Steel Fault Type Diagnosis Project

Background

As is known to all, steel is a very important material, which is widely used in various industries, such as construction industry and mechanical industry, to build skyscrapers and make cars. Steel plate is one of the end use forms of steel, the car industry consume steel plate by stamping it into car skins of different appearance.

Steel plants, manufacturers of steel plates, produce steel plate using a production line with several rolling mills, which will roll the thick steel plate into a thinner one, just like how we roll dough into dumpling skins with a rolling pin.

They need to be very careful during the production, since there are several types of faults might happen in the process. Since the mill roll is also made of steel, a much harder steel, it tend to cause some faults on the steel plate, such as scratches and bumps. Besides, water are sprayed while rolling, which also brings in some faults on plate surface, such as stains and dirtiness.

The faults are not avoidable, the engineers are trying their best to mitigate the faults, one way is to get feedback while producing and adjust producing parameters, that they are detecting the occurrence of the faults, and adjust the parameters of the production line based on the different fault types. The faster a fault type is diagnosed, the quicker the adjustment could be made. Therefore, a Fault Diagnosis Service which could quickly diagnose the type of a fault quickly will be very helpful to enhance the quality of the manufacturing and reduce the cost of product testing.

Speaking of Fault Diagnosis, it is a pattern recognition problem, which could be treated it as a classification problem in machine learning, that is why Data Science is a good tool to tackle this problem.

Target

In this project, I got from UCI (University of California at Irvine) machine learning repository. The target is to train on the dataset and build a model with best params that could been used to make predictions with fresh new steel plate fault observations.

Then the product test department could leverage this service to quickly diagnose the type of a fault and help the engineers to improve steel plate's quality.

Content

I go through the following steps to get the project done:

- Firstly, prepare the data by loading the data into the notebook as data frame.
- Secondly, understand the data by peeking the data and drawing plots, diagrams with the data.
- Thirdly, preprocessing the data to make the data ready to model training.
- Fourthly, classification with different classifiers, and dig in the one which perform the best by tweaking its parameters and finally find out a best model.
- Finally, create a callable micro-service for others to better leverage my effort.

1. Prepare the data

Read in the data into the notebook as data frame.

2. Understand the data

From the shape of the data, we know that there are 1941 instances.

```
In [8]: # get the shape of the data file
        print(data.shape)

        (1941, 34)
```

This is the column names, there are 34 columns in total, among which 27 are features and the rest 7 are Fault type, which is the Class in classification.

```
In [7]: print(data.columns)

        Index(['X_Minimum', 'X_Maximum', 'Y_Minimum', 'Y_Maximum', 'Pixels_Areas',
               'X_Perimeter', 'Y_Perimeter', 'Sum_of_Luminosity',
               'Minimum_of_Luminosity', 'Maximum_of_Luminosity', 'Length_of_Conveyer',
               'TypeOfSteel_A300', 'TypeOfSteel_A400', 'Steel_Plate_Thickness',
               'Edges_Index', 'Empty_Index', 'Square_Index', 'Outside_X_Index',
               'Edges_X_Index', 'Edges_Y_Index', 'Outside_Global_Index', 'LogOfAreas',
               'Log_X_Index', 'Log_Y_Index', 'Orientation_Index', 'Luminosity_Index',
               'SigmoidOfAreas', 'Pastry', 'Z_Scratch', 'K_Scatch', 'Stains',
               'Dirtiness', 'Bumps', 'Other_Faults'],
              dtype='object')
```

Here is a glimpse of the data, we could see that all of the values are in numeric format, to test this, we could print out its info, all the values are either int or float.

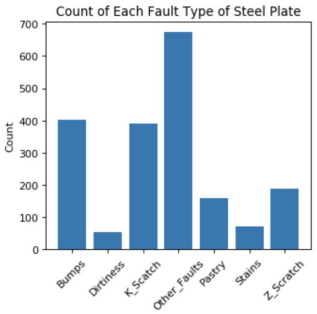
```
In [9]: data.head(5)
```

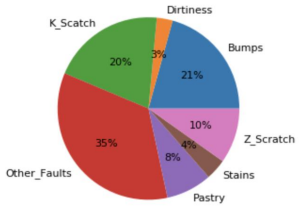
Out[9]:

	X_Minimum	X_Maximum	Y_Minimum	Y_Maximum	Pixels_Areas	X_Perimeter	Y_Perimeter	Sum_of_Luminosity	Minimum_of_Luminosity	Maximum_of_Lumir
0	42	50	270900	270944	267	17	44	24220	76	
1	645	651	2538079	2538108	108	10	30	11397	84	
2	829	835	1553913	1553931	71	8	19	7972	99	
3	853	860	369370	369415	176	13	45	18996	99	
4	1289	1306	498078	498335	2409	60	260	246930	37	

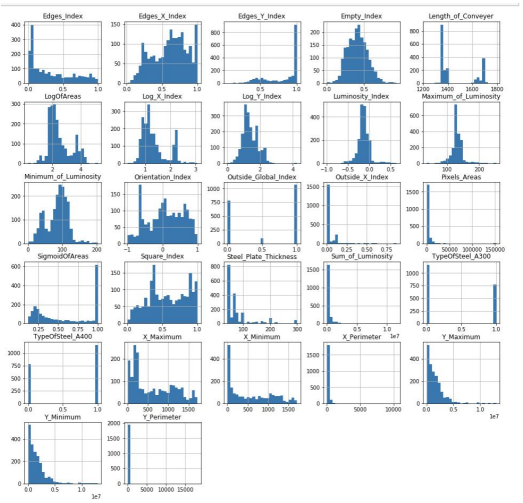
5 rows × 34 columns

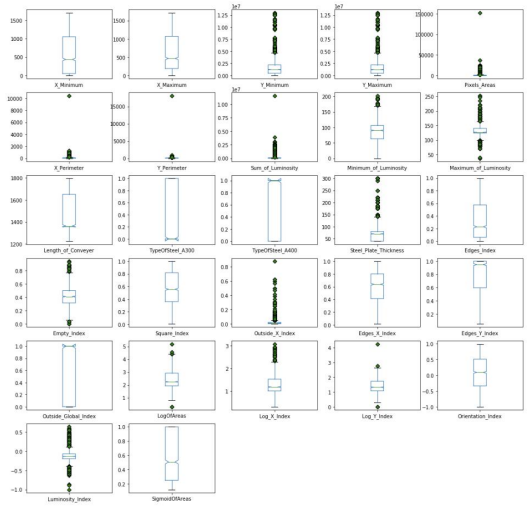
Next I merge the 7 classes into one column, and here is the bar plot and pie chart of distribution of the classes. As we could see that the target classes are not evenly distributed.



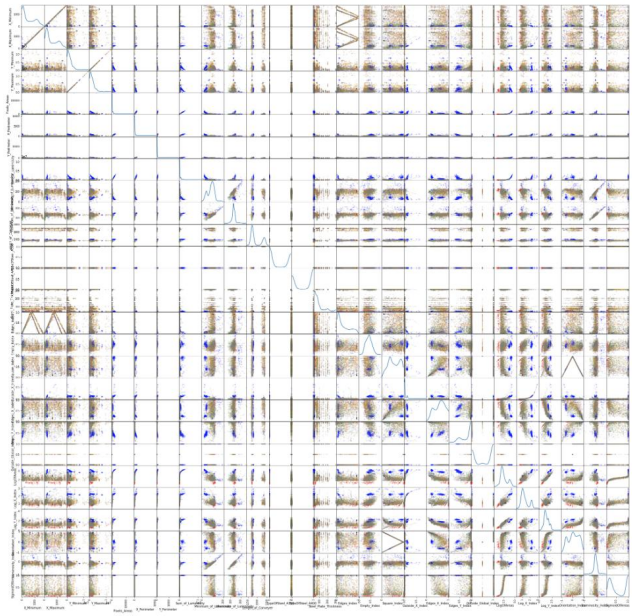


Then, I use the hist diagram and box plot to get the distribution the features. We could see that most of the columns have small values, while some of them have values in million level. And several features have outliers, for which I will deal with them in the preprocessing step.



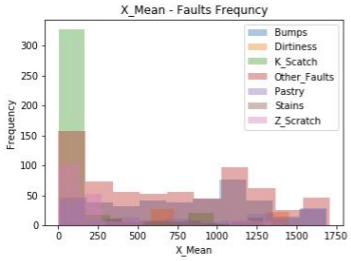


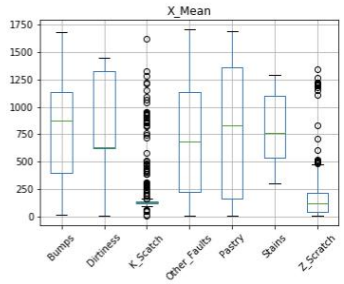
Then, the scatter matrix shows that there are some patterns between features and the class.



To dig deep, I research on the relationship between each feature and the class. I got some interesting findings.

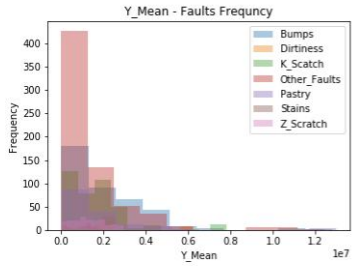
1) Here is the how the location related to the classes. Below is X, which is the width direction of a steel plate.

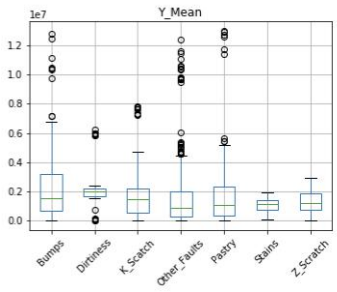




We could see that there are a big potion of faults locate in the range between [0, 200]. We could name that most of faults here are K_Scatch, Z_Scatch, and I also calculate the ratio of faults happens in this range, it accounts for 37% of all faults.

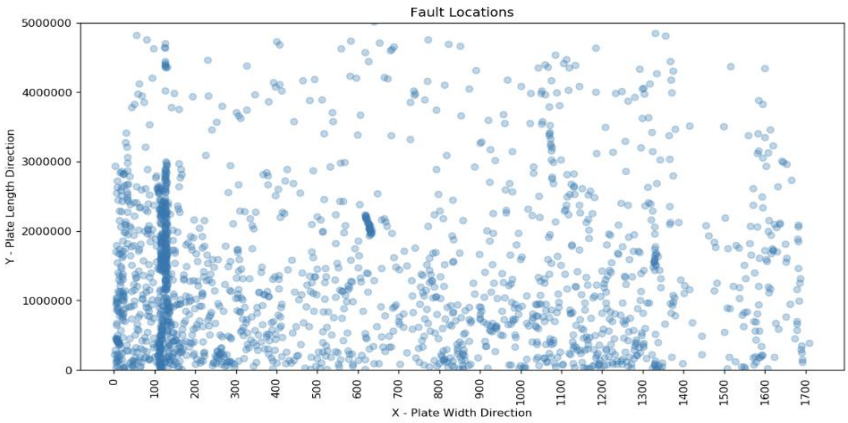
Similarly, there are also pattern between Y and the fault types.





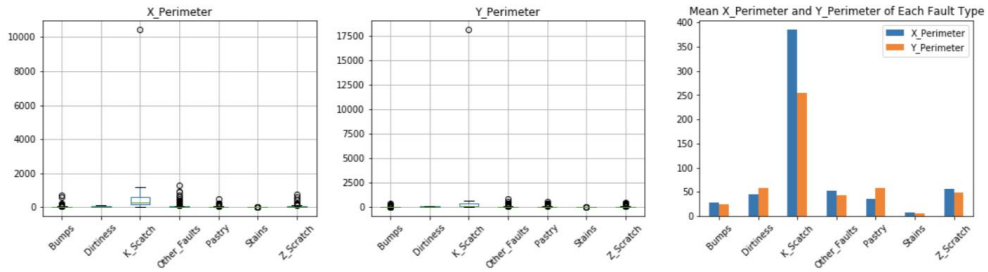
We could see that more than 86% of faults happen before 0.3 * 1e7, and more than 96% happen before 0.5 * 1e7. The length of 0.5 * 1e7 accounts for more than 38% of the total Y range, which means the faults happen mostly in the first half of the steel plate.

This X - Y diagram give us a whole picture of the locations all the faults occur.



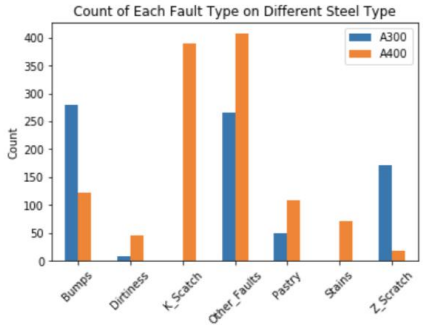
We could easily tell that a lot of faults form a linear strip region that near the X with value of 110 through all the length of the plate. There should be some special reasons why fault tend to occurs in this region, let's leave this to the engineers in the steel plate.

2) Here is how the X_Perimeter and Y_Perimeter related to the classes.



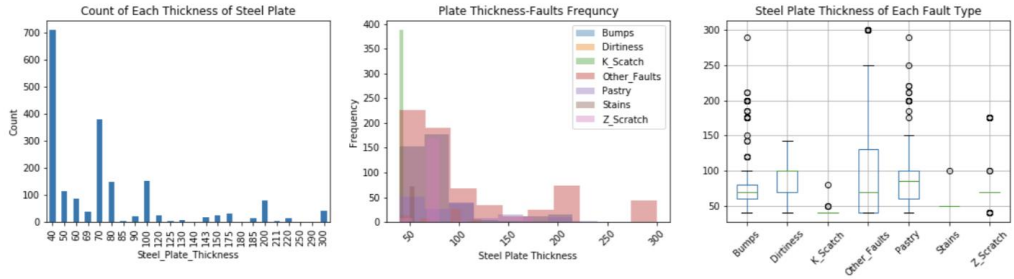
We could see K-Scatch has a relatively higher X perimeter and Y perimeter, and its mean X perimeter is larger than Y perimeter. Besides, Stains has a very small perimeter on both X and Y direction.

3) Here is how the steel type related to the classes.



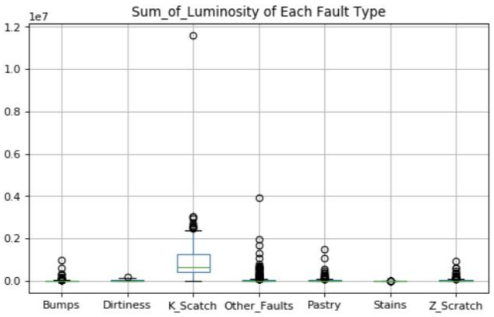
We could see that While most instances of Z_Scratch happen in A300 steel. More bumps happen in A300 than A400. More Dirtiness and Pastry happen in A400 than A300.

4) Here is how the steel plate thickness related to the classes.



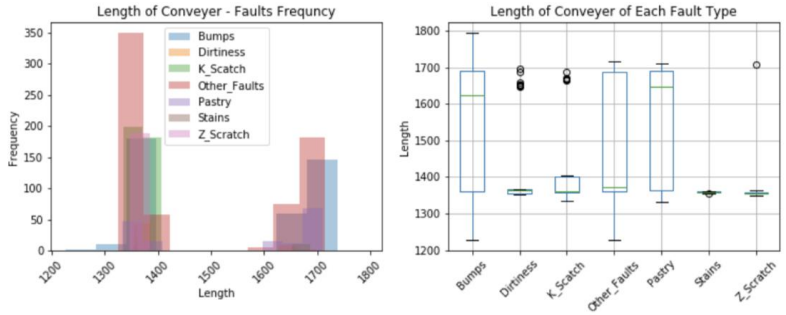
As we could see that, a lot of plates have thickness of 40 and 70. K_Scratch mostly happen in plates with thickness of smaller than 50. Stains mostly occur in plates with thickness of around 50. While Z_Scratch happen thickness of around 70. Bumps, Dirtiness and Pastry happen in plates with thickness between 50 and 100.

5) Here is how Sum of Luminosity related to the classes.

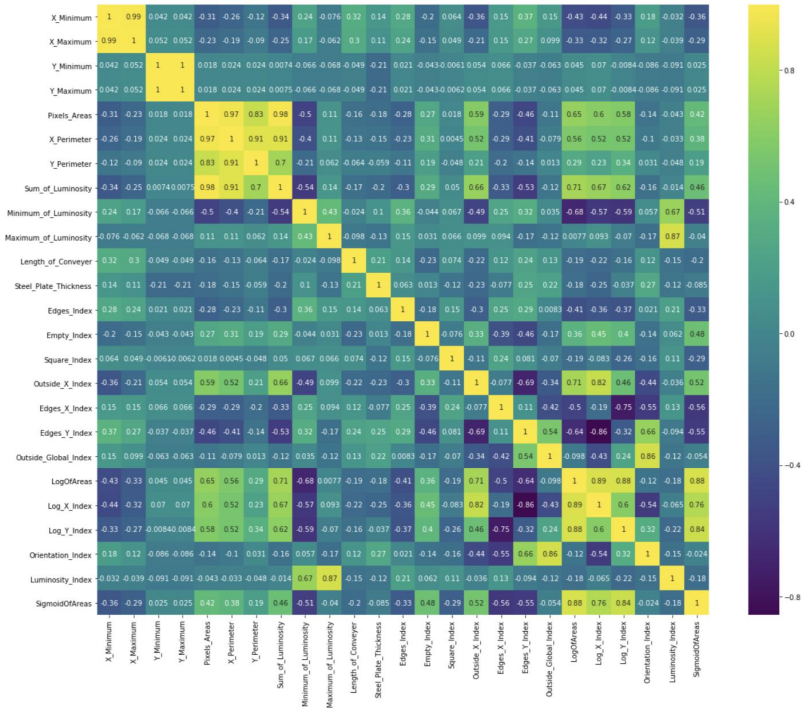


We could see that the sum of luminosity of K_Scatch is larger than the others, while its maximum of luminosity is of the same level of the others', so it is partly because of its relatively larger Pixel Areas.

6) Here is how the Length of Conveyor related to the classes



We could see that Bumps, Pastry and Other Faults have a larger scale of distribution on length of conveyor. All of the other four occur when the conveyor's length is in the range of 1350 to 1400.



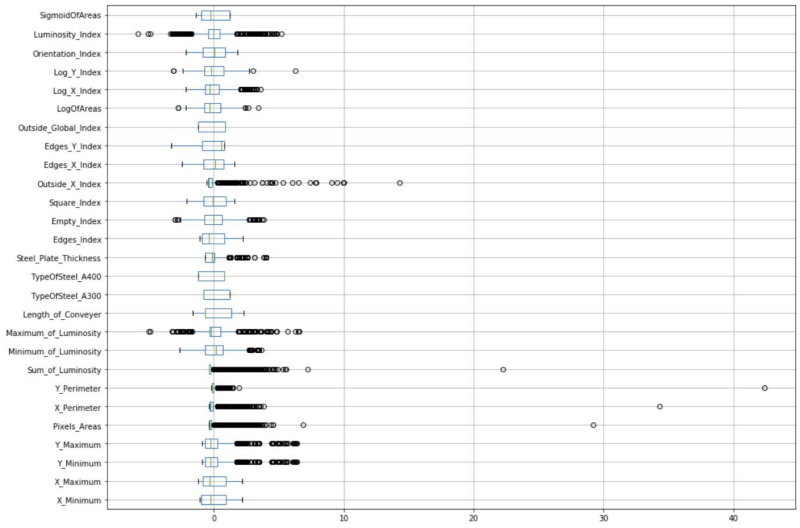
Next, I research on the relationships between the features using the heatmap.

As we could see from the heatmap that the features are not totally independent, the yellowest features are more correlated with each other, such as X_Maximum and X_Minimum, Y_Maximum and Y_Minimum and the other three features that are obviously correlated are Pixel Areas, X_Perimeters and Sum of Luminosity.

All of these findings are helpful for us to better understand the data and instruct us on how to moving forward next.

3. Preprocess the data

Firstly, I standardize the data with the Standard Scaler.



Secondly, after the standardization, we could see that there are still some outliers exist. For every feature, I use 1.5 times the interquartile range as the outlier threshold and remove the instance with more than 2 outliers. After that, there are 1494 instances left.

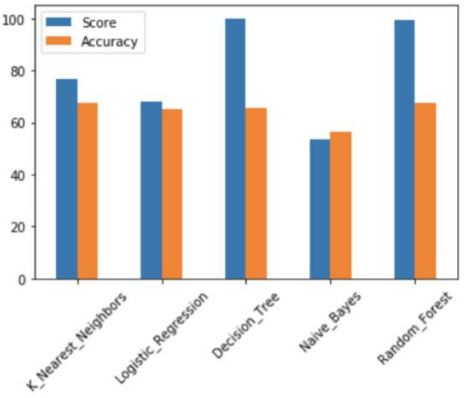
4. Classification

Train the data with 5 different classifiers:

- K Nearest Neighbors
- Logistic Regression
- Decision Tree
- Naive Bayes
- Random Forest

Split the data into 70% training set and 30% testing set. Conduct the training, here are the result.

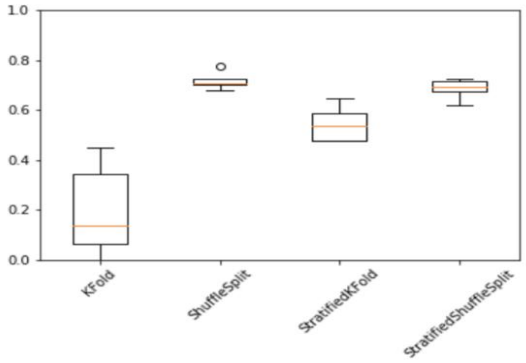
	Score	Accuracy
K_Nearest_Neighbors	76.842105	67.706013
Logistic_Regression	68.133971	65.033408
Decision_Tree	100.000000	65.478842
Naive_Bayes	53.588517	56.347439
Random_Forest	99.138756	67.483296



As we could see from the plot above, Random Forest has both a better score and accuracy, with score of 99% and accuracy of more than 67%.

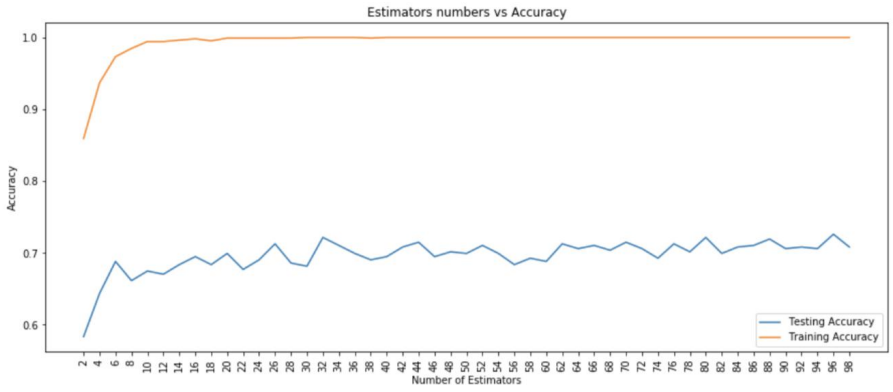
Next, we are gonna dig deep the Random Forest Classifier by tuning the parameters of the model and try to find the best parameters of it.

1) I try to find the best cross validation method for the classifier. I have tried with the following four of them, K Fold,Shuffle Split,Stratified K Fold and Stratified Shuffle Split.

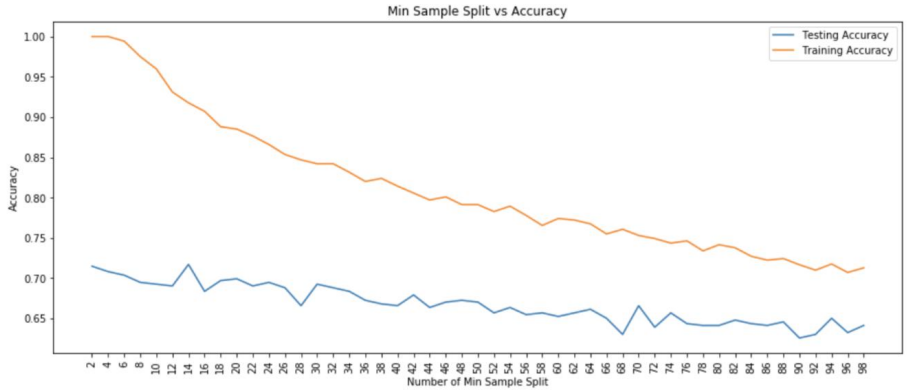


As we could see from the boxplot, StratifiedShuffleSplit works the best.

2) Then, I try to research on how the number of trees affects the performance of Random Forest Classifier. For which, I leave the other parameter default and trying with a series of estimator in the [2, 100], 2 each step. I find out that the best accuracy happens when the Estimators is 76.



3) Then, I try to research on how the number of split affects the performance of Random Forest Classifier. Similarly, I leave the other parameter default, and trying with a series of min sample split in the [2, 100], 2 each step. I find out that the best accuracy happens when Min Sample Split is 8.

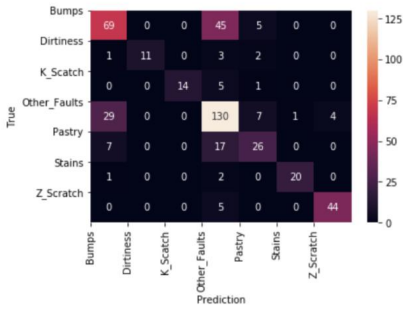


4) Then I adopt a grid search on the two parameters Estimators and Min Sample Split around 76 and 8 respectively using Stratified Shuffle Split as cross validation. I end up finding out that the best model with params of Estimator being 90 and Min Sample Split being 4, its performance are as follows.

```
score: 0.9971291866028709
accuray: 0.6993318485523385
report:
              precision    recall  f1-score   support

           0       0.64      0.58      0.61       119
           1       1.00      0.65      0.79        17
           2       1.00      0.70      0.82        20
           3       0.63      0.76      0.69       171
           4       0.63      0.52      0.57        50
           5       0.95      0.87      0.91        23
           6       0.92      0.90      0.91        49

    accuracy                           0.70       449
   macro avg       0.83      0.71      0.76       449
weighted avg       0.71      0.70      0.70       449
```



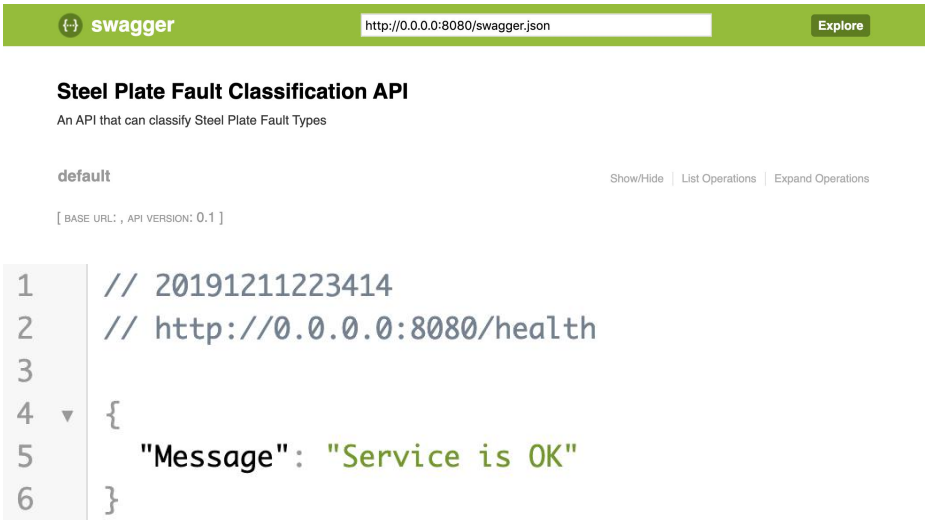
5. Callable Service

I design the interface with Flask and Swagger. And Create a Docker container for it. It will expose the model as an web url, other users could go to its predict path to make prediction with fresh new data by making a post request.

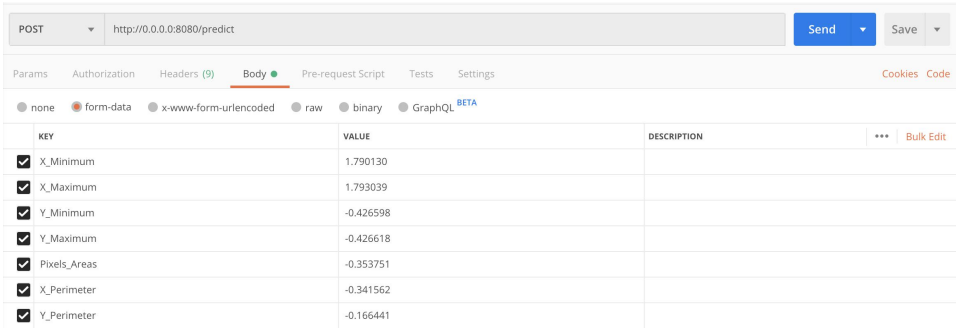
1) Start the micro-service

```
➜ final_project git:(master) ✗ docker run cs6220/final_project
 * Serving Flask app "steel_plate_fault_classification_api" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
/opt/conda/lib/python3.7/site-packages/sklearn/externals/joblib/externals/cloudpickle/cloudpickle.py
see the module's documentation for alternative uses
  import imp
/opt/conda/lib/python3.7/site-packages/sklearn/base.py:251: UserWarning: Trying to unpickle estimato
ight lead to breaking code or invalid results. Use at your own risk.
  UserWarning)
/opt/conda/lib/python3.7/site-packages/sklearn/base.py:251: UserWarning: Trying to unpickle estimato
ight lead to breaking code or invalid results. Use at your own risk.
  UserWarning)
 * Running on http://0.0.0.0:8080/ (Press CTRL+C to quit)
```

2) Check the status of the micro-service, it is ready to serve.



3) Go to Postman and make a post request with a piece of data.



4) Get the prediction

Conclusion

In this project, the data has been understood, preprocessed and model trained. Some interesting findings about the relationship between features and classes has been found. Several classifiers has been adopted to train the data and their appearance are being compared. It turns out that the Random Forest Classifier works the best. A best model is found by tweaking the classifier's parameters. Finally a callable micro-service for the steel plate fault type diagnosis is built.

By leveraging of this service, engineers in the steel plants could quickly diagnose the type of a fault. In this way, they could improve their manufacturing quality and reduce the cost for product testing.