Machine Learning Engineer Nano Degree – Capstone Project

CNC Mill Tool Wear Predictive Analysis and Building Closed-Loop Control System

Saravanan Balasubramanian
Nov 3, 2018

Definition

Project Overview

Large industrial/manufacturing facilities operate multiple equipment with thousands of sensors/variables driving the manufacturing process. The equipment used in industrial/manufacturing process are subject to wear and tear. The tool wear directly affects the quality of the yield and introduces unscheduled production shut-down in shop floors leading to economic impacts to business.

Milling is a cutting process that uses cutter to remove material from the surface of a workpiece. Computer Numerical Control (CNC) milling machines are used widely in the milling process. The cutters in the CNC machines are subject to gradual failure due to regular operation. This project aims to reduce economic impacts to business caused by tool wear with CNC Milling machines.

Problem Statement

For this project, I am trying to get data from CNC machines/Historian and predict tool wear so that Operations personnel could see the trend and take preventive actions. Predicting CNC machine tool wear is a binary classification problem. Multiple machine learning algorithms can be applied for this binary classification problem and they have quantitative metrics such as accuracy, precision, f1 score etc. to measure the results.

The CNC Milling Dataset is obtained from Kaggle donated by Sharon Sun. This data set was generated by running series of experiments on CNC milling machine in the System-level Manufacturing and Automation Research Testbed (SMART) at the University of Michigan.

Dataset link : https://www.kaggle.com/shasun/tool-wear-detection-in-cnc-mill/home

Solution Description

As identifying the CNC machine wear is a classification problem, multiple classification machine learning algorithms will be trained with this dataset to identify the final model which generalizes the data set well.

The data will be consolidated first and visually explored to identify relevant features and drop features that are not required. Once the features are selected, the data set will be split to training set and test set.

Identified classification algorithms will be trained to pick the right model for the data set. The models will be validated using the metrics defined in the following section.

Metrics

The models accuracy will be evaluated using F1 score (derived by precision & recall using a confusion matrix).

Confusion Matrix	Actual Positive	Actual Negative
Predicted Positive	TP	FP
Predicted Negative	FN	TN

F1 score considers both precision and recall of the test to compute the score – to be precise, F1 score is a harmonic average of the precision and recall. F1 score range from 0 to 1. It reaches 1 for perfect precision and recall and worst at 0.

Analysis

Data Exploration

As part of the data set from Kaggle, we are given with data from eighteen experiments in individual csv files (experiment1.csv to experiment18.csv). The train.csv file contain the meta data on the tool condition.

Code for consolidating the experiment CSV files and adding target variable to the dataset is available in “Dataset Consolidation” section of the notebook (cnc_mill.ipynb).

The features available in the machining datasets are:

- X1_ActualPosition: actual x position of part (mm)

- X1_ActualVelocity: actual x velocity of part (mm/s)
- X1_ActualAcceleration: actual x acceleration of part (mm/s/s)
- X1_CommandPosition: reference x position of part (mm)
- X1_CommandVelocity: reference x velocity of part (mm/s)
- X1_CommandAcceleration: reference x acceleration of part (mm/s/s)
- X1_CurrentFeedback: current (A)
- X1_DCBusVoltage: voltage (V)
- X1_OutputCurrent: current (A)
- X1_OutputVoltage: voltage (V)
- X1_OutputPower: power (kW)
- Y1_ActualPosition: actual y position of part (mm)
- Y1_ActualVelocity: actual y velocity of part (mm/s)
- Y1_ActualAcceleration: actual y acceleration of part (mm/s/s)
- Y1_CommandPosition: reference y position of part (mm)
- Y1_CommandVelocity: reference y velocity of part (mm/s)
- Y1_CommandAcceleration: reference y acceleration of part (mm/s/s)
- Y1_CurrentFeedback: current (A)
- Y1_DCBusVoltage: voltage (V)
- Y1_OutputCurrent: current (A)
- Y1_OutputVoltage: voltage (V)
- Y1_OutputPower: power (kW)
- Z1_ActualPosition: actual z position of part (mm)
- Z1_ActualVelocity: actual z velocity of part (mm/s)
- Z1_ActualAcceleration: actual z acceleration of part (mm/s/s)
- Z1_CommandPosition: reference z position of part (mm)
- Z1_CommandVelocity: reference z velocity of part (mm/s)
- Z1_CommandAcceleration: reference z acceleration of part (mm/s/s)
- Z1_CurrentFeedback: current (A)
- Z1_DCBusVoltage: voltage (V)
- Z1_OutputCurrent: current (A)
- Z1_OutputVoltage: voltage (V)
- S1_ActualPosition: actual position of spindle (mm)
- S1_ActualVelocity: actual velocity of spindle (mm/s)
- S1_ActualAcceleration: actual acceleration of spindle (mm/s/s)
- S1_CommandPosition: reference position of spindle (mm)
- S1_CommandVelocity: reference velocity of spindle (mm/s)
- S1_CommandAcceleration: reference acceleration of spindle (mm/s/s)
- S1_CurrentFeedback: current (A)
- S1_DCBusVoltage: voltage (V)
- S1_OutputCurrent: current (A)
- S1_OutputVoltage: voltage (V)
- S1_OutputPower: current (A)

- S1_SystemInertia: torque inertia (kg*m^2)
- M1_CURRENT_PROGRAM_NUMBER: number the program is listed under on the CNC
- M1_sequence_number: line of G-code being executed
- M1_CURRENT_FEEDRATE: instantaneous feed rate of spindle

The data set contains 25286 records to train with.

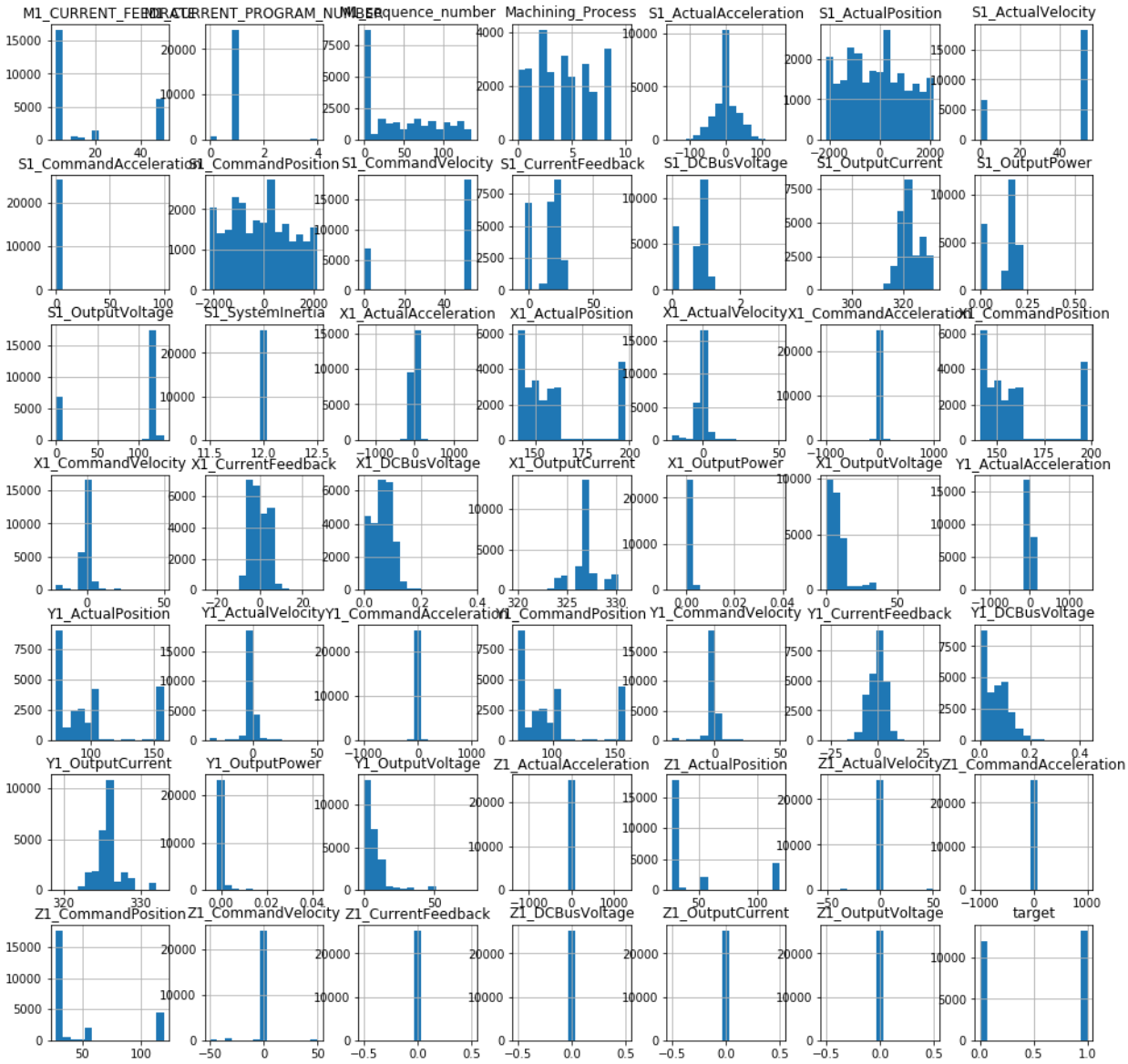
	X1_ActualPosition	X1_ActualVelocity	X1_ActualAcceleration	X1_CommandPosition	X1_CommandVelocity	X1_CommandAcceleration	X1_CurrentFeedback
count	25286.000000	25286.000000	25286.000000	25286.000000	25286.000000	25286.000000	25286.000000
mean	159.052045	-0.288657	0.094264	159.050700	-0.283076	0.253215	-0.469714
std	19.330873	5.658260	93.877623	19.331144	5.664309	72.594951	4.220750
min	141.000000	-20.400000	-1280.000000	141.000000	-20.000000	-1000.000000	-23.400000
25%	145.000000	-2.050000	-31.300000	145.000000	-2.050000	0.000000	-3.930000
50%	153.000000	0.000000	0.000000	153.000000	0.000000	0.000000	-0.666000
75%	162.000000	0.200000	25.000000	162.000000	0.000000	0.000000	3.140000
max	198.000000	50.700000	1440.000000	198.000000	50.000000	1000.000000	27.100000

	Y1_ActualPosition	Y1_ActualVelocity	Y1_ActualAcceleration	Y1_CommandPosition	Y1_CommandVelocity	Y1_CommandAcceleration	Y1_CurrentFeedback
count	25286.000000	25286.000000	25286.000000	25286.000000	25286.000000	25286.000000	25286.000000
mean	99.230064	-0.422932	0.928832	99.226271	-0.412075	1.484158	-0.061681
std	29.244880	6.006439	85.074580	29.242802	6.004721	81.358073	4.469548
min	72.400000	-32.800000	-1260.000000	72.400000	-32.400000	-1000.000000	-27.800000
25%	77.500000	-0.075000	-18.800000	77.500000	0.000000	0.000000	-3.090000
50%	90.000000	0.000000	0.000000	90.000000	0.000000	0.000000	0.146000
75%	105.000000	0.100000	18.800000	105.000000	0.000000	0.000000	2.900000
max	158.000000	50.400000	1460.000000	158.000000	50.000000	1000.000000	30.700000

	Z1_ActualPosition	Z1_ActualVelocity	Z1_ActualAcceleration	Z1_CommandPosition	Z1_CommandVelocity	Z1_CommandAcceleration	Z1_CurrentFeedback
count	25286.000000	25286.000000	25286.000000	25286.000000	25286.000000	25286.000000	25286.0
mean	47.780638	-0.328759	-0.103658	47.778031	-0.328464	0.134118	0.0
std	34.255650	7.635223	66.442671	34.252517	7.637045	61.988683	0.0
min	27.500000	-51.500000	-1260.000000	27.500000	-50.000000	-1000.000000	0.0
25%	28.500000	0.000000	-6.250000	28.500000	0.000000	0.000000	0.0
50%	29.500000	0.000000	0.000000	29.500000	0.000000	0.000000	0.0
75%	55.500000	0.000000	6.250000	55.500000	0.000000	0.000000	0.0
max	119.000000	50.900000	1270.000000	119.000000	50.000000	1000.000000	0.0

Exploratory Visualization

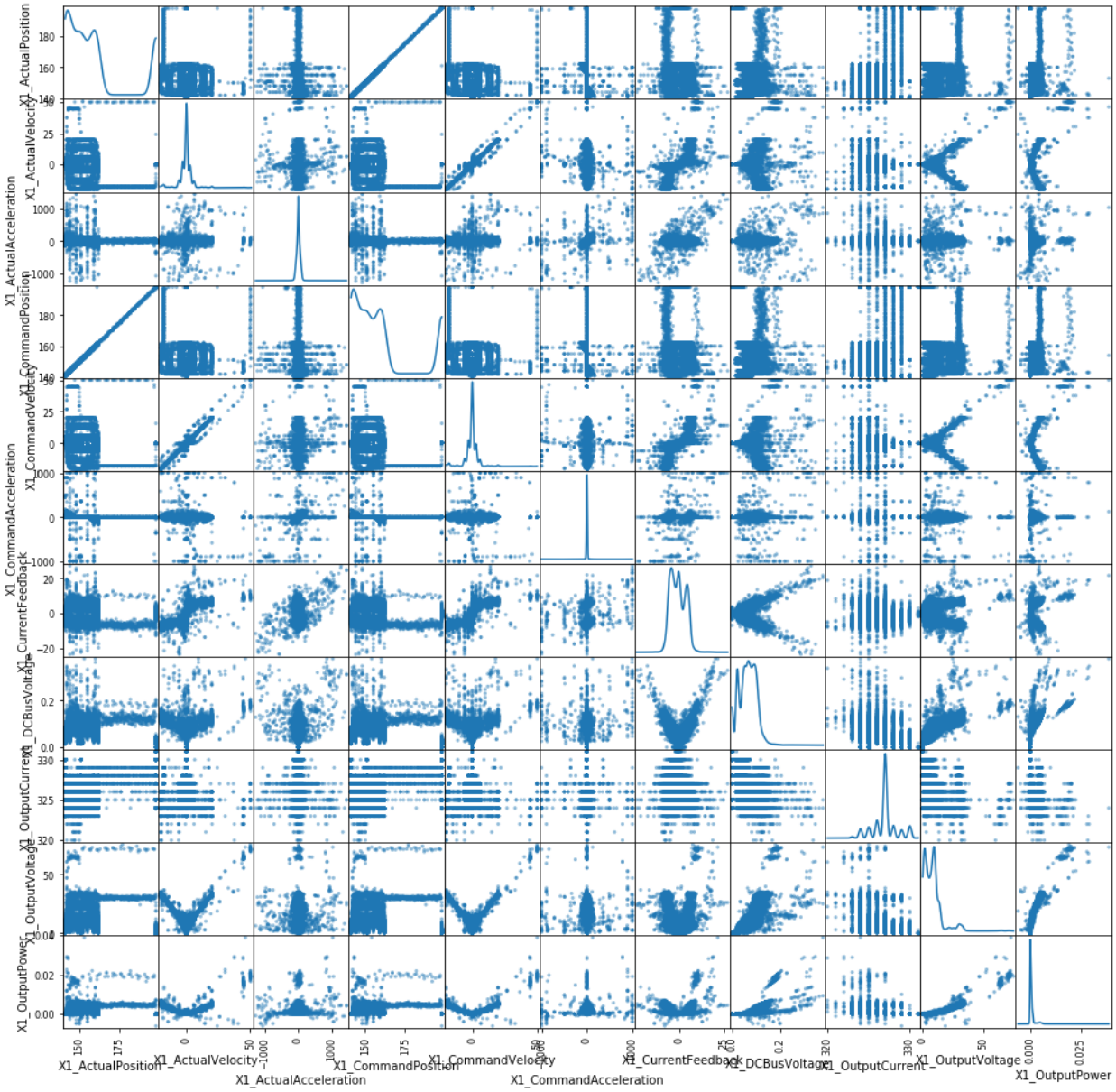
Once I consolidated all the experimental CSV files into one data set and obtained some basic insights about the features, I tried to visualize all the variables as below.

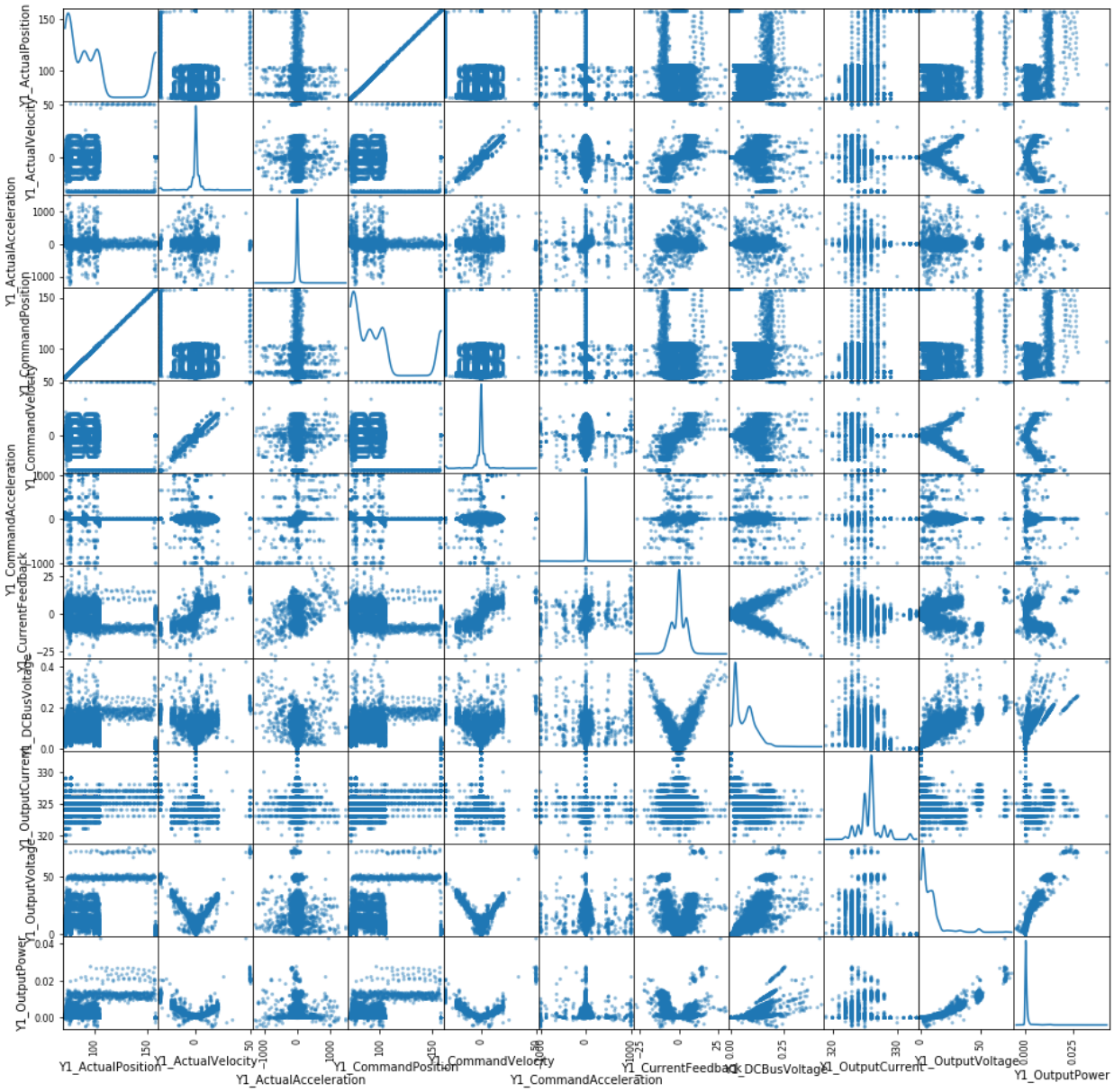


From the histograms, we could notice that the following features don't have any meaningful data to inform the model and can be dropped from the data set.

- Z1_CurrentFeedback
- Z1_DCBusVoltage
- Z1_OutputCurrent
- Z1_OutputVoltage

Scatter plotting each variable against another variable provides some insight on most obvious linear relationships.

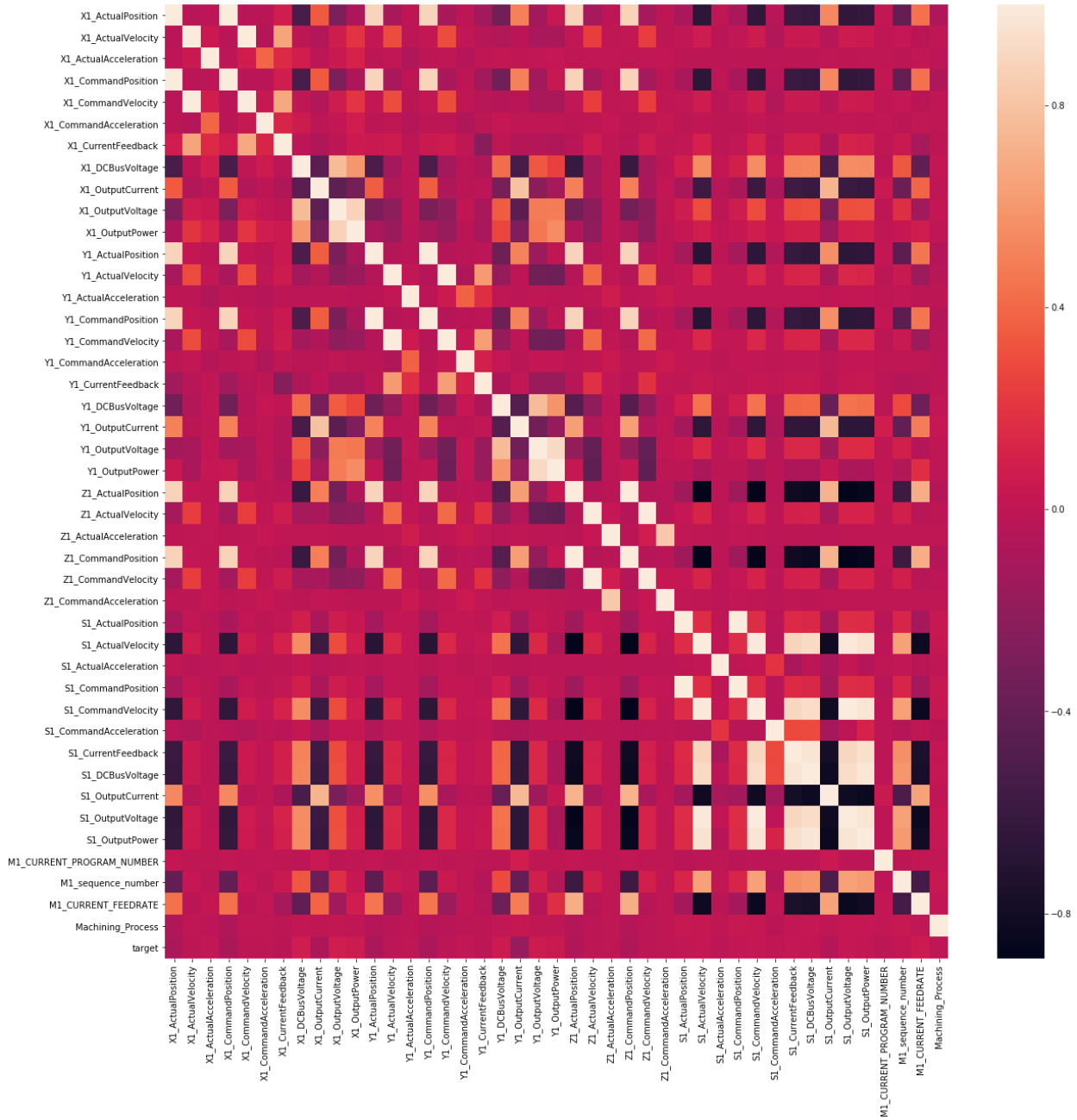




Linear relationship between the following variables are obvious from the scatter plots:

- Y1_ActualPosition vs Y1_CommandPosition
- X1_ActualPosition vs X1_CommandPosition

The following visualization tells that there are stronger correlation between variables:



Algorithms and Techniques

As there are multiple classifiers that can be applied to this problem and since we don't know which one a good fit for this problem would be, I will evaluate the following algorithms.

Logistic Regression (LR)

Logistic regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function" (Wikipedia)

Logistic Regression is more like a linear classifier which uses the calculated logits (score) to predict the target class. As this is binary classification problem, Logistic Regression would be a good fit and hence marked as the benchmark model.

Decision Trees

Decision Trees are basically rule sets that indicate how to best split the data into groups recursively. The best split consists of picking the attribute that has the most information gain or the attribute that most reduces the entropy in the newly split groups.

Being a tree-based algorithm, Decision Trees can handle features with bad scales reducing the amount of pre-processing to be done. They also can learn non-linear relationships as well as the relationships between features.

Support Vector Machines (SVM)

Support Vector Machines are based on the maximum margin algorithm. SVMs can be best explained for classification but is also valid for regression. A linear SVM for binary classification is a linear separator that will find a "maximum-margin" line that is in the center of the two different classes.

Gaussian Naïve Bayes (NB)

Naïve Bayes classifiers are probabilistic classifiers which are based on applying Bayes Theorem with independence assumptions between the features. (Wikipedia)

Naïve Bayes classifiers are highly scalable and operate by identifying the probability. Computing conditional probability between categorical features would be easier where as it become compute intensive for continuous variables. The Gaussian Naïve Bayes works with a typical assumption that the continuous values associated with each class are distributed according to a Gaussian distribution. The algorithm first segments the data based on mean and variance in each class and then computing probability for each of the classes.

Random Forests (RF)

Random Forests are an ensemble learning method for classification and regression tasks, that operate by constructing multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean predition (regression) of the individual trees. (Wikipedia)

Random Forest is a flexible algorithm that yields great result most of the time even without hyper-tuning. Random Forest build multiple decision trees and merges them together to get a more accurate and stable prediction.

The accuracy of the algorithm will be measured using the metrics outlined in Metrics section.

Benchmark

For this binary classification problem, the benchmark model will be Logistic Regression on unscaled data.

The Logistic Regression model has these observations:

- Accuracy: 0.5819
- F1 Score: 0.6218
- Area under ROC curve: 0.5777
- Training time (seconds) : 12.098269

Methodology

Data Preprocessing

The following features didn't have meaningful data in the data set and hence had to be removed before training.

- Z1_CurrentFeedback
- Z1_DCBusVoltage
- Z1_OutputCurrent
- Z1_OutputVoltage

The data set is split to training set (80%) and test set (20%). After splitting the training and test sets,

- 20228 rows of data to train the model
- 5058 rows of data to test the model

Implementation

With the training and test sets split, the training set was trained with the identified models in Sec. Algorithms and Techniques.The models were validated with Accuracy and F1 Score metrics. Confusion matrix for each model was also printed to see True Negative, False Positive, False Negative and True Positive.

For each chosen algorithm, I created the classifier, trained it on the training set and validated the accuracy using the test set.

Logistic Regression Classifier

```
from datetime import datetime

lm = LogisticRegression()
t1 = datetime.now()
lm.fit(x_train, y_train)
t2 = datetime.now()
delta = t2 - t1

lm_pred = lm.predict(x_test)

print("Trained on {0} observations and scoring with {1} test samples.".format(len(x_train), len(x_test)))
print("Accuracy: {0:0.4f}".format(accuracy_score(y_test, lm_pred)))
print("F1 Score: {0:0.4f}".format(f1_score(y_test, lm_pred)))
print("Area under ROC curve: {0:0.4f}".format(roc_auc_score(y_test, lm_pred)))
print("training time (seconds) : ", delta.seconds + delta.microseconds/1E6)
```

Following are the results for various untuned models on unscaled data.

Model	Accuracy	F1 Score	Area under ROC curve	Training Time (seconds)
Logistic Regression	0.5757	0.6067	0.5730	8.4
Decision Tree	0.9871	**0.9879**	0.9872	0.3
SVM	0.5708	0.5950	0.5695	30474
Gaussian Naïve Bayes	0.5579	0.6733	0.5374	0.03
Random Forest	0.9881	**0.9889**	0.9881	0.43

Refinement

As this is a binary classification problem, initially I thought to consider Logistic Regression as the benchmark model and wanted to get a model better than that. From the previous section, it is visible that both Decision Tree and Random Forests did lot better than Logistic Regression on the unscaled data and without tuning the hyper parameters.

The Random Forests model provided an F1 score of 0.9881. So I decided to refine it further by tuning the hyper parameters to get a higher F1 score. Following are the details of the refinement with results observed.

Tuned hyper-parameters:

Hyper-parameter	Untuned	Tuned
max_depth The maximum depth of the tree	None	40
min_samples_leaf The minimum number of samples required to be at the leaf node	1	4
n_estimators The number of trees in the forest	10	200

After tuning the above parameters, the F1 score on test set was improved to 0.9931. I could have used GridSearchCV to find the optimal hyper-parameters, but I got luck with tuning few parameters to improve the score.

Results

Model Evaluation and Validation

The tuned model provided better F1 score and Accuracy than the untuned model. It did lot better than the logistic regression model.

Model	F1 score	Accuracy	Training Time (secs)
Logistic Regression	0.6067	0.5757	8.4
Random Forest (untuned)	0.9889	0.9881	0.4
Random Forest (tuned)	**0.9931**	**0.9927**	**8.1**

Decision Trees and Random Forests are prone to overfit the model. To overcome overfitting by the final Random Forest model, the hyper parameter “n_estimators” is set to 200, which directs the algorithm to build the number of trees before taking the maximum voting or taking average of the predictions.

Given the large number of trees, the classifier won’t overfit the model as there are enough trees in the forest. More the number of trees in the forest, the training performance will be degraded.

In this case, the untuned Random Forest was trained in 0.4 seconds. The tuned Random Forest model took 8.1 seconds to train. Though the training time increased significantly, this is still reasonable for the problem.

Justification

Based on the improvements recorded above, the final tuned Random Forest model can be deemed as satisfactorily generalizing the given data set and can be used to predict.

Conclusion

Free-Form Visualization

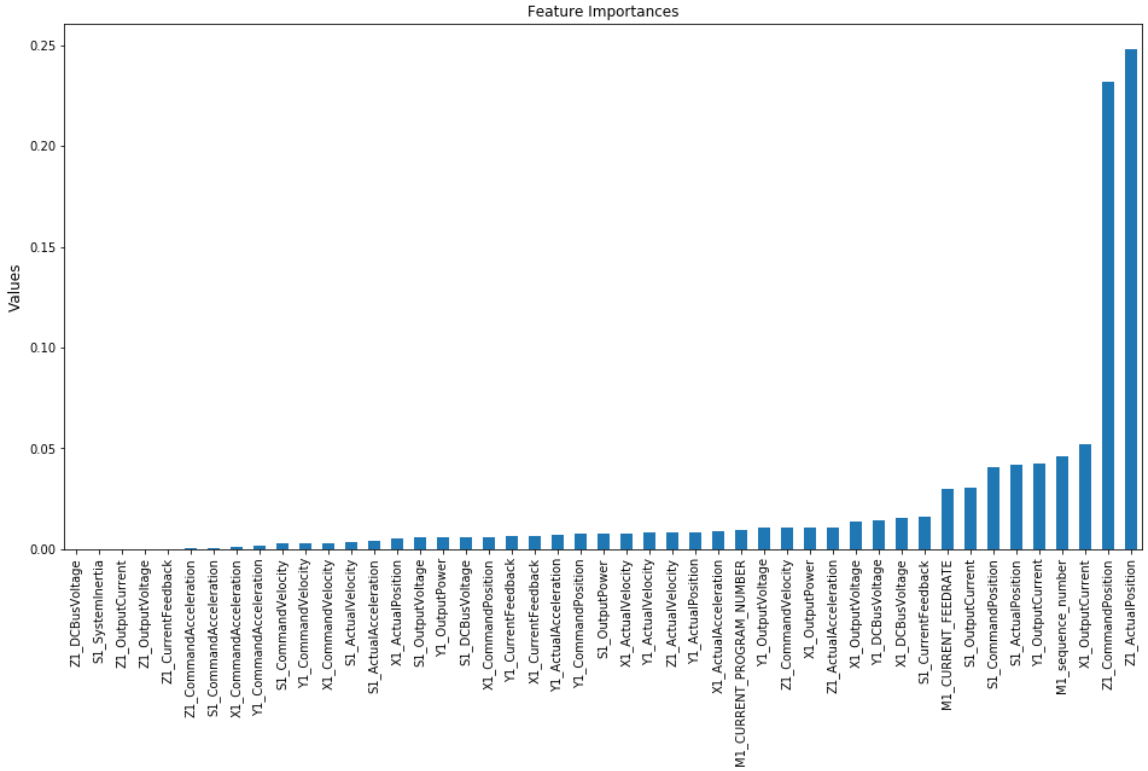
The feature importance per the tuned model is as follows:

```
Most important feature = Z1_ActualPosition
Least important feature = Z1_CurrentFeedback

Top 5 most important features:-
Z1_ActualPosition
Z1_CommandPosition
X1_OutputCurrent
M1_sequence_number
Y1_OutputCurrent

Top 5 least important features:-
Z1_DCBusVoltage
S1_SystemInertia
Z1_OutputCurrent
Z1_OutputVoltage
Z1_CurrentFeedback
```

Feature importance are visually represented as below:



Z1_AcutalPosition and Z1_CommandPosition features affects the tool condition significantly than any other features.

Reflection

Having been able to work on totally unknown dataset and build a model with good accuracy helped me to understand the potential of machine learning algorithms and their possible applications.

I would summarize the project as follows:

a. Identify the problem and gather right data to figure out the pattern; determine the right problem space : classification vs regression
b. Load the data set, visualize and explore the raw data to identify early patterns, unwanted features etc.
c. Preprocess the data (categorical variables) and select relevant features to train the models
d. Identify the algorithms to train the data with and metrics to validate the model
e. Apply the identified algorithms and visualize the results
f. Choose the best algorithm and refine it by tuning the hyper-parameters

Improvement

The solution could further be improved in the following ways:

a. Being a binary classification problem, I was expecting Logistic Regression to provide good score but ended up with a low accuracy and F1 score. Not only Logistic Regression, all algorithms (SVM, Naïve Bayes) other than tree based models (Decision Tree, Random Forest) provided similar low scores. Those non-tree based models could be fine-tuned with hyper parameter to identify why they under-fit the data.

b. The feature importance chart shows very few features such as Z1_CommandPosition, Z1_ActualPosition dominate the relationship with the target feature. Better feature engineering would yield better results for all the algorithms.

c. For the identified Random Forest model, Grid Search could have been used to tune the hyper parameters and identify the right parameters instead of taking a manual approach.

Conclusion

For the CNC mill wear detection problem, the final tuned Random Forest model is recommended as a solution. This model could be deployed real time in production where the data from sensors could be fed near real-time to determine the tool condition trend. With the incoming data directly from the sensors, the model could make prediction near real-time (within minutes) and feedback the production engineering personnel/systems on the tool condition trend to take preventive actions (closed loop feedback control systems).