Activity Recognition for Automatic Music Selection

Bryan P. Beatrez, Jordan M. O'Connor, *Rochester Institute of Technology*
December 15, 2014

***Abstract*—Much research has been done in automatic classification, but not music selection based on environmental factors. This project used a Bayesian Network Classifier, a Decision Tree Classifier, and a Support Vector Machine to recognize activity. RGB D data from twelve different activities each with four different people was used. The algorithms tested whether the activity given was a certain activity or it was not that activity. This process was repeated for each activity. Our results show that the algorithms did an extremely good job of recognizing the activities. Our results also show that the Bayesian Network Classifier performed significantly worse than the other two algorithms for certain activities. However, our results also show that the algorithms sometimes focused on the wrong information, such as a single joint, to identify an activity.**

I. Introduction

A common problem in public or private events is selecting music that is appropriate for the type of setting. When hosting such an event, it is imperative to select appropriate music as to not interfere with the mood. Selecting this music can also be time consuming and, thus, annoying for the host who wishes to enjoy the gathering. This task can also be thankless if the guests do not appreciate the music The Auto Music Selection project seeks to solve this issue

This project includes two parts. First, based on video data, the environment must be classified. Second, based on this classified environment, appropriate music will be selected automatically. The music selection will automatically adapt if the environment changes. However, it is important to note that in the scope of this specific project, only the first part will be completed, classifying the environment.

Much research has been conducted in developing algorithms to automatically characterize music by components such as beat, rhythm, and instrumentation. However, very minimal research has been done on recommending music based on the environment. The goal of this project is to begin to bridge this knowledge gap [1] [2].

The plan for this project is to create an intelligent algorithm that will be able to recognize activity based on video data alone. The project will allow for a deeper understanding of the image processing needed to determine an environment.

The necessary input data for this project is videos demonstrating different characteristics activities. Some of these activities should be contrasting to ensure accuracy of the algorithm. The output data of this project will be a description of the activity. The intent is that this description could be later used to select songs that are appropriate to the recognized activity.

The major objectives of this project are as follows:

- We develop the baseline algorithm which will be a Bayesian Network Classifier (BNC).
- We develop the first comparison algorithm which will be a Decision Tree Classifier (DTC).
- We develop the second comparison algorithm which will be a Support Vector Machine (SVM).
- The results of the two comparison algorithms will be compared to the baseline algorithm to gauge the performance of the three algorithms. Tables and figures will be produced to summarize our findings.

II. Methods

A. *Bayesian Network Classifier (BNC)*

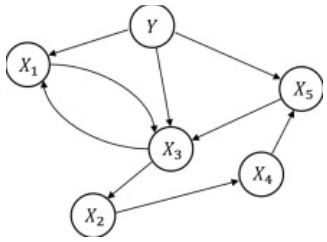

Figure 1. Bayesian Network Classifier

Many different algorithms exist for types of Bayesian network classifiers. A Naïve-Bayes lets the classification node be the parent of all other nodes and then learns the parameters. Other Bayesian networks include the Tree Augmented Naïve-Bayes, a Bayesian network augmented Naïve-Bayes, and a general Bayesian network [13].

A naïve Bayesian classifier which has strong assumptions of independence is both simple and effective. A Tree Augmented Naïve Bayes can outperform the Bayes, but also maintain low computational power meaning there is no search required. By inducing classifiers, the main concern of assuming independence that is clearly not is mitigated [14].

A computationally efficient method for creating selective Bayesian network classifiers involves using information-theoretic metrics. This learning occurs in two steps. First, node selection takes place using an information-theoretic metric. Second, network construction occurs where the network is constructed from the subset of attributes which were selected by the first phase [15].

While Bayesian networks are powerful in representing probabilities, they tend to perform poorly when learned in the standard way. Increasing accuracy while minimizing computational power can be accomplished by a simple approximation. This approximation maximizes the

conditional likelihood and sets the parameters by the maximum likelihood [16].

We will use the built-in MATLAB function fitNaiveBayes to train and the predict function to predict the activity. The MATLAB function assigns a new observation to the most probable class, assuming the features are conditionally independent. While the features are not independent in our implementation, we will nevertheless use this function. These functions are a part of the statistics toolbox for MATLAB.

B. Decision Tree Classifier (DTC)

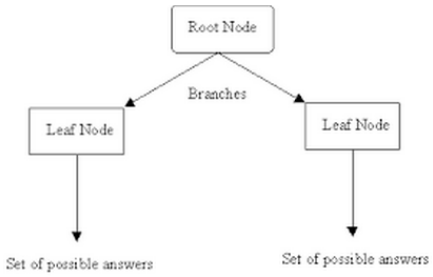

Figure 2. Bayesian Network Classifier

Decision Tree Classifiers (DTCs) have many practical applications in diverse areas. The major takeaway from these DTCs is that it can break down a complex decision into many smaller, less complex decisions. They are also efficient as a sample is tested only against certain smaller classes which reduces the number of unnecessary computations made. The goal is to make as simple of a tree as possible [10].

The design of a decision stages is very important to ensure the best possible performance of the decision tree. Two approaches for the design of decision trees are reported: the manual design and toward optimized design tree design. The manual design requires collecting statistics for all the classes and producing a graph where the means and variances are plotted for all of these classes. However, it is only possible to achieve an optimal design in an extremely simple case. The second method uses a heuristic search which is described as a "guided search with forward pruning" [11].

Decision tree classifiers are useful for several reasons including being easily interpreted by humans and being efficient and, thus, applicable to large sets of training data. PUBLIC is an improved decision tree which has two steps. First, it builds the tree. Second, it prunes the tree. This algorithm is set apart from other similar algorithms because it will not expand a node if it determines that it will be next pruning phase which, in turn, reduces overfitting [12].

Classification with large datasets has many practical applications. However, processing time and memory suffer heavily when utilizing such large datasets. Methods have been introduced to help and solve these problems such as discretization and sampling but these methods cause a significant reduction in accuracy. CLOUDS samples the splitting points and then estimates to narrow the search space. RGBD data can be very large making processing time and memory consumption large [13].

We will use the built-in MATLAB function fitctree to train and the predict function to predict the activity. The MATLAB function create a binary tree based on the input variables where each branching node is split based on the values in the first column. It then predicts the activity based on this produced binary tree. These functions are located in the statistics toolbox for MATLAB.

C. Support Vector Machine (SVM)

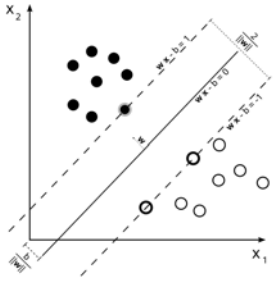

Figure 3. Support Vector Machine

A support vector machine works to divide the dataset into separate, distinct regions which can be separated by a line. This line it fit so that it is the maximum possible distance between the two sets of data. After training with some data, the placement of new data points can be predicted based on where the data falls. This algorithm has many practical applications as it can be used to separate many datasets and has proven extremely effective [6].

An SVM uses a kernel mapping to map the data to a high-dimensional feature space where the data can become linearly separable. Then, based on the position of the data to be classified, a prediction can be made based on where the point lies. The SVM can be applied to both regression and pattern recognition. There are many kinds of kernels that can be used including the Gaussian and polynomial kernels. Some of these can improve performance in certain circumstances [7].

The support vector machine gained much popularity in the early 2000's. Before this time, the algorithm of choice was a neural network which are less susceptible to the problem of model mis-specification and are noise tolerant. The SVM can perform well given no local minima and sparse representation of the solution. The SVM is the same as solving a linearly constrained quadratic programming problem. As a result, its solution is unique and globally optimal [8].

There are many ways to fit a linear classifier in an SVM, including linear regression, Fisher's linear discriminant analysis, and logistic regression. Non-linear models can be used, but they become very complicated and can result in over-fitting. While in many situations, we can have sufficient variables to guarantee a separation, we may elect to avoid the maximum margin separator. Doing so will result in a more regularized solution involving more observations [9].

We will use the built-in MATLAB function svmtrain to train and the svmclassify model to make predictions. These functions are located in the statistics toolbox for MATLAB

III. EXPERIMENTS

The experiments conducted involved training three individual artificial intelligence algorithms with the same

data provided to them, and comparing the performance of each algorithm at classifying the data provided.

Cornell University's Robot Learning Lab is the source for our RGBD which is comprised of data from their CAD-60 package. This package includes the following: four subjects including two male, two female, and one left-handed, five different environments including an office, a kitchen, a bedroom, a bathroom, and a living room, and 12 different activities. These activities are included in our results in the subsequent sections. This lab conducted a study on human activity detection from RGBD images. This study performed detection and prediction of unstructured human activity in unstructured environments. The RGBD data is from the Microsoft Kinect [3] [4].

The code from The Robert Learning Lab is available on Cornell's website. The code is divided into two parts. First, the raw RGBD data is run through a C++ program which produces an output file containing the frame number, and details about the frame with numbers separated by commas. This file is then read by MATLAB which produces a visual showing the skeleton of the person with his/her joints indicated by dots. This output is a 3D which can be easily seen using the MATLAB graph figure tools [3] [4].

In order to give the three algorithms a proper dataset to train with, the pre-classified data was converted into a CSV file that contains the x, y, and z coordinates for 15 joints of 15 activities from the CAD-60 dataset, for four people. At 20 frames per activity, this put the experiment at 1200 classified frames of x, y, and z coordinates to train the algorithms. An example of a classified frame of joints is shown in Figure 4 and Figure 5.

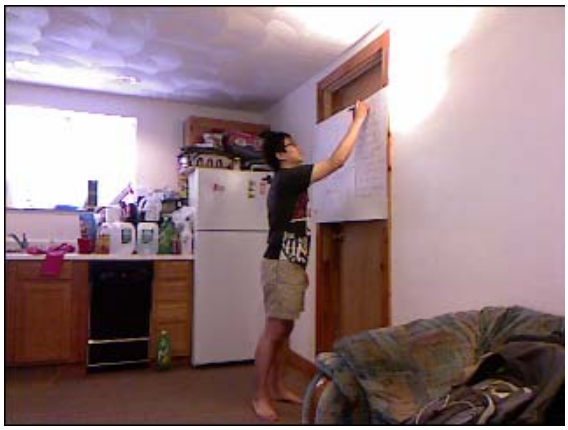

Figure 4. RGB Raw Image

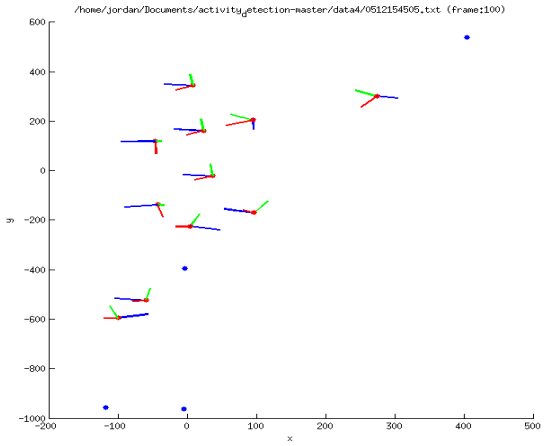

Figure 5. Generated Skeleton

The classified frame consists of one row in the CSV file, and it contains the x, y, and z coordinate of each of the 15 joints in the frame. Therefore, there are 45 data points of information in the classified activity. The joints are as follows: Head, Neck, Torso, Left Shoulder, Left Elbow, Right Shoulder, Right Elbow, Left Hip, Left Knee, Right Hip, Right Knee, Left Hand, Right Hand, Left Foot and Right Foot. In the experiment, before cross validation and training, a 46th column is concatenated to the dataset to provide the classification that the models will be trained with. There are 15 experiments conducted, each trying to train the algorithms to recognize the current action in a set of frames. Therefore, the majority of the time, the frames represent an action that is not the current action that is being inspected. The algorithms are trained with a limited number of positive classes, while the rest of the classes are considered not positive, and in this case zero. Since this experiment is designed around a binary classification, one or zero, its complexity is rather simple compared to multi-class classification datasets.

Several MATLAB scripts and custom functions have been used to compile the generated data, train the algorithms, and present the results in a way that was understandable. Each algorithm's results are presented using a confusion matrix, which displays the number of true positives, true negatives, false positives, and false negatives that were generated by the model trained. From this information, the precision, accuracy and recall of the algorithms can be generated to truly determine the performance. In the case of the DTC, a visual decision tree is generated for each iteration to show what joints played a crucial role in deciding the classification of a frame tested with the tree.

As the primary difficulty in this project was processing the RGB D data, we, naturally, ran into problems with doing so initially. At first, we wanted to collect our own data, but we realized that this was not reasonable. Luckily, we were able to locate Cornell University's Robotics Lab's website early on. The purpose of their project was almost identical to the first part of our project. As such, the website provided a comprehensive dataset as well as detailed documentation.

Through the early stages of our project, we struggled with how we wanted to present the training data to the algorithm. At first, we considered just using one axis of each joint and

simplified the activity recognition. Specifically, we were simply classifying if the person was sitting or standing. However, after working more with the functions that we used in MATLAB, we determined that it was feasible to present all three axes to the algorithm for learning. Based on the information from one person, the algorithm was able to determine for three other individuals if he/she was doing or if he/she was not doing the specified activity.

IV. Discussion

There are several different interesting results that have been observed from this experiment. Some pertain to the differences in the training dataset, while others pertain to the individual performance of the algorithms. The success of the experiment lies heavily in the preparation of the training data, and in the difficulties that it presents to the trained algorithm models. The performance metrics generated are the precision, accuracy, and recall of the algorithms. Other results include the decision tree for the DTC, and timing statistics for each algorithm.

A brief summary of our results is located in Figure 6 and Figure 7. These figures show a graph of the accuracy of each algorithm for each of the tested activities. We provide more detailed statistics later in this section. As shown, the Bayesian Network Classifier performed significantly worse than the Decision Tree Classifier and Support Vector Machine in certain activities. The activities where the Bayesian Network Classifier performed significantly worse were activities that closely resembled another activity. For example, putting in contacts and drinking water all require the person to lift one arm up pretty high. Another interesting note is that certain distinctive activities, such as working on the computer or whiteboard, our algorithms were able to work perfectly.

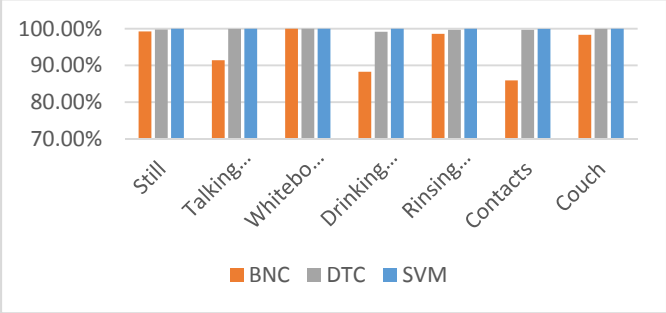

Figure 6. Summary of Accuracy

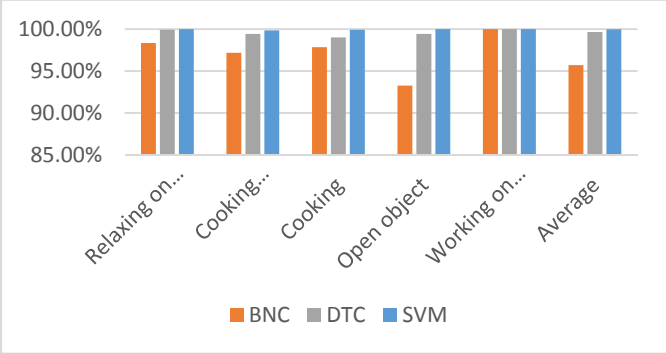

Figure 7. Summary of Accuracy

The statistical results of our algorithm performance is very high, as shown in TABLE I through TABLE III. The SVM performed with an accuracy of 99.97%, the DTC with 99.97% and the Naive Bayes with 95.69% average accuracy. The SVM was expected to perform the best, as it typically does based on its complexity. These numbers are high for a number of reasons, most of which stem from the fact that the actions performed in the classified datasets were too different. This made classification easy for the models, and they were very successful at determining differences. The only time error was seen in the results, is when the models were trying to classify an action that had other similar actions in the dataset. For example, brushing teeth is similar to talking on the phone, which is also similar to drinking a glass of water. But all of those are very different from sitting on the couch, and thus the models were very good about classifying the actions that were very different.

TABLE I. BNC Results

Action	Accuracy	Precision	Recall
Still	99.25%	99.20%	100.00%
Talking on the phone	91.42%	90.80%	100.00%
Whiteboard	100.00%	100.00%	100.00%
Drinking water	88.25%	87.41%	100.00%
Rinsing mouth with water	98.58%	98.48%	100.00%
Contacts	85.92%	83.75%	100.00%
Couch	98.33%	98.21%	100.00%
Relaxing on couch	98.33%	98.21%	100.00%
Cooking (stirring)	97.17%	96.96%	100.00%
Cooking	97.83%	97.68%	100.00%
Open object	93.25%	92.29%	97.08%
Working on computer	100.00%	100.00%	100.00%
Average	**95.69%**	**95.25%**	**99.76%**

TABLE II. DTC Results

Action	Accuracy	Precision	Recall
Still	99.75%	99.91%	97.50%
Talking on the phone	99.92%	99.91%	100.00%
Whiteboard	100.00%	100.00%	100.00%
Drinking water	99.17%	99.82%	90.00%
Rinsing mouth with water	99.67%	99.73%	98.75%
Contacts	99.67%	99.71%	99.38%
Couch	99.92%	100.00%	98.75%
Relaxing on couch	99.92%	100.00%	98.75%
Cooking (stirring)	99.42%	99.73%	95.00%
Cooking	99.00%	99.55%	91.25%
Open object	99.42%	99.58%	98.75%
Working on computer	100.00%	100.00%	100.00%
Average	**99.65%**	**99.83%**	**97.34%**

TABLE III. SVM Results

Action	Accuracy	Precision	Recall
Still	100.00%	100.00%	100.00%
Talking on the phone	100.00%	100.00%	100.00%
Whiteboard	100.00%	100.00%	100.00%
Drinking water	100.00%	100.00%	100.00%
Rinsing mouth with water	100.00%	100.00%	100.00%
Contacts	99.92%	100.00%	99.38%
Couch	100.00%	100.00%	100.00%
Relaxing on couch	100.00%	100.00%	100.00%
Cooking (stirring)	99.83%	100.00%	97.50%
Cooking	99.92%	100.00%	98.75%
Open object	100.00%	100.00%	100.00%
Working on computer	100.00%	100.00%	100.00%
Average	**99.97%**	**100.00%**	**99.64%**

The decision tree classifier provided interesting visuals for how it decided the classification of individual frames. In general. The DTC, in general, had more success when the decision tree was more complex. For example, Figure 8 and Figure 9 compare the resulting trees of cooking (stirring) and sitting at the computer. The cooking DTC depended only on the 45th data point, which is the z coordinate of the right foot. Not only does the 45th coordinate not matter much when clarifying stirring, the algorithm did not do as well, boasting 99.83% accuracy, its worst performance. This shows that the training dataset did not emphasize the right components of the data to the algorithms. Sitting at the computer, on the other hand, held 100% accuracy, and had a very complex decision tree to sift through.

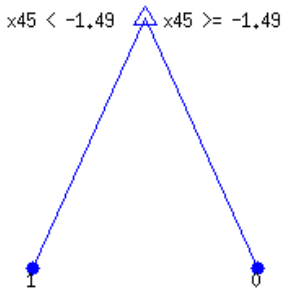

Figure 8. Simple DTC

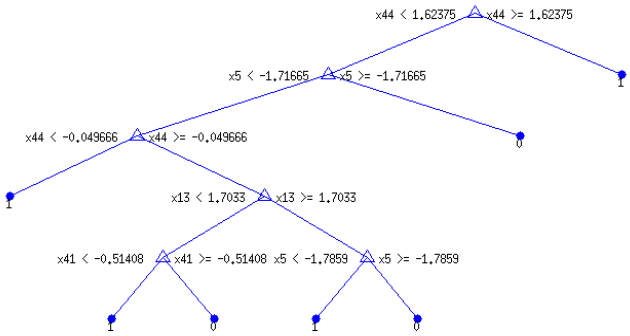

Figure 9. Complex DTC

The computation time of the algorithms proved to be consistent throughout the experiment. This characteristic is important for applying these algorithms in a real-time dataset, when decisions need to be made in a short amount of time. Overall, the SVM, though the most powerful and successful, performed the slowest with an average of 3 seconds per complete classification of the dataset. The amount of time is not as important, since this test was completed on a low-end processor, but rather the comparison to the other algorithms that stands out. The DTC performed approximately 10 times faster than the SVM, and the Naive Bayes classifier 100 times faster. Though the SVM was more successful with classification, it sacrifices a lot in the scope of how much computation time it consumes. Our complete timing results are shown in TABLE IV.

TABLE IV. TIMING RESULTS (IN SECONDS)

Activity	BNC	DTC	SVM
Still	0.0663	0.222	1.18
Talking on the phone	0.0694	0.270	1.17
Whiteboard	0.0753	0.221	1.15
Drinking water	0.0669	0.256	1.23
Rinsing mouth with water	0.0674	0.254	1.21
Contacts	0.0751	0.261	1.35
Couch	0.0641	0.221	1.12
Relaxing on couch	0.0638	0.191	1.07
Cooking (stirring)	0.0648	0.227	1.14
Cooking	0.0633	0.224	1.21
Open object	0.0636	0.300	1.33
Working on computer	0.0652	0.192	1.04
Average	**0.0671**	**0.237**	**1.18**

The timing results proved interesting in reference to the DTC. The timing characteristics of the DTC had a direct correlation to the complexity of the decision tree that was used to classify the test data. For example, classifying opening an object took 30% longer than classifying working on the computer. It is important to note, however, that the timing data is only meaningful when compared to other timing data run on the same computer as execution times will vary greatly with different hardware configurations.

V. CONCLUSION

In conclusion, it is obvious that the dataset provided to the algorithms did not emphasize the correct components of the classified frames. This error plays a large part in why the results of our accuracy, precision and recall were so high. In a real system, and in Cornell's comprehensive results, the highest precision and recall were 93.8% and 94.5% respectively. The complexity of our experiment is minimal compared to those results, which raises a red flag about the validity of our results [3] [4].

Despite these drawbacks, our algorithms still performed very well, despite using multiple individuals. These results prove that normalizing the positions of the joints worked. We were able to create a more dynamic intelligent algorithm that has many different practical applications.

The primary difficulty in the activity recognition part of this project was processing the RGB D data. A future improvement of our project could be completing the second part of this project which is selecting music based on a recognized activity. The challenging part in implementation would be understanding a user's preferences.

Another future improvement to our project would be not simply doing binary classification, where we are comparing doing versus not doing an activity. Doing so would entail using a multi-class network. Such a network would be very complex compared to the work which we performed. Another improvement could be using the video data instead of considering each frame individually. Doing so would make a more dynamic intelligent algorithm.

This project could also use the other data provided by Cornell University which is their CAD-120 dataset. This dataset also contains RGB D video files, but the activities are more complicated such as making cereal, stacking objects, microwaving food, etc.

The idea of incorporating a system that learns the actions of an environment and system that picks a music play list based on that environmental information, is a very forward-thinking and interesting application of the power of artificial intelligence.

REFERENCES

[1] McKinney, Martin F., and Jeroen Breebaart. "Features for audio and music classification." *ISMIR*. Vol. 3. 2003.

[2] Xu, Changsheng, M. C. Maddage, and Xi Shao. "Automatic music classification and summarization." *Speech and Audio Processing, IEEE Transactions on* 13.3 (2005): 441-450.

[3] Learning Human Activities and Object Affordances from RGB-D Videos, Hema S Koppula, Rudhir Gupta, Ashutosh Saxena. *International Journal of Robotics Research (IJRR)*, in press, Jan 2013.

[4] Unstructured Human Activity Detection from RGBD Images, Jaeyong Sung, Colin Ponce, Bart Selman, Ashutosh Saxena.*International Conference on Robotics and Automation (ICRA)*, 2012.

[5] Spinello, Luciano, and Kai Oliver Arras. "People detection in rgb-d data."*Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011.

[6] Tong, Simon, and Daphne Koller. "Support vector machine active learning with applications to text classification." *The Journal of Machine Learning Research*2 (2002): 45-66.

[7] Zhang, Li, Weida Zhou, and Licheng Jiao. "Wavelet support vector machine."*IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*34.1 (2004): 34-39.

[8] Cao, Li-Juan, and Francis Eng Hock Tay. "Support vector machine with adaptive parameters in financial time series forecasting." *Neural Networks, IEEE Transactions on* 14.6 (2003): 1506-1518.

[9] Hastie, Trevor, et al. "The entire regularization path for the support vector machine." *Journal of Machine Learning Research*. 2004.

[10] Safavian, S. Rasoul, and David Landgrebe. "A survey of decision tree classifier methodology." *IEEE transactions on systems, man, and cybernetics* 21.3 (1991): 660-674.

[11] Swain, Philip H., and Hans Hauska. "The decision tree classifier: Design and potential." *Geoscience Electronics, IEEE Transactions on* 15.3 (1977): 142-147.

[12] Rastogi, Rajeev, and Kyuseok Shim. "PUBLIC: a decision tree classifier that integrates building and pruning." *VLDB*. Vol. 98. 1998.

[13] Ranka, Sanjay, and V. Singh. "CLOUDS: A decision tree classifier for large datasets." *Knowledge discovery and data mining* (1998): 2-8.

[14] Grossman, Daniel, and Pedro Domingos. "Learning Bayesian network classifiers by maximizing conditional likelihood." *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004.

[15] Friedman, Nir, Dan Geiger, and Moises Goldszmidt. "Bayesian network classifiers." *Machine learning* 29.2-3 (1997): 131-163.

[16] Singh, Moninder, and Gregory M. Provan. "Efficient learning of selective Bayesian network classifiers." (1995).