Bearing Fault Diagnosis using Boruta Feature Selector and k-Nearest Neighbor Algorithm

Md Arafat Habib (20195851)
Course Instructor: Jongmyon Kim, PhD
Course name: IT based fault detection and diagnosis
University of Ulsan, South Korea

Rolling element bearing is a crucial part of modern-day machines and faults in any parts of them can lead to failure of the whole machine. Vibration-based diagnostic techniques are common, and they are highly effective for bearing health assessment. For this project, An RK-4 bearing testbed has been used. Figure 1 shows the setup. Specification regarding the test can be found in Table 1.

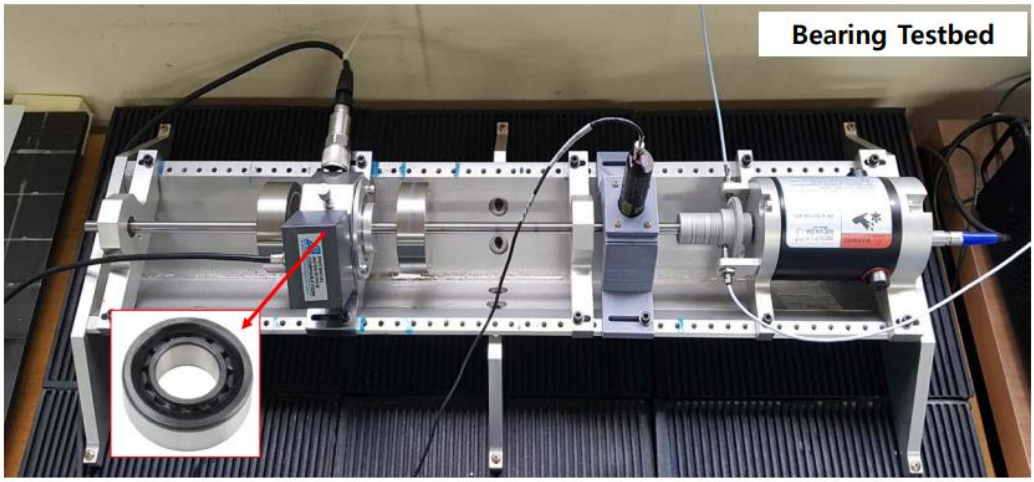


Table 1. Test specification for bearing

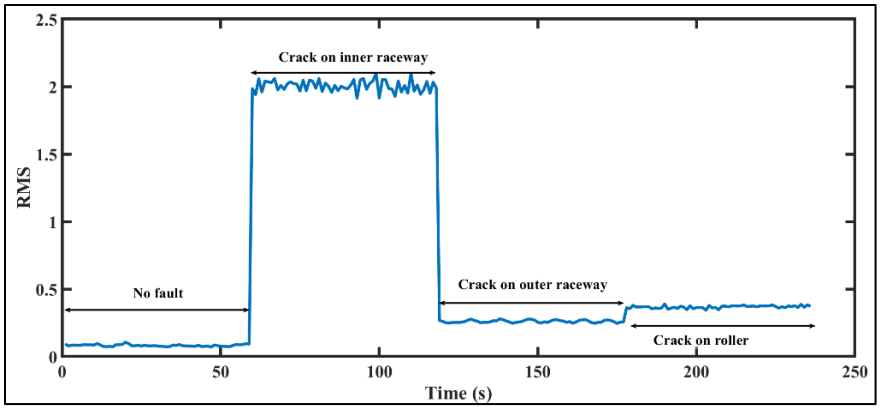
Specification	Values
RPM	1800
Sampling rate	256000 Hz
Signal length	1s
Bearing type	NJ206-E-XL-TVP2
Type of sensor	Vibration (622B01)

Three types of fault were considered. They are:

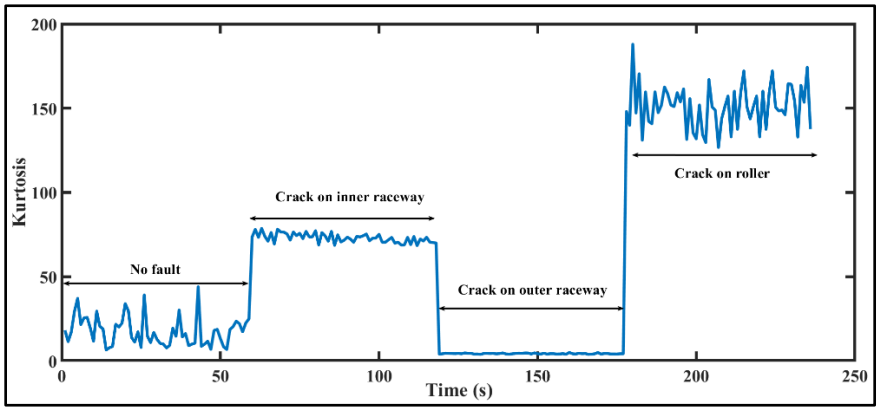
1. Crack on outer raceway

2. Crack on inner raceway
3. Crack on roller

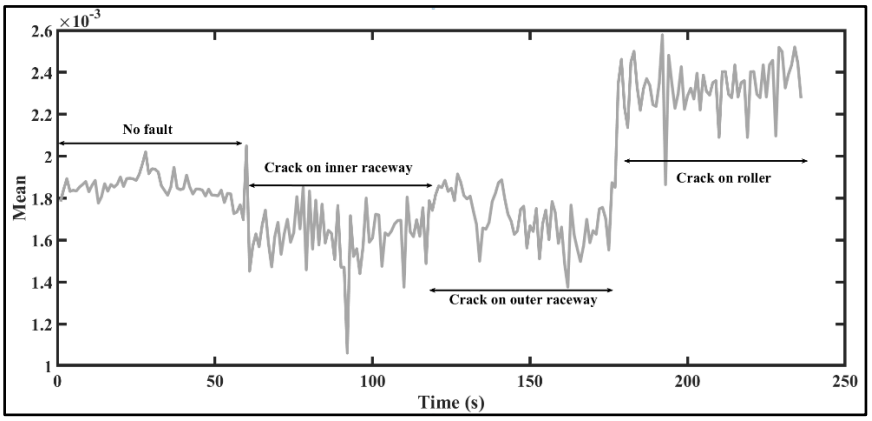
First of all, some of the common statistical features were obtained from the data using MATLAB. Five features were considered: rms, kurtosis, mean, median, and skewness. For each of these features, concatenated values for each fault type is presented in Figure 2a-e.



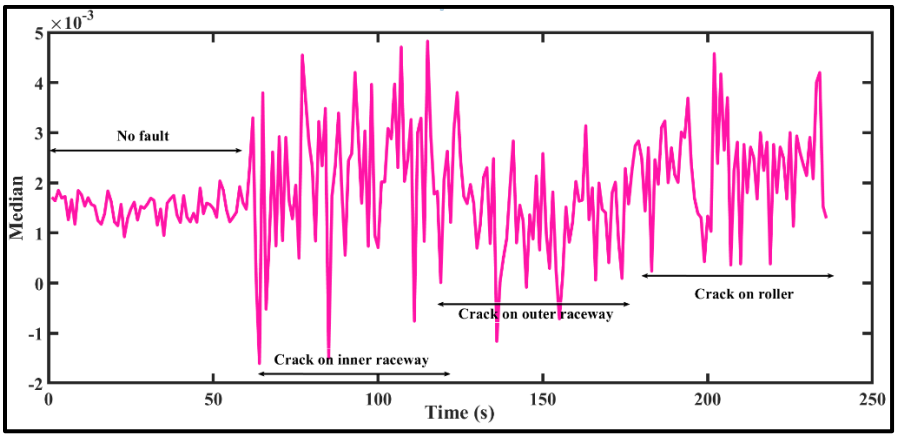
(a) rms



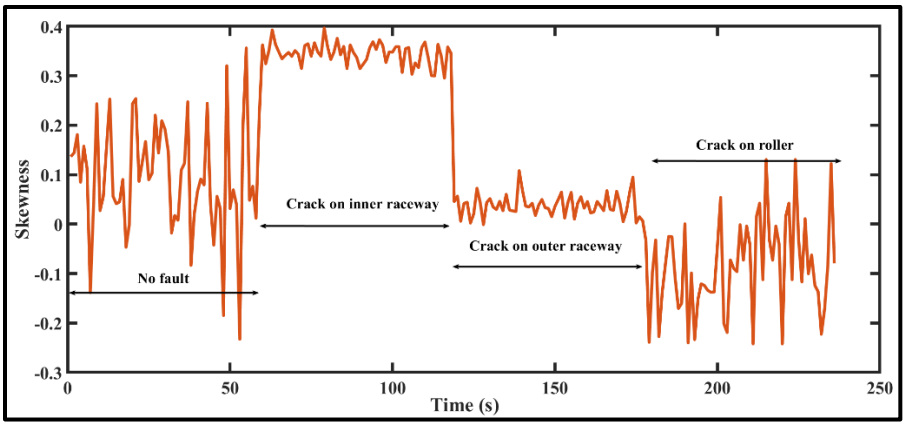
(b) kurtosis



(c) mean



(d) median



(e) skewness

Figure 2. Different statistical features calculated using the bearing data

Skewness, rms, and kurtosis could successfully distinguish among different faults whereas the other two features (mean and median) failed to differentiate between inner

and outer raceway faults. We have used Boruta feature selector algorithm to rank the features. It is a wrapper-based feature selector that uses random forest algorithm. Figure 3 shows the ranking of the features.

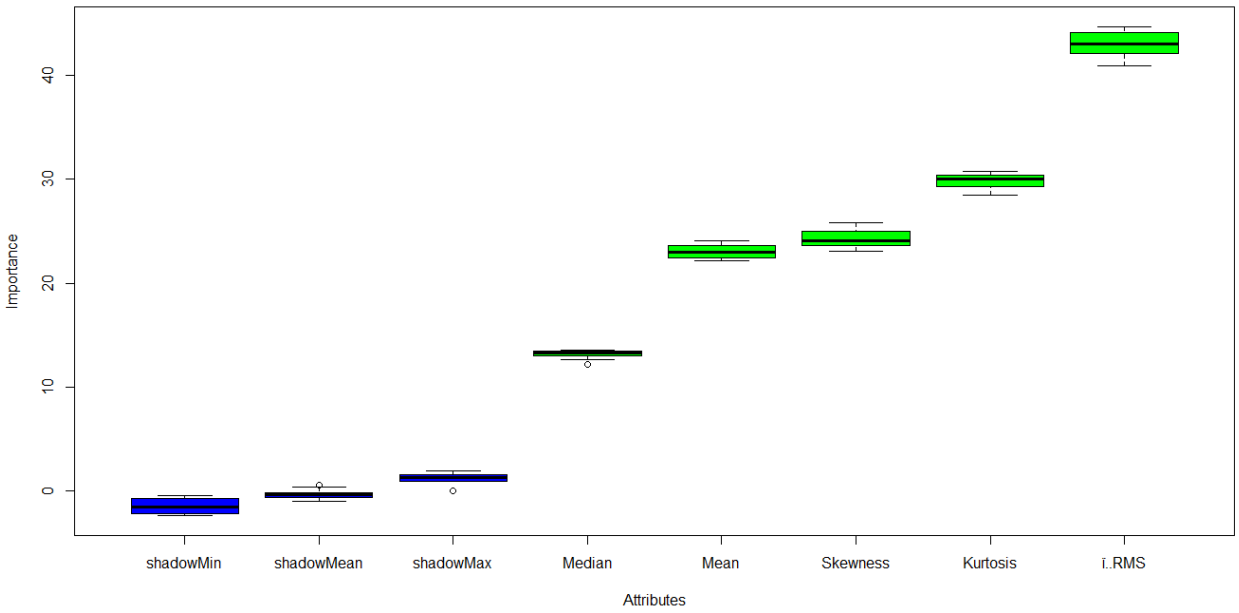


Figure 3. Feature ranking using Boruta algorithm

Next, we have applied k-nearest neighbor (k-NN) classifier for classifying different kinds of faults. The k-NN algorithm has acceptable performance for data with a small number of features, particularly if there are not many distinguishable outliers. The algorithm works based on three basic principles: 1) The calculation of distances among the neighbors, 2) the determination of the k closest neighbors to handle the bias invariance trade-off for finding a solution to the overfitting/underfitting phenomena, and 3) votes for labeling the data. Deciding the value of k in this algorithm is quite tricky. The algorithm was run using multiple values of k. 50 different values were used. Figure 5 shows us the accuracies we got for different values of k.

```
In [34]: Ks=50
         mean_acc=np.zeros((Ks-1))
         for n in range(1,Ks):
             neigh=KNeighborsClassifier(n_neighbors=n).fit(X_train,Y_train)
             yhat=neigh.predict(X_test)
             mean_acc[n-1]=accuracy_score(Y_test,yhat)

In [35]: print(mean_acc)

[0.98591549 0.97183099 0.98591549 0.98591549 0.98591549 0.98591549
 0.98591549 0.97183099 0.97183099 0.97183099 0.97183099 0.97183099
 0.95774648 0.97183099 0.95774648 0.95774648 0.95774648 0.94366197
 0.94366197 0.94366197 0.92957746 0.95774648 0.95774648 0.94366197
 0.92957746 0.92957746 0.90140845 0.90140845 0.91549296 0.88732394
 0.90140845 0.90140845 0.88732394 0.90140845 0.90140845 0.91549296
 0.91549296 0.90140845 0.91549296 0.90140845 0.90140845 0.90140845
 0.90140845 0.88732394 0.87323944 0.87323944 0.85915493 0.84507042
 0.85915493]
```

Figure 4. Variation of accuracy for different values of k

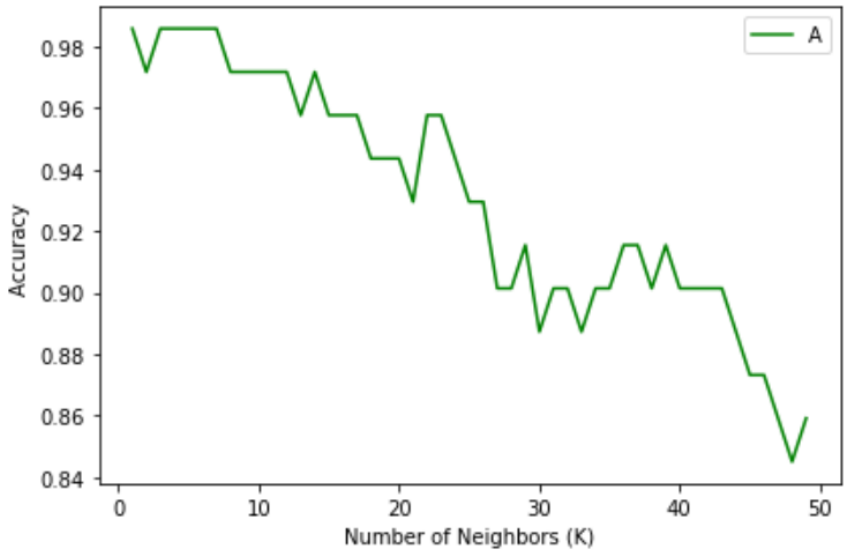


Figure 5. Accuracy Vs. k-value plot

Finally, the value for k was selected to be 3. Other performance measures were used to evaluate the proposed method. The average classification accuracy was calculated using following equation:

$$Avg.\_accuracy = \frac{True\_positive + True\_negative}{Total\_number\_of\_samples}, \quad (1)$$

where the term "*true_positive*" refers to the samples that were correctly classified and the term "*false_positive*" refers to the samples that are negatively classified from the provided test

data to the algorithm. We have also calculated the recall, precision and F1-score for performance analysis and they can be calculated using the following equations:

$$recall = \frac{True\_positive}{True\_positive+False\_negative}. \quad (2)$$

$$precision = \frac{True\_positive}{True\_positive+False\_positive}. \quad (3)$$

$$F1 - score = 2 \times \frac{Recall \times Precision}{Recall+Precision}. \quad (4)$$

Table 2 summarizes the values we got for these measures.

Table 2. Results of the classifier in terms of the evaluation parameters

Parameters	Value (%)
Average classification accuracy	98.59
Recall score	99.61
Precision	98.61
F1-score	98.64

Figure 6 presents the confusion matrix for the proposed method. The left most column presents the true labels and the topmost one presents the predicted labels. We can see that only two samples were misclassified.

	normal	Inner	Outer	Roller
normal	14	0	0	0
Inner	0	19	0	1
Outer	1	0	19	0
Roller	0	0	0	17