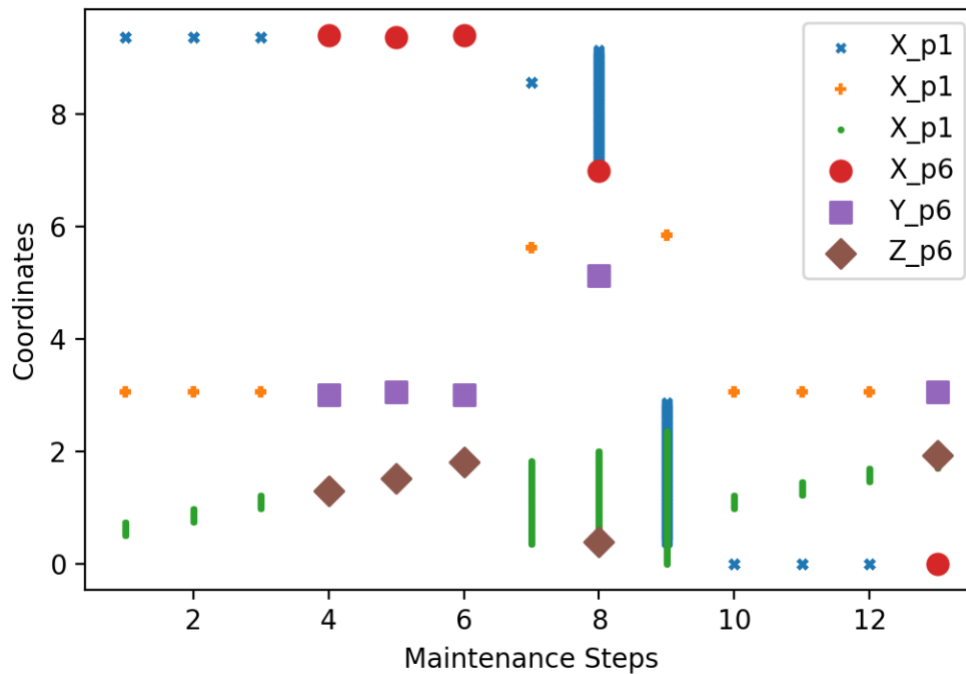


Figure 2.6. 1: Comparison of Predicted to Original Data

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 Classification 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()

```