Detection and Classification of Interference In an Industrial Wireless System

Donovan Quimby
dquimby3@gatech.edu
ISYE 7406 Data Mining and Statistical Learning
Group 120
GTID:xxxxxx532
13March2022

1 Abstract

Network-based wireless communication systems for cyber-physical systems, such as wirelessly controlled robots in a factory, are increasingly enabling technologies such as smart manufacturing. However, challenges such as wireless signal interference may impair the quality of the wireless information flow resulting in costly or dangerous equipment malfunctions. This work seeks to study data mining and machine learning algorithms that can detect and classify interference in the wireless communication system to alert operators or automatically take corrective measures to avoid system issues. The Machine learning models are developed using the "Measurement Data for a Wireless Force Seeking Apparatus" dataset available at the National Institute for Standards and Technology (NIST) public data repository. K-nearest neighbor, linear discriminant analysis, neural network, and support vector machines are tuned and trained on data sets of varying time-segment lengths. This work demonstrates that creating time segments and proper tuning and selection of models is required for good classification of multiple levels of interference. Furthermore, this work shows that model performance generally increases as the length of the time segment increases. The support vector machine and neural network models perform significantly better than the other models across all segment lengths greater than $M = 5$.

2 Introduction

The rate of adoption of industrial wireless systems (IWS) for cyber-physical systems (CPS) in factories has increased significantly recently. IWS systems are used in numerous applications such as smart manufacturing, process control, safety systems, and inventory control [2]. The IWS systems' benefits are significant and include lower installation cost, flexibility, and scalability due to a lack of cabling. However, these benefits come with challenges of increased information loss, latency, error uncertainty, and the possibility of significant transient signal interference [11, 12, 17]. Sources of interference can include signal reflections, other wireless systems, non-communication equipment electromagnetic emissions, and intentional jamming [4].

Interference may cause significant issues for an IWS system. For example, a wireless robotic assembly machine may be controlled via a closed-loop system based on information collected by sensors on the robot. The information is sent wirelessly to a controller unit, which plans and adjusts the robot's movement based on the received signal. The commands from the controller are sent wirelessly back to the robot. A degradation or error in either of these signals can lead to costly damage to the robot or product and possible hazardous conditions for workers. Therefore, it is advantageous to detect interference and classify the severity to deploy proactive mitigation.

This work aims to develop machine learning models that will classify if the signal of a wireless robotic system is experiencing no interference, mild interference, or severe interference based on features extracted from time-series data of the robot's movement. However, it is important to be able to classify the data accurately and be able to classify within as short of a time frame as possible. Therefore, the models are evaluated to determine the effectiveness of the models at different time segment lengths ranging from 1 repetition of the robot's movement to 40 repetitions. This is challenging because the chosen models need to balance the trade-off between the effectiveness of the multi-class classifier with the ability to do it with as little data (shortest time) as possible.

Figure 1. Photograph of robots in a industrial manufacturing scenario [13].

Four models were investigated in this work: K-Nearest Neighbors (KNN), Linear Discriminate Analysis (LDA), Single Hidden Layer Feed-Forward Neural Network (NN), and a Support Vector Machine (SVM). When applicable, model hyperparameters were tuned using cross-validation.

Evaluation of the models' performance is challenging because it is a multi-class classification problem where the real-world application needs to consider the trade-off between sensitivity and specificity. On the one hand, correctly detecting the presence of interference is essential for safety concerns and to avoid costly damage to equipment or material. On the other hand, high false-positive rates can lead to unnecessary costly shutdowns or operator conditioning which may lead to ignoring warnings from the algorithm. For this work, multi-class AUC, balanced accuracy, and multi-class confusion matrices are used to evaluate the performance of the models.

3 Data

3.1 Data Description

The data used for this work will be the "Measurement Data for a Wireless Force Seeking Apparatus" dataset available at the National Institute for Standards and Technology (NIST)

public data repository [1]: Link to data website. This practical industrial wireless use case contains data involving a robotic manipulator control system, integrated wireless force-torque sensor, and a remote vision-based observation system. A source of interference is injected directly into the wireless communication channel via a jamming device. The receiver distance to the jammer controls the power of the interference at the receiver. The collected raw time series-based data is processed, and 5 features are extracted for various interference levels. Figure 2 shows a schematic of the experimental setup.

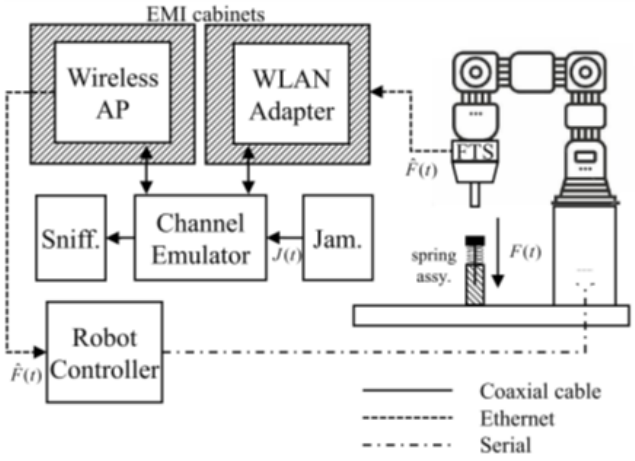


Figure 2. Experimental setup for the dataset used in this analysis [3].

This work examines a subset of the dataset focusing on classifying the data into three signal-to-interference power ratio (SIR) categories:

1. No interference (baseline)
2. -1 dB interference
3. -9 dB interference,

where SIR at the receiver is defined in decibels (dB) as:

$$SIR = Power_{signal} - Power_{interference} \quad (1)$$

Two things are noted to avoid confusion. First, the decibel (dB) scale is logarithmic, which leads to the signal-to-interference ratio definition, typically expressed as a division in non-logarithmic scales, simplifying to the subtraction in equation 1. As a result, a smaller SIR value indicates more interference, and an interference level of 0 indicates equal signal and interference powers, not the absence of interference. A negative value of SIR means the interference is stronger than the signal. Hence, -9 dB has a higher interference than -1 dB or +9 dB.

The data provided in the publicly available data set consists of 5 features engineered from raw time-series data where every observation represents 1 cycle of the robots movement. The robot's cycle consists of a probe at the end of a plunger that starts from an elevated baseline position, descends downward, pushes a spring past a pre-determined threshold, and then ascends back to the baseline position. Each observation (row) in the provided data corresponds to 1 cycle of the robot's movement. The cycle is repeated multiple times to provide $n \approx 800$ observations, or rows, for each interference level. The 5 engineered features are:

1. z_d The length of the robot's probe decent measured in mm
2. Δt_{ab} The duration in seconds the robot waits before starting to move the probe dwon
3. Δt_{bc} The length of time it takes to move the probe beyond a threshold limit, Z_{th}
4. Δt_{cd} The duration the probe remains below threshold Z_{th}
5. Δt_{ac} The duration of the full cycle

The observations in the provided data set represent 1 cycle of the robot movement. However, modeling in this work is performed on data segments comprised of successive cycles where segment size is denoted as M, which is composed of $\frac{M}{5}$ cycles of the robot's movement. Therefore, each observation used in this analysis is a vector of size M with a label corresponding to the SIR value.

The creation of the segments is accomplished by using a sliding window of size M. A representative illustration of the sliding window is shown in figure 3. Analysis of engineered times-series segments like this has been shown to increase the accuracy of models [3]. The following M values were investigated, noting that $M = 5$ is essentially the unsegmented data:

- $M = 5$, 1 cycle of robot's movement
- $M = 50$, 10 cycles of robot's movement
- $M = 100$, 20 cycles of robot's movement
- $M = 150$, 30 cycles of robot's movement
- $M = 200$, 40 cycles of robot's movement

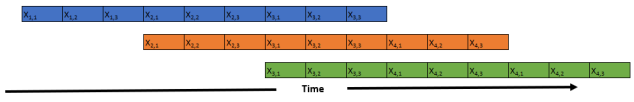


Figure 3. Example sliding window used to create segments for analysis. In this example, the data has 3 features and the sliding window size is composed of 3 observations, or $M = 9$. $X_{i,j}$ is defined as feature j from observation i. The created segments are distinguished by color.

Data segments constructed like this result in highly correlated features not only by including the same engineered features, collected at successive times, but also because it is time-series data that can display high auto-correlation. The high correlation could result in poor performance for models

highly sensitive to correlated features and must be considered in choosing appropriate models. Additionally, care must be taken in splitting the data into training and testing sets to avoid training data leaking into the testing data.

3.2 Data Exploration

Data exploration is conducted on the dataset for $M = 5$. Histograms, Pearson correlation coefficients, and feature interaction scatter plots were calculated and plotted for inspection as demonstrated in Figures 7, 8, 9 in the appendix. As seen in the figures, all features have a weak correlation with each other except for z_d (Touch.Down.position.mm) and Δt_{bc} (Delta.t.plunge), which are strongly correlated. Furthermore, the scatter plots and histograms show that many feature values are discrete in practice, although they are supposed to be continuous. This is likely due to the resolution of the measurement equipment. A handful of possible outliers are noted, but are left in the analysis because of a lack of context on their nature. Furthermore, this would likely better mimic real-world streaming data, which would undoubtedly contain similar outliers.

Box plots of each feature for each of the labels are shown in Figure 4. Similar to what was noted from the scatter plots, there are a handful of outliers. However, it is a relatively low number for this dataset size. Again, the outliers are left in the analysis for the reasons discussed above. The -9 dB label displays a noticeable separation of the median and interquartile range when compared to the other labels except in the Δt_{cd} (Delta.t.Bottom) feature. This indicates that classification of -9 dB Vs. the 2 other labels may be relatively easy, but classification between -1 dB and no interference labels may be difficult. This observation is not surprising since the -9 dB case has significantly higher interference and should cause more data deviations from the no interference case.

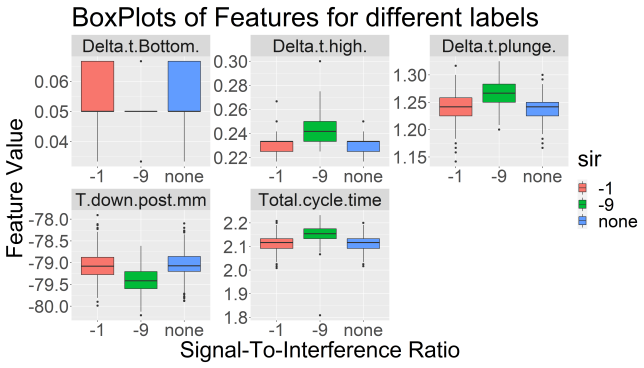


Figure 4. Box plots of all features for each classification label

A breakdown of the number of training and testing observations for each segment size, M, can be seen in Table 1. Each label is represented by an even split of the total values shown in the table. It is important to point out that the numbers for the training and testing set size represent the number of segments. Although this number decreases as the segment size increases, the actual size of the training matrix increases as $M * n$, where n is the number of segments and M is the segment size. The total training matrix size is also included for reference.

M	training	testing	Training Matrix
5	1515	999	7575
50	1497	855	74850
100	1479	693	14790
150	1461	531	219150
200	1443	369	288600

Table 1. Total number of observations for the training and testing data sets for each M and the total training matrix size.

4 Methodology

4.1 Models

The R statistical software [14] is used for all the analyses performed in this work. Four models are investigated: K-Nearest Neighbors (KNN), Linear Discriminate Analysis (LDA), Single Hidden Layer feed-forward Neural Network (NN), and a Support Vector Machine (SVM). A brief discussion of each model, the rationale for choosing it as a candidate, and its implementation are provided below.

4.1.1 KNN. The KNN classifier is a non-parametric supervised learning algorithm in which the classification of a new observation depends only on the majority count of its K-nearest neighbors. The K-nearest neighbors are determined using a distance measurement, which in this case was the Euclidean distance. Features are scaled for the KNN classifier to eliminate the effects of different scales for each feature. This model is chosen for its ease of implementation, simple interpretation, and good performance on classification problems that are linear in euclidean space [8].

The KNN model is built using the *caret* implementation of the *knn* function from the *class* library in R [18] [10]. K-values in the range of $5 - 33$ are explored to determine which value gives the best performance as measured by the accuracy metric with 5-fold cross-validation. The final model is used to make predictions with the testing set.

4.1.2 Linear Discriminate Analysis. LDA is a supervised linear transformation technique often used for dimensionality reduction, classification, and data visualization. The goal of LDA is to find the projections of the data onto component axes which maximize both the variance of the data and separation between class labels and to then use the linear discriminant analysis to assign labels [5] [15]. Implementation of LDA is often used as a benchmarking method because of its ease of implementation, low computational costs, and

relatively good performance. This method is also considered for this work due to the transformation properties of LDA, which may help to mitigate issues caused by the highly correlated data in the segments.

The LDA model is built using the *lda* function from the *MASS* library in R [19]. All LDA projections are included in the model, and no tuning of parameters is performed. The trained model is used to make predictions with the testing set.

4.1.3 Neural Network. The single hidden layer feed forward neural network is a non-linear artificial neural network consisting of 3 layers of computational units (nodes) connected in a feed forward manner. The layers of this NN include an input layer, hidden layer, and output layer. The input layer consists of n units, where n is the number of input features in the dataset. The input layer feeds these inputs into the hidden layers perceptrons and multiplies the inputs by learned weights. The resulting value is then "activated" using a non-linear function such as the Relu function. The resulting values are passed to the output layer, which uses the output of all the previous layers to calculate a class probability via a softmax activation function. In this case, there are 3 units in the output layer for the 3 class labels.

Neural networks such as these are known for their ability to accurately represent non-linear data and their high performance on classification tasks when lots of data is available. The NN method was chosen as a candidate in this work to determine if the flexibility provided by NN would lead to better performance and to understand if these benefits are held for low data (small segment size) situations.

The NN model is built using the *caret* implementation of the *nnet* function from the *nnet* library in R [20] [10]. For this study, the hidden layer number of units and decay rate are tuned using a grid search methodology, and the accuracy metric as computed using 5-fold cross-validation. The number of hidden layers units used for tuning were: 2, 3, 4, 5, 6, 7, 8, 9, 10. The values of decay rate were: 0.00316, 0.01, 0.0316, 0.100 0.316 1.00.

4.1.4 Support Vector Machine. The SVM is a supervised learning method that works well with a small number of observations and is comparatively fast. The SVM works by maximizing the margin of hyperplanes used to separate linearly separable data. The linear SVM can be trivially extended to a non-linear case by use of kernels, and the kernel trick [8]. Furthermore, non-separable data can be classified with the SVM by allowing misclassification to happen and introducing a slack variable into the primal form of the optimization problem. A regularization parameter, C, is used to balance the trade-off between maximizing the margin and minimizing the miss-classification loss. This method was chosen for evaluation due to its relatively low computational costs, ability to fit linear and non-linear boundaries and good performance on small datasets.

The SVM models are built using the *svmLinear*, *svmRadial*, and *svmPoly* functions in the *kernlab* library in R [9]. Each function represents a SVM model utilizing a different kernel. The hyperparameters for each kernel are tuned using the accuracy metric and 5-fold cross validation. The values of the hyper parameters for each model are chosen automatically by using the *tuneLength* option in *caret*. Table 2 lists the tune length and variables for each model.

Table 2. Tuning hyper parameters and tune length for each SVM kernel model

Kernel	Tune Length	Parameters
Linear	20	C
Radial	4	C, scale, degree
Poly	10	C, sigma

4.2 Metrics

Classification problems, such as the one in this work involving more than two classes, are known as multi-class classification problems. Understanding and choosing appropriate performance metrics is crucial to selecting a model that will perform well according to a given use case scenario. In this work, it is imperative to correctly detect the presence of interference to prevent costly damage to equipment, material, and personnel. However, high false-positive rates can lead to unnecessary costly shutdowns or operator conditioning which may lead to ignoring warnings from the algorithm. Therefore, the chosen model needs to balance the trade-off between true positives and false positives.

Numerous metrics can be used to evaluate the performance of a model in a multi-class classification problem [6]. Considering this application's trade-offs discussed above, the Multi-Class Area Under the ROC Curve (AUC) is used as the ultimate model metric as described by Hand and Till, where AUC ranges between 0 and 1 and a higher value is more desirable [7]. This value is determined using the *multiclass.roc* function from the *pROC* library in R [16]. In addition, the models' confusion matrix and balanced accuracy, computed using the *confusionMatrix* function in R, is also used to get a better-detailed understanding of how each model performed on each label [10].

5 Analysis and Results

5.1 KNN

The KNN model is trained on the training datasets for all M values. The best performing K-values as determined by 5-fold cross-validation are shown in Table 3 and range from values 5-13 depending on the segment size. The AUC as calculated on the testing data set for each M is also shown in Table 3. As shown in the table, the AUC values for M5-M100 are around 0.50. This is only marginally better than

random guessing, which is equivalent to an AUC of 0.50. As M increases beyond $M = 100$, the scores increase to values above 0.60.

	M5	M50	M100	M150	M200
K-Value	7	13	9	9	5
Test AUC	0.51	0.53	0.54	0.68	0.67

Table 3. Best KNN K-value and AUC For Different M

Despite using AUC as the primary metric to evaluate the performance of the models, it does not elucidate the finer details of how the model is performing. In order to study the performance in more detail, a multi-class confusion matrix and the balanced accuracy for each label of the KNN model for $M = 150$ is shown in Table 4. The $M = 150$ was chosen to enable performance comparison between different models close to each model's peak performance. From the confusion matrix and balanced accuracy results, we can see that despite relatively low AUC values, the model was able to identify the -9 dB case with 100% accuracy and no false positives. However, it was unable to distinguish between the -1 dB case and no interference effectively. This aligns with the observation in the data exploration that separating the -1 dB, and no interference cases may be difficult using the features available.

		predicted		
		-1	-9	none
Truth	-1	2	0	11
	-9	0	177	0
	none	175	0	156
Bal. Accuracy		0.49	1.0	0.71

Table 4. Confusion Matrix and Balanced Accuracy for KNN Model with $M = 150$

5.2 LDA

The LDA model is trained on the training datasets for all M values, and the resulting model was used to predict the labels of the testing data set. The AUC as calculated on the testing data set for each M is shown in Table 5. The AUC value for $M = 5$ is 0.53, which is similarly poor in performance compared to the KNN model. However, as M increases, substantial improvements to the AUC are noted. LDA has a maximum AUC of 0.80 when segment size $M = 150$. A small decrease in performance is noted for $M = 200$.

	M5	M50	M100	M150	M200
Test AUC	0.52	0.68	0.73	0.80	0.77

Table 5. LDA Test AUC For Different M

The confusion matrix and the balanced accuracy for each label of the LDA model for $M = 150$ are shown in Table 6. The LDA model correctly labels the -9 dB case with 100% accuracy and no false positives. However, it also had difficulty distinguishing between the -1 dB case and no interference, but it performs marginally better than the KNN model.

		predicted		
		-1	-9	none
Truth	-1	177	0	118
	-9	0	177	0
	none	0	0	59
Bal. Accuracy		0.83	1.0	0.67

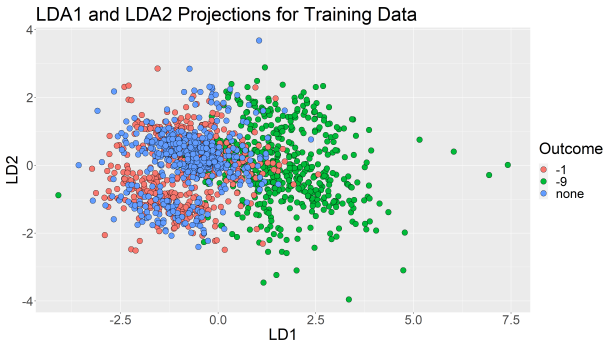
Table 6. Confusion Matrix and Balanced Accuracy for LDA Model with $M = 150$

One interesting outcome of using LDA is that one can visualize the labels projected onto lower-dimensional space. Plots of the first and second LDA components with the data points colored by labels are shown in Figure 5 for $m = 5$ and $m = 150$. The difference between the $m = 5$ and $m = 150$ projections is striking and explains the significant increase in AUC for the $m = 150$ case. For $M = 5$, the components are barely separated, contributing to the inability of the model to classify barely better than random guessing. However, the $m = 150$ case shows a clear linear separation between -9 dB and the other 2 labels. The other 2 labels, although not linearly separable, are better separated with less overlap between the labels.

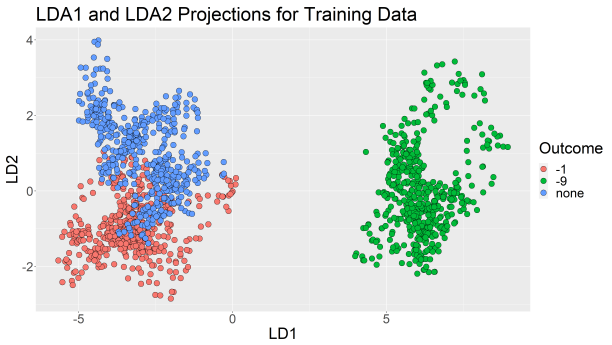
For reference, the LDA predicted test case projected onto the first 2 components is provided in Figure 10 in the appendix. The linear boundaries, constructed on the training data shown above, are evident in these figures and help us visually understand the performance of the LDA model.

5.3 Neural Network

The NN model is trained on the training datasets for all M values. The best performing combination of the number of hidden layer units and learning rate as determined by 5-fold cross-validation are shown in Table 7 for all M values. The AUC as calculated on the testing data set for each M is also shown in Table 7. The AUC values for M5 is 0.54, which is only slightly better than the previous models. The AUC increases significantly up to $M = 150$ and then experiences a slight decrease in performance. The NN has a maximum AUC of 0.89 when the segment size is $M = 150$.



(a) LDA1 and LDA2 of training data $M = 5$



(b) LDA1 and LDA2 of training data $M = 150$

Figure 5. LDA1 and LDA2 Components of Training Data

	M5	M50	M100	M150	M200
Hidden Units	9	2	2	2	2
Learning Rate	0.00316	0.316	0.316	0.316	0.1
Test AUC	0.55	0.74	0.84	0.89	0.88

Table 7. Best NN Tuned Hyper-parameters and AUC For Different M

The confusion matrix and the balanced accuracy for each label of the NN model for $M = 150$ is shown in Table 8. The NN model correctly labels all of the -9 dB cases with 1 false positive. On the other hand, it does a significantly better job of separating the -1 dB and non-interference cases. This is reflected in the high balanced accuracy of all labels.

		predicted		
		-1	-9	none
Truth	-1	158	0	19
	-9	0	177	1
	none	19	0	157
Bal. Accuracy		0.92	1.0	0.92

Table 8. Confusion Matrix and Balanced Accuracy for NN Model with $M = 150$

5.4 Support Vector Machine

The SVM model is trained on the training datasets for all M values. The best performing combination of kernel and kernel-specific hyper-parameters as determined by 5-fold cross-validation are shown in Table 9 for all M values. The AUC as calculated on the testing data set for each M is also shown in Table 9. The AUC value for $M = 5$ is 0.54, the same as the NN model. The AUC increases significantly up to $M = 200$ with an AUC of 0.92, the highest of all models and M values. It should be noted that the high AUC numbers were achieved with relatively non-complex SVM models and, unlike all other models, continued to see gains in performance for M values greater than 150.

	M5	M50	M100	M150	M200
Kernel	Linear	Linear	Poly	Linear	Poly
C Value	1.6	1.9	2.0	2.0	0.5
Scale	-	-	1.0	-	1.0
degree	-	-	1.0	-	1.0
Test AUC	0.54	0.75	0.84	0.90	0.92

Table 9. Best SVM Tuned Hyper-parameters and AUC For Different M

The confusion matrix and the balanced accuracy for each label of the SVM model for $M = 150$ are shown in Table 10. The SVM model correctly labels all of the -9 dB cases with no false positives. It also does a significantly better job separating the -1 dB and non-interference cases than the KNN and LDA models and slightly better than the NN model. This is reflected in the high balanced accuracy of all labels.

		predicted		
		-1	-9	none
Truth	-1	156	0	16
	-9	0	177	0
	none	21	0	161
Bal. Accuracy		0.92	1.0	0.93

Table 10. Confusion Matrix and Balanced Accuracy for SVM Model with $M = 150$

5.5 Combined Comparison

The AUC score for all models is shown in Figure 6. There are several takeaways that one can infer from the picture. First, all models perform almost equally poorly when data segmentation is not used, and the performance of the models generally increases as segment length M increases. All models except the SVM show a slight drop in performance at $M = 200$. This is likely due to a decrease in the testing set size and may not indicate the evaluated performance if more data were available for the test set.

It is also apparent that the relatively simple versions of the SVM perform consistently the best, although the NN model is only slightly worse. The LDA model performed worse than the previous 2 models, but not bad considering its simplicity and lack of hyper-parameter tuning. The KNN model performed the worse at all points and only realized a modest increase in AUC with increasing M.

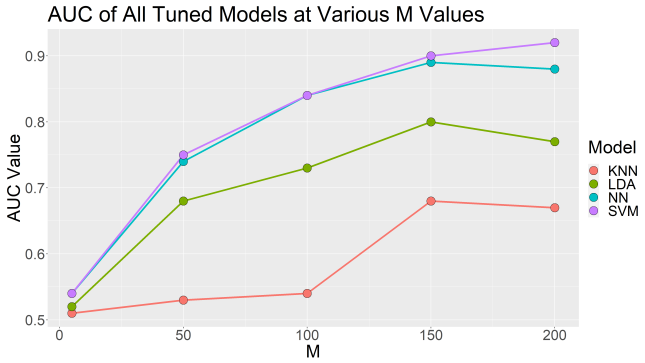


Figure 6. The AUC of all models for All M values

Not mentioned until now is that all models at all M values did an excellent job of separating the high interference cases from the no-interference and low interference cases. If the goal is a binary classification that only distinguishes between high and low/no interference in order to alert an operator when dangerous interference is present, all these models performed admirably. The simplest model, likely the LDA, should be a fast and sufficient model for detection in this case. However, this is not true if the goal is to distinguish between different interference levels.

6 Conclusions

In this work, multiple data mining and machine learning methods are used to classify the level of interference present in data collected from a practical use case of an industrial wireless network system of a robot performing a repetitive task. 5 features engineered from raw time-series data of 3 interference levels were used in the analysis. Segments consisting of varying lengths of time-sequential repetitions of the data were constructed to investigate the sensitivity of each model's performance to the length of time that data is sampled. KNN, LDA, NN, and SVM models were trained and tuned for each segment length on a training data set and evaluated on a separate testing data set. AUC, balanced accuracy, and confusion matrices are used to evaluate the performance of the models.

This study found that none of the models performed well when segment size M corresponded to 1 repetition of the robots movement. As segment size increased, all models improved performance up to a segment size of $M = 150$. The SVM was the only model to show improvement in performance at segment sizes greater than $M = 150$. The NN and SVM models significantly outperformed the other models at all levels, with the SVM model displaying the best performance. Inspection of the confusion matrices and balanced accuracy revealed that all models did a reasonable job separating the no interference and low interference from the high interference cases in a binary manner. However, an additional separation between the −1 dB and no interference cases proved difficult without the proper choice of model, SVM or NN, and higher segment lengths.

In this test, LDA can be used to help separate classes by projecting their higher dimensional representations into lower dimensional space while maximizing both the variance of the data and the separation between class labels. In future work, the LDA may be used as a dimensionality reduction technique in conjunction with the other models, such as the NN and SVM, to increase performance and reduce computational costs. Additional future work may also include creating stacked ensemble models comprised of each of the current models to take advantage of their various predictive strengths.

6.1 Lessons Learned

There are three major lessons learned from this project/class regarding data mining:

1. I think it is essential to introduce data sets and applications that are not as simple, clean, or with solutions that are not as cut and dried as the toy data sets presented in this class and most classes in the data analytics program. I have never had a project for class or project for real-world work where things were as straightforward as is often suggested by homework or examples. Understanding the actual data, cleaning and manipulating the datasets, and feature engineering seem to be a considerable part of the analysis process that often gets breezed over. Two of the proposals I peer-reviewed for this class were analyzing the "wine" dataset, which has been done for thousands of examples. I don't think much would be learned with a project on something as simple as that unless maybe for developing and testing a new type of algorithm or model.
2. I think that requiring students to do some sought of real literature search would be beneficial. I learned a lot about different techniques and cutting-edge models and algorithms by taking advantage of Georgia Techs' access to published journals and papers. For example, I had never even heard of creating segments for engineered time-series data like what is used in this report. Through my experience working with project teams in other classes, not a single person knew how to access papers through the VPN or GT library. I think it would be a very beneficial skill to learn.

3. I personally would like to have a little more discussion on what models/techniques are good for different types of situations and why. We learn so many different techniques and models with only a broad idea of what to use them for. Currently, the lessons only seems to say this is a clustering model, or this is a classification model, or a regression model. I know there are no steadfast rules for application, but something like "This model would be a good choice for a classification problem with highly correlated data because ...", would be appreciated.

References

[1] Rick Candell. 2019. Measurement Data for a Wireless Force Seeking Apparatus, National Institute of Standards and Technology. (2019). https://doi.org/10.18434/M32077

[2] Richard Candell, Mohamed T Hany, Kang B Lee, Yongkang Liu, Jeanne T Quimby, Catherine A Remley, et al. 2018. Guide to industrial wireless systems deployments. (2018).

[3] Richard Candell, Karl Montgomery, Mohamed Kashef, Yongkang Liu, and Sebti Foufou. 2019. Wireless interference estimation using machine learning in a robotic force-seeking scenario. In *2019 IEEE 28th International Symposium on Industrial Electronics (ISIE)*. IEEE, 1334–1341.

[4] Tapiwa M Chiwewe, Colman F Mbuya, and Gerhard P Hancke. 2015. Using cognitive radio for interference-resistant industrial wireless sensor networks: An overview. *IEEE Transactions on Industrial Informatics* 11, 6 (2015), 1466–1481.

[5] RA Fisher. 1950. The use of multiple measurements in taxonomic problems, Annual Eugenics, 7, Part II, 179-188 (1936); also in Contributions to Mathematical Statistics.

[6] Margherita Grandini, Enrico Bagli, and Giorgio Visani. 2020. Metrics for multi-class classification: an overview. *arXiv preprint arXiv:2008.05756* (2020).

[7] David J Hand and Robert J Till. 2001. A simple generalisation of the area under the ROC curve for multiple class classification problems. *Machine learning* 45, 2 (2001), 171–186.

[8] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2001. The elements of statistical learning. Springer series in statistics. *New York, NY, USA* (2001).

[9] Alexandros Karatzoglou, Alex Smola, Kurt Hornik, and Achim Zeileis. 2004. kernlab – An S4 Package for Kernel Methods in R. *Journal of Statistical Software* 11, 9 (2004), 1–20. http://www.jstatsoft.org/v11/i09/

[10] Max Kuhn. 2021. *caret: Classification and Regression Training*. https://CRAN.R-project.org/package=caret R package version 6.0-88.

[11] Lucia Lo Bello, Johan Åkerberg, Mikael Gidlund, and Elisabeth Uhlemann. 2017. Guest editorial special section on new perspectives on wireless communications in automation: From industrial monitoring and control to cyber-physical systems. *IEEE Transactions on Industrial Informatics* 13, 3 (2017), 1393–1397.

[12] Zhibo Pang, Michele Luvisotto, and Dacfey Dzung. 2017. Wireless high-performance communications: The challenges and opportunities of a new target. *IEEE Industrial Electronics Magazine* 11, 3 (2017), 20–25.

[13] Pti. 2020. Industries in rural areas allowed to operate from April 20. https://www.rediff.com/money/report/industries-in-rural-areas-can-operate-from-april-20/20200415.htm

[14] R Core Team. 2021. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/

[15] C Radhakrishna Rao. 1948. The utilization of multiple measurements in problems of biological classification. *Journal of the Royal Statistical Society. Series B (Methodological)* 10, 2 (1948), 159–203.

[16] Xavier Robin, Natacha Turck, Alexandre Hainard, Natalia Tiberti, Frédérique Lisacek, Jean-Charles Sanchez, and Markus Müller. 2011. pROC: an open-source package for R and S+ to analyze and compare ROC curves. *BMC Bioinformatics* 12 (2011), 77.

[17] Emiliano Sisinni, Abusayeed Saifullah, Song Han, Ulf Jennehag, and Mikael Gidlund. 2018. Industrial internet of things: Challenges, opportunities, and directions. *IEEE transactions on industrial informatics* 14, 11 (2018), 4724–4734.

[18] W. N. Venables and B. D. Ripley. 2002. *Modern Applied Statistics with S* (fourth ed.). Springer, New York. https://www.stats.ox.ac.uk/pub/MASS4/ ISBN 0-387-95457-0.

[19] W. N. Venables and B. D. Ripley. 2002. *Modern Applied Statistics with S* (fourth ed.). Springer, New York. https://www.stats.ox.ac.uk/pub/MASS4/ ISBN 0-387-95457-0.

[20] W. N. Venables and B. D. Ripley. 2002. *Modern Applied Statistics with S* (fourth ed.). Springer, New York. https://www.stats.ox.ac.uk/pub/MASS4/ ISBN 0-387-95457-0.

A appendix

A.1 Exploratory Data Analysis

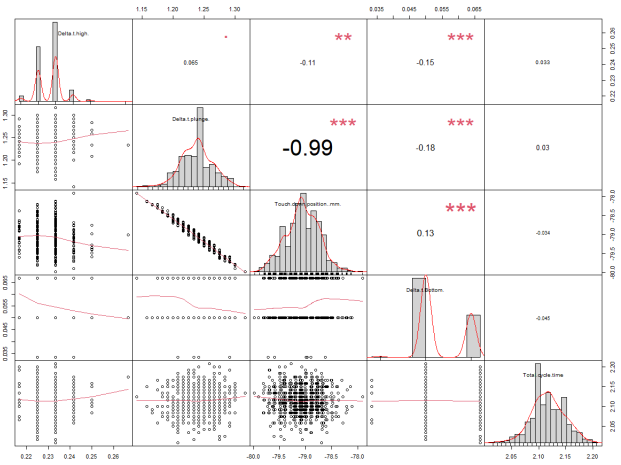


Figure 7. Correlation, histograms, and scatter plots for all features at −1 db.

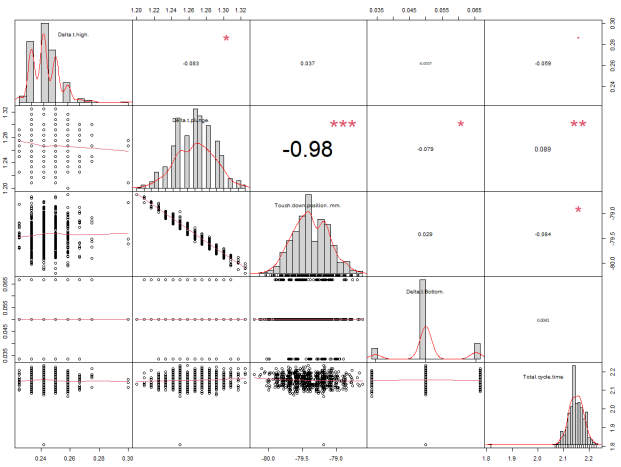


Figure 8. Correlation, histograms, and scatter plots for all features at −9 db.

[!htbp]

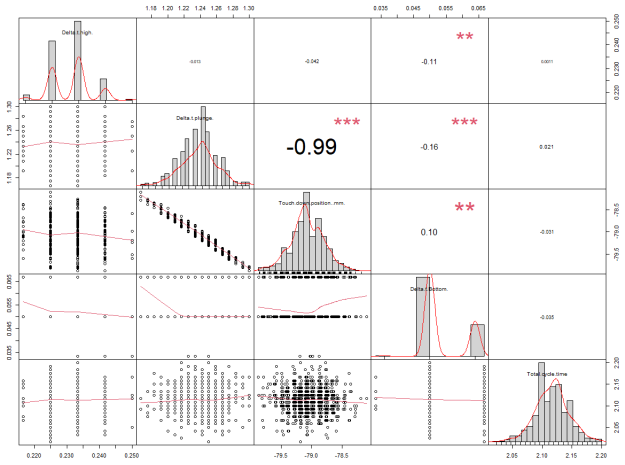
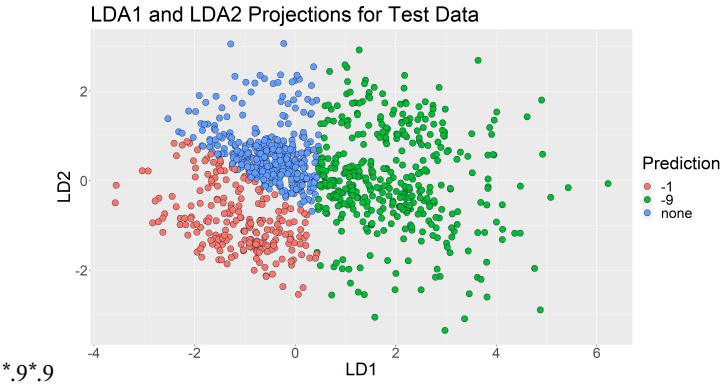
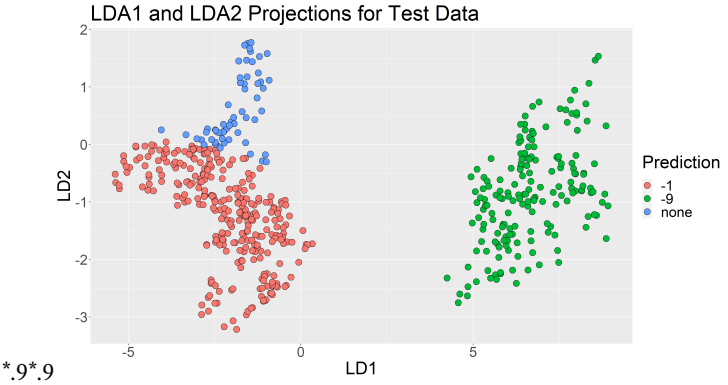


Figure 9. Correlation, histograms, and scatter plots for all features with no interference.

A.2 LDA Results



(a) LDA1 and LDA2 of testing Prediction $M = 5$



(b) LDA1 and LDA2 of testing Prediction $M = 150$

Figure 10. LDA1 and LDA2 Components on Predicted Data