Drill bit fault detection and classification using k-nearest neighbor, decision tree, artificial neural network, support vector machine, and Naive Bayes classifier

Bhavesh Parkhe and Nino Figliola
University of Massachusetts Amherst

Abstract **- Accurate tool condition monitoring methods are imperative for modern manufacturing, including drilling processes. In this study, five different classification methods are used to detect faults and classify common modes of drill bit wear. The methods studied in this work are k-nearest neighbor, decision tree, artificial neural network, support vector machine, and Naive Bayes classifier. The detection and classification models are applied to data from the penetration drilling condition, as well as the steady-state condition. The models are also evaluated for their ability to detect specific types of drill bit wear.**

I. INTRODUCTION

In contemporary manufacturing, an increasing number of processes are becoming automated, including drilling. Because the effectiveness of a drill bit diminishes as it becomes worn, it is critical to monitor the condition of drill bits throughout their lifetime and replace worn bits once their performance decreases to a level warranting replacement. To realize viable tool condition monitoring methods, vibration sensors have been utilized to collect vibration data from functioning drill bits. Different types of drill bit wear result in different dynamic responses and vibration signatures, making vibration data an effective method for analyzing drilling [10]. Vibration analysis is especially advantageous for monitoring tool condition because it is noninvasive and enables continuous monitoring without interrupting the manufacturing process [1].

In the present work, vibration data was analyzed by five different classification models to detect drill bit wear and to determine which specific type of wear was affecting the drill bit. The following five classification models were used: k-nearest neighbor (kNN), decision tree, artificial neural network (ANN), support vector machine (SVM), and Naive Bayes classifier. The models exclusively used vibration data for tool condition monitoring [1]. The performance of these five models was compared for fault detection, fault classification, and detection of individual wear types. Three different types of wear that commonly result in drill bit failure were studied: flank wear, chisel wear, and outer corner wear [1].

II. EXPERIMENTAL SETUP

The dataset used for this work was sourced from the drill data repository from the experiments in [10]. The data was derived from a 3-axis CNC EMCO Concept Mill 105 which performed drilling experiments on a mild-steel work-piece with 9mm HSS twist drills [10]. Specifications for the uniaxial accelerometer and data acquisition card used to acquire and digitize the vibration signals are listed in Table 1 [10]. 15 different drilling conditions were tested, with each condition consisting of a unique combination of the drill feed rates and cutting speeds listed in Table 1.

Table 1
Parameters used in experimental procedure

Uniaxial accelerometer	Model: PCB Piezotronics 603C01 Measurement range: ± 50 g Sensitivity: 100 mV/g Frequency range: 0.5 to 10k Hz
Data acquisition card	Type: National Instruments DAQ Chassis model: NI cDAQ 9172
Drill feed rates	4, 8, and 12 mm/min
Drill cutting speeds	160, 170, 180, 190, and 200 rpm

The set of 15 tests was performed for a drill bit in perfect condition as well as for drill bits in each of the three artificially-induced wear conditions:

flank wear, chisel wear, and outer corner wear [10]. A single test spanned a time period of eight seconds and collected data at a rate of 32,768 samples per second [10]. Data was collected for two different feeding conditions: steady-state, in which the drill bit does not travel vertically, and penetration, in which the drill contacts and passes through the work-piece in the vertical direction [10].

III. DATA PREPROCESSING

The raw data was separated into steady-state and penetration experimental data sets in order to perform the fault detection and classification analyses separately for each condition. The separation yielded two datasets of 60 experiments each. Pre-processing of the experimental data was necessary in order to remove the significant noise present in the vibration sensor data [10].

A. Digital filtering

The noise is digitally filtered from the data with a low-pass Butterworth filter of order 20 and cutoff frequency of 12kHz. Filtering the data gives similar results when compared to the method of cutting off frequencies above a certain range for FFT analysis. However it is a good practice to filter data since it influences the time domain features accordingly.

B. Feature extraction

A set of 21 features were extracted from the pre-processed data. The set consisted of eight time-domain features, eight frequency-domain features, and five Morlet wavelet features [10]. The time-domain features consisted of mean, maximum peak, root mean square, variance, kurtosis, crest factor, shape factor, and skewness. Frequency domain features were derived from a spectral analysis of the vibration data from each experiment. The vibration frequency response was divided into eight equal bins and for each bin, a feature was calculated by finding the ratio of spectral energy contained in that bin to the total spectral energy across all eight bins [10]. The Morlet wavelets were calculated using the parameters specified in [10]. The features derived from the convolution of the wave and signal were its standard deviation, wavelet entropy, kurtosis, skewness and variance.

C. Feature selection

In order to use the features for fault detection it is necessary to make sure there is a correlation between the faults and the features. Features which have low correlation would consume additional data for little value addition and may influence the output negatively in worse cases. A correlation cutoff of 30% was set and the resulting features were sent for further processing.

The features were further processed using principal component analysis (PCA). The purpose of applying PCA is to reduce the dimensionality of the features and have fewer variables for analysis. For a comparative analysis, PCA was performed on both the reduced correlated features and the original set of 21 features. Using eigenvalue decomposition, the features with highest correlation between them were combined.

For the fault-correlated features, the number of final features available varied depending on the fault-types selected. Using a leave-one-out approach, the correlated features were calculated for all sets of faults and the results were combined to give a set of all features with correlation of more than 30% with the faults in either case.

The number of principal components that would be input into the subsequent detection and classification models varied between 1 and the maximum number of available principal components in order to determine the effect of retaining different quantities of principal components on the final classification accuracy [10].

IV. CLASSIFIER MODELLING

kNN is a nonlinear classification method in which a new object is classified based upon its distance to the nearest training samples of known classification. The new object is assigned to the class that accounts for the majority of the “k” nearest training samples. kNN is a simple method whose strength is its ability to classify nonlinear, multimodal, unlabeled samples based upon their

Table 2
Specifications for classifier design

k-nearest neighbor (kNN)	
Number of nearest neighbors:	3
Distance measurement metric:	Euclidean
Equal or weighted voting:	Weighted
Distance weight:	Squared-inverse
Decision tree	
Maximum number of splits:	5
Split predictor technique:	CART
Artificial neural network (ANN)	
Neural network type:	Feedforward
Number of layers:	3
Number of hidden layers:	2
Neurons per hidden layer:	20
Transfer function (hidden layers):	Tangent-sigmoid
Transfer function (output layer):	Pure-linear
Training method:	Levenberg-Marquardt
Performance metric:	Mean-squared error
Support vector machine (SVM)	
Kernel Function	Linear
Box Constraint	0.1
Coding Matrix	Onevsone & onevsall
Standardization	False
Naive-Bayes	
Distribution name	Kernel
Width	9
Kernel	Normal

similarity to training samples [7]. It is widely used in industry for fault classification [7]. To design a kNN classifier, it is necessary to specify the number of nearest neighbors (k) used to classify a new sample, the measurement method used to determine the distance between sample data points, and the voting scheme used to determine the new sample's classification. The variable k acts as a smoothing parameter. Low values of k can result in a noisy model, while high values can yield a model that is excessively smooth. k-values of three, five, and seven were tested, and optimal performance resulted from a k value of three. The Euclidean distance metric was used to determine the distance between data points. For voting scheme, it was found that "squared-inverse", a type of weighted-sum voting, yielded high detection and classification accuracies. Instead of voting based on majority, weighted-sum voting gives near-neighbors greater influence on the classification than distant neighbors [7]. For squared-inverse, the weight is the reciprocal of the squared distances [7].

A decision tree is a type of classification model based upon if/then rules that consists of a network of binary branches originating from an initial root node. Links between branches represent decision points that successively work to classify the new input data point. The final layer of branches on the decision tree leads to endpoints called "leaves" that represent classification labels. The decision tree in this work followed the standard CART split predictor selection technique. The CART algorithm often generates deep decision trees that overfit the data [9]. Deep trees can achieve high accuracy on the training data, but are prone to poor performance when classifying a novel test set [9]. When classifying new tests sets, simple, shallow, generalized decision trees are often more robust than deep trees [9]. In order to prevent overfitting, the maximum number of splits for each decision tree was limited to five.

An artificial neural network is a classification method based upon the human brain that receives input data and implements weight, biases, and transfer functions to deliver an output. During training, the weights and biases are adjusted so that the ANN's output matches the target values [5]. ANN is an effective tool for recognizing patterns in noisy, complex data and determining their nonlinear relationships [1]. An ANN consists of at least one hidden layer followed by an output layer. The output layer consists of as many neurons as the system has outputs. In this work, the detection ANN included two neurons in the output layer because there were two possible outputs, whereas the classification ANN included three, corresponding to the three wear types that constituted the output. Multilayer networks are more powerful than single layer networks, but it is rare for practical neural networks to have more than three layers [5]. In this work, tests were performed with ANNs of varying numbers of layers and neurons. A three-layer feedforward network, comprised of two hidden layers and one

output layer, with 20 neurons per hidden layer, was found to achieve a good balance between classification accuracy and computing time. The trained ANN was generated using Matlab's Deep Learning Toolbox™. In a feedforward ANN, data is input into the first layer and subsequently processed by the following layers until an output is provided by the output layer [4]. A recurrent ANN was not chosen because the input data was not time-dependent [5]. In addition to the number of layers and neurons, a transfer function must be specified for each layer. Transfer functions determine the output of each individual layer. Tan-sigmoid transfer functions were used for the hidden layers and a linear transfer function was used for the output layer.

SVM is a binary classification technique which is used to establish boundaries between classes based on maximum margin from the closest points in two distinct classes. Multiclass application of SVM consists of training N binary SVM models and combining them together. MATLAB offers an error correcting output codes (ECOC) toolbox for multi-class SVM which trains a model based on multiple binary solvers. For the binary solvers, a positive and negative class should be defined. The coding matrix defines the positive or negative classes for the solvers in each column. One-vs-one and one-vs-all approaches are more suitable for our application since we do not have an unequal preference for a fault occurring in the drill. Standardizing the input affects the classification negatively since they have been standardized during the PCA scores calculation. The parameters for optimization were calculated using hyper parameters optimization with grid search.

Naive Bayes classifier is a probabilistic model which ignores correlation between individual input features while assigning probabilities for each class label. It is based on the Bayes Rule of probability where calculating the probability of an event given certain predictor value can be estimated using individual probability of the events and the probability of the predictor given a particular class. In this paper, data is kernel smoothing with a window width of 9 has been used based on grid search. A Gaussian kernel has been used to specify distribution for kernel smoothing.

V. TRAINING AND TESTING

Following PCA of the feature selection data, the set of reduced features was used to train and test the five detection and classification model types: kNN, decision tree, ANN, SVM, and Naive Bayes. Several practices are recommended in [6]. The usual practice of testing the model using cross validation within the training data gives a training error. For testing error, the samples must be physically separated from the training data set and labels must be compared after predicting the labels for the test samples using the trained model. Applying this methodology over the entire domain and averaging the errors yields the test error. Standard deviation of this error is also helpful in determining the effectiveness of the model.

In this work, the given classified data was used for both purposes by being separated into testing sets and training sets. The training data of known classification was used to train each classifier for fault detection, fault classification, and detection of specific fault types for both penetration data and steady-state data.

Following training, the testing data set, consisting of samples that were not included among the training data, was used to determine the accuracy of each classifier. Accuracy was determined by comparing the output of the classifier to the known correct output and dividing the quantity of correct classifications by the total number of test samples. The results from each fold's test set were combined and used to determine an overall accuracy for the classifier. For detection classifiers, the two possible outputs consisted of "faulty" and "non-faulty". Thus, the detection classifiers functioned as binary classifiers. For the fault classification classifiers, there were three possible outputs representing the wear types: "flank", "chisel", and "outer corner". The models were also trained for an alternate classification method in which a specific type of wear served as the detection target. In that case, the known classifications consisted of two classes, for instance, flank-wear and non-flank-wear. The process described here was applied separately to the penetration data and the steady-state data in

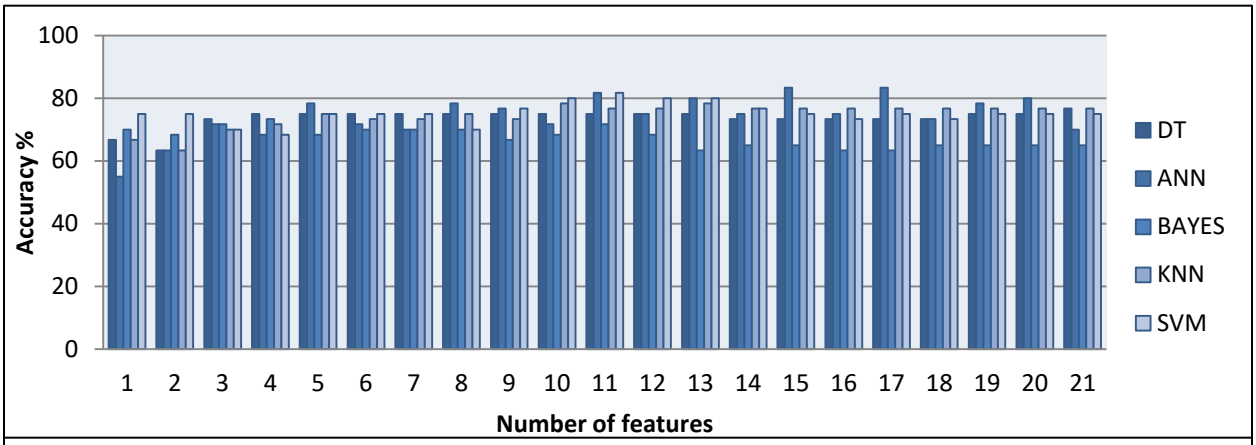

Figure 1: Fault detection accuracy based on penetration data

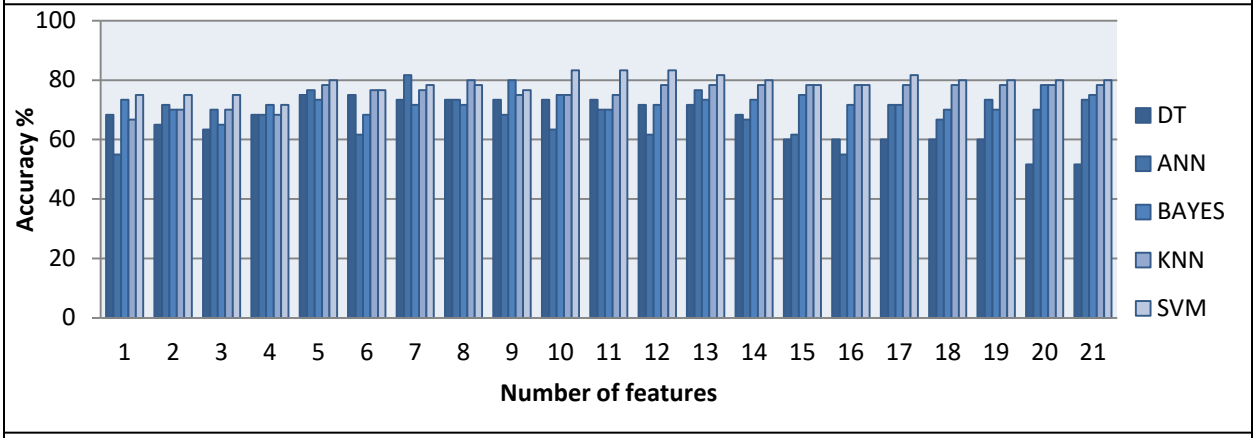

Figure 2: Fault detection accuracy based on steady-state data

order to train and test detection and classification models for both conditions independently.

VI. RESULTS AND DISCUSSION

By plotting detection and classification accuracies against the number of principal components retained, it is possible to observe stabilizing levels of accuracy as the number of retained principal components increases. The results from penetration and steady-state data are given as follows.

A. Penetration stage data

As seen in the penetration stage data in Figures 1 and 3, the kNN classifier achieved a stable accuracy of 77% for fault detection and 73% for classification. The decision tree produced similar results for fault detection, reaching a stable accuracy of 75%. However, it performed worse for classification, reaching approximately 50% accuracy. Results for the ANN classifier were less consistent. For detection, typical accuracies fell between 70% and 83%. For classification, typical accuracies ranged from 51% to 73%. SVM and Naive Bayes achieved 75% and 65% detection accuracies, respectively, while for classification, they reached 87% and 71% accuracy, respectively.

The classifiers detected flank wear with accuracies between 62% and 82%. The accuracy of the kNN classifier stabilized between 80% and 82% while the decision tree consistently reached accuracies between 62% and 71%. The ANN's classification accuracies were less consistent, falling between 63% and 78%. SVM and Naive Bayes detected flank wear with 82% and 76% accuracy, respectively.

Accuracies of chisel wear detection between 60% and 93% were achieved by the penetration data classifiers. The accuracy of kNN stabilized

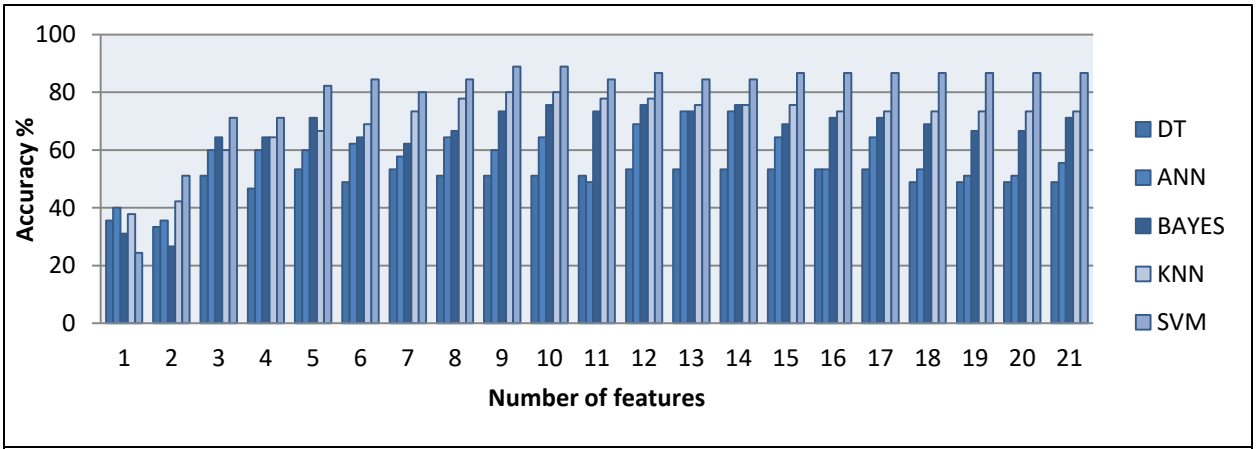

Figure 3: Fault classification accuracy based on penetration data

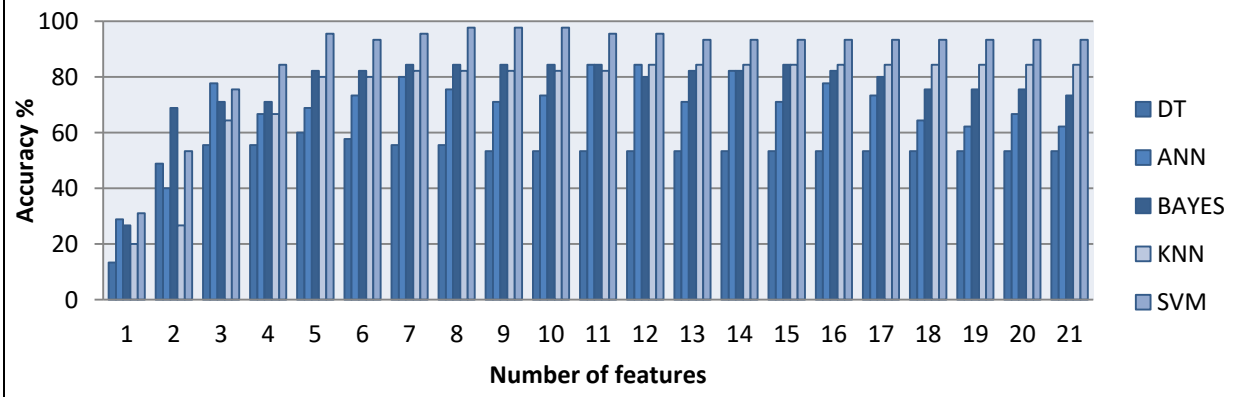

Figure 4: Fault classification accuracy based on steady-state data

between 80% and 82% and that of the decision tree stabilized between 64% and 67%. Accuracy of the ANN classifier spanned a wider range, from 55% to 77%. SVM and Naive Bayes detected chisel wear with 91% and 93% accuracy, respectively.

The classifiers were able to detect outer corner wear with accuracies ranging from 60% to 96%. Stable accuracies of 91% and 71% were reached by kNN and decision tree respectively. The accuracy of ANN varied from as low as 60% to as high as 83%. SVM and Naive Bayes detected outer corner wear with 96% and 71% accuracy, respectively.

B. Steady-state stage data

As seen in the steady-state stage data in Figures 2 and 4, kNN reached a stable detection accuracy of 78% and a stable classification accuracy of 84%. The decision tree classifier demonstrated inconsistent detection performance, with accuracies ranging from 52% to 75%. It reached a stable classification accuracy of 53%. The detection accuracies of the ANN classifier fell between 55% and 82% while its classification accuracies ranged from 62% to 84%. SVM and Naive Bayes reached detection accuracies of 80% and 75%, respectively, and classification accuracies of 93% and 73%, respectively.

For detecting flank wear, accuracies between 75% and 93% were reached. kNN achieved a stable accuracy of 93%, while the decision tree classifier reached a stable accuracy of 76%. ANN returned relatively stable accuracies of 81% to 90% accuracy. SVM and Naive Bayes detected flank wear with 87% and 80% accuracy, respectively.

Chisel wear was detected with accuracies ranging from 60% to 91%. kNN reached a stable accuracy of 87%. Decision tree reached a stable accuracy of

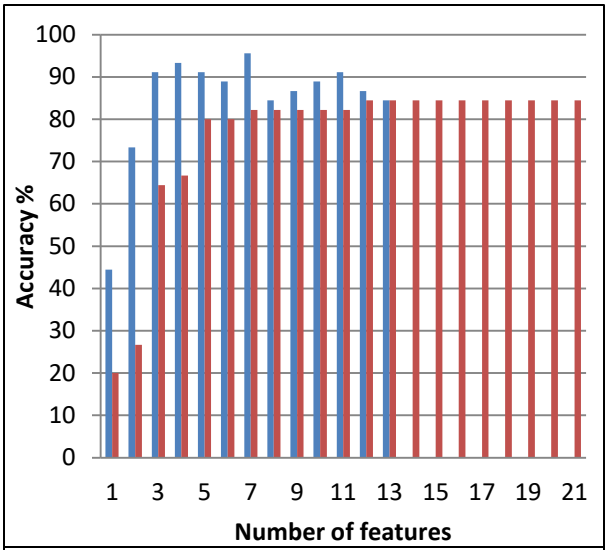

Figure 5: kNN fault classification accuracy comparison between the full set of features (red) and the reduced set of correlated features (blue). Results based upon steady-state data.

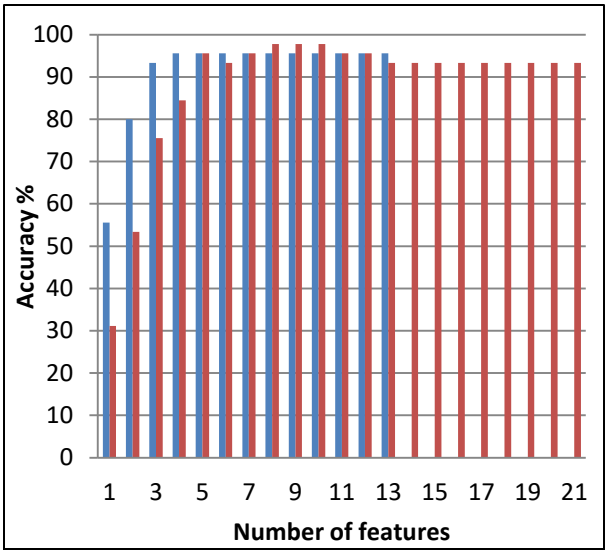

Figure 6: SVM fault classification accuracy comparison between the full set of features (red) and the reduced set of correlated features (blue). Results based upon steady-state data.

73% and the accuracy of ANN fell between 60% and 82%. SVM and Naive Bayes detected chisel wear with 91% and 73% accuracy, respectively.

Detection of outer corner wear reached accuracies between 62% and 100%. The output accuracy of kNN stabilized at 93%. The accuracies of decision tree fell between 62% and 67%. Those of ANN ranged from 65% to 85%. SVM and Naive Bayes detected outer corner wear with 100% and 80% accuracy, respectively.

Detection accuracies were generally higher than classification accuracies, which was expected since the models functioned as binary classifiers for detection and multi-class classifiers for classification.

Detection accuracy was generally higher for penetration data than for steady-state data. However, the classifiers exhibited higher classification accuracy based off of steady-state data than for penetration data. The greater noise present in the raw penetration data may have obfuscated the data enough to reduce classification accuracy. By reducing noise further by optimizing the data preprocessing, it may be possible to achieve higher classification accuracy.

It is evident from the results that SVM and kNN generally achieved the highest accuracies. Both classifiers achieved high, stable accuracies for fault detection, fault classification, and individual detection of flank wear, chisel wear, and outer corner wear. The strong performance aligns with kNN's strength in classifying nonlinear, multimodal data.

Possible explanations for the poor performance of the decision tree are excessive sensitivity to changes in the training data, and the fact that each leaf of the decision tree represents a singular, constant output, thereby resulting in discontinuities among leaves [2].

The poor performance of the ANN is most likely due to overfitting. The trained ANN was able to classify training data without error, but could not do so for testing data. A means to prevent overfitting is to provide more training samples. Doubling the number of training samples could be accomplished by splitting each sample of 8-second duration into two samples of 4-seconds each.

It is evident from Figures 5 and 6 that the method of training classifiers with a reduced set of highly fault-correlated features, as described in section III-C, increases the classification accuracy at low

numbers of retained features for kNN and SVM classifiers. Similar improvement was observed when the reduced sets were used to detect specific wear types.

VII. CONCLUSIONS

With varying degrees of success, five classification methods were used to detect and classify faults from vibrational drilling data. SVM and kNN classifiers performed with the highest accuracy and are thus recommended for application in drill bit fault monitoring systems. SVM and kNN were able to achieve stable detection accuracies of 75% and 78%, respectively, and stable classification accuracies of 93% and 84%, respectively. The other classifiers tested, decision tree, ANN, and Bayes classifier, did not reach the same levels of accuracy.

Results from this work demonstrate that classifiers are generally able to classify faults more accurately based on steady-state data than penetration data. Likewise, the classifers' detection of individual wear types was generally more accurate when based on steady-state data.

Further work could be performed to integrate the aforementioned five classifiers into ensemble classification methods, whereby considering the output of multiple classifiers could yield even higher detection and classification accuracies. Additionally, improved ANN performance could be attained by re-training the classifier with a greater number of training samples.

Promising results were obtained by using a reduced set of highly correlated features to train kNN and SVM classifiers. Further work would also involve expanding this approach to the other classification methods.

REFERENCES

[1] Abu-Mahfouz, I. "Drilling wear detection and classification using vibration signals and artificial neural network". International Journal of Machine Tools & Manufacture, vol. 43, pp. 707-720, 2003.

[2] Bishop, C.M. *Pattern Recognition and Machine Learning*. New York: Springer, 2006.

[3] Dimla, D.E. "Sensor signals for tool-wear monitoring in metal cutting operations—a review of methods." International Journal of Machine Tools and Manufacture, vol. 40, no. 8, pp no. 1073-1098, 2000.

[4] "Feedforwardnet". *Mathworks*, The Mathworks Inc., 2018, https://www.mathworks.com/help/deeplearning/ref/feedforwardnet.html

[5] Hagan, M.T., Demuth, H.B., Beale, M.H., and De Jesus, O. *Neural Network Design - 2nd Edition*. Martin Hagan, 2014.

[6] Hastie, T., Tibshirani, R. and Friedman, J."The Elements of Staistical Learning". Springer, 2009, https://www.mathstat.dal.ca/~aarms2014/StatLearn/docs/05_annotated.pdf

[7] He, Q.P. and Wang, J. "Fault Detection Using the k-Nearest Neighbor Rule for Semiconductor Manufacturing Processes". IEEE Transactions on Semiconductor Manufacturing vol. 20, no. 4, pp. 345-354, 2007.

[8] "Improve shallow neural network generalization and avoid overfitting". *Mathworks*, The Mathworks Inc., 2018

[9] "Improving classification trees and regression trees". *Mathworks*, The Mathworks Inc., 2018, https://www.mathworks.com/help/stats/improving-classification-trees-and-regression-trees.html

[10] Kumar, A., Ramkumar, J., Verma, N.K., and Dixit, S. Detection and Classification for Faults in Drilling Process using Vibration Analysis. *International Conference on Prognostics and Health Management*, Cheney, WA, USA, 22-25 June 2014. IEEE.