Spectral Classification of Stars By Spectrophotometry

Madison Elizabeth Flaherty

Department of Computer Science
Rochester Institute of Technology

Abstract

Multiple approaches to classifying stars using detailed spectrophotometric wavelengths procured from the SLOAN Digital Sky Survey is presented with an emphasis on teaching the process such that another interested party can recreate the experiment. In this paper, great detail is spent on the collection and preprocessing of the data due to complexity of the raw data. Specifically, Principal Component Analysis is discussed indepth in respect to processing the data directly with Weka using both the GUI and command line interface. Once the training and testing data is processed, the training data is used to generate models for classification through the use of an artificial neural net and decision tree with Gini indexing. An attempt at classification is also seen through the k-Nearest Neighbors algorithm. These classification and clustering models are then compared according to accuracy and their confusion matrices.

I. INTRODUCTION

The classification of stars is -- even today -- a task which is far too often left to humans to manually perform. This paper hopes to further the understanding of classification of raw spectral data to further aid spectral scientists and analysts. The raw spectrophotometry data (stored in FITS files) are gathered and preprocessed with principle components analysis to reduce the number of attributes for each record to help speed up the training time as well as increase the accuracy of the classifiers.

Due to the natural difficulty of any given multi-class classification problem, the problem is reduced in difficulty; instead creating multiple classifiers that simply report whether a given star is of a particular class or not (rather than reporting the actual class). More specifically, a program will be written that generates a classifier given equal sets of spectral data from each major star class (O,B,A,F,G,K,M) and a single "guess class". This classifier will then produce another program that accepts a single set of data for a single star and returns whether or not that star can be classified as the "guess class" that was given to the original classifier. This way a single program will ultimately generate seven classifier programs to differentiate between the main spectral classes. If desired a final "wrapper" classifier program could be generated for easier use which would return the class of the given star.

TABLE 1
MAIN FEATURES OF SPECTRAL TYPES IN THE MK SYSTEM [6]

Type	Color	Prominent Lines	Examples
O	Bluest	Ionized He	10 Lacertra
B	Bluish	Neutral He, Neutral H	Rigel
A	Blue-white	Neutral H	Sirius
F	White	Neutral H, Ionized Ca	Canopus
G	Yellow-white	Neutral H, Ionized Ca	The Sun
K	Orange	Neutral Metals (Ca, Fe), Ionized Ca	Arcturus
M	Red	Molecules and Neutral Metals	Antares

As can be seen by Table 1, each class of star is distinguished by its color and the prominent lines found in its spectroscopic data. From this correlation, it should be possible to classify stars with only their given spectra.

With the ease of access for the data, a separate testing set will be generated with each of the different star classes. This testing set will be completely unique to the training sets so as to increase the chance of catching overfitting problems. Each of the classifiers will be evaluated on their accuracy of classifying the known stars class types.

Due to the success found in so many previous research projects, an Artificial Neural Net will be attempted. However, due to the nature of neural nets and their "black box" processes, very little information on how the class was actually determined can be gleaned from an ANN.

Therefore, this problem will also be solved with Decision Trees using Entropy and Information Gain. If decision trees are successful they will prove much easier to derive exactly what qualities and wavelengths were most important in classifying a given stellar class.

K-Nearest Neighbors will also be attempted despite the lack of focus on it from previous research. The distinct clustering seen in the transformed datasets makes kNN a likely candidate for stellar classification.

Considering that the data is purely an effort to classify astronomical data and has very little to do with the present moral or ethical climate, it is unlikely that the findings from this data could prove ethically objectionable.

II. Previous Work

Many different groups have researched the possibility (and application) of automatic star classification using spectral data. Currently there are models to produce automatic classification of stars, however many of these models require prior knowledge or restrictions on what possible classifications exist within a system. Most of these researcher groups simply wish to be able to use the petabytes of raw spectral data available, but cannot waste the time manually classifying billions of individual stars.

Past researchers faced many different challenges in their different attempts at this problem. A very common problem is that although there is a lot raw data available to researchers and data scientists, a very small fraction of it has been professionally classified. This means there is a relatively small data set to form classifiers and test the results (Lepine, 2004). Another common problem is the high dimensionality of the problem itself. Due to the nature of spectral data, thousands of data points must be used to represent a single star's wavelength. This means each which can lead to incredibly long training times where time complexity can be very important to keep track of and minimize (Zhao, 2007).

Many different methods and algorithms were used in the past in order to accurately classify spectral data. The most common approach was to use an artificial neural net or a mixed neural net to detect unknown patterns in the raw spectral data (Klusch, 1993). In multiple research projects concerning this or similar astronomical classification problems, neural nets of some variety were typically compared with other algorithms and were found to be more accurate overall. In many of these cases however, very little was done in terms of preprocessing the data which likely gave artificial neural networks an advantage over other algorithms because they excel at finding patterns in high dimensional problems.

Outside of neural nets, other common algorithms include hierarchical clustering and decision trees. One particular research group compared many different kinds of decision trees and compared them. In their study, they found that different kinds of decision trees worked better for different kinds of data. They found that more complicated data required different decision trees than when they tested mass quantities of data which only further emphasizes the importance of trying multiple algorithms when data mining (Zhao, 2007). Some researchers compared their algorithms against other methods of classification such as statistical clustering or expert systems (Rodriguez, 2004).

An explicit list of all algorithms seen by different research groups is seen below:

- Artificial Neural Nets
- Hierarchical Clustering
- Statistical Clustering
- Expert Systems
- Decision Trees
    - REPTree
    - Random Tree
    - Decision Stump
    - Random Forest
    - AdTree

III. Experiment and Results

Aspects of the experiment dealing directly with manipulating the data are particularly rigorous and are explained in practical detail below such that a potential reader could similarly pull down the data and access it for their own inspection.

A. Collecting the Data

The data was collected from the SLOAN Digital Sky Survey. Using their batch Spectroscopic Query Form, a selection of star data from the DR9 Science Archive Server (SAS) was collected. This data is in the format of a FITS data image file.

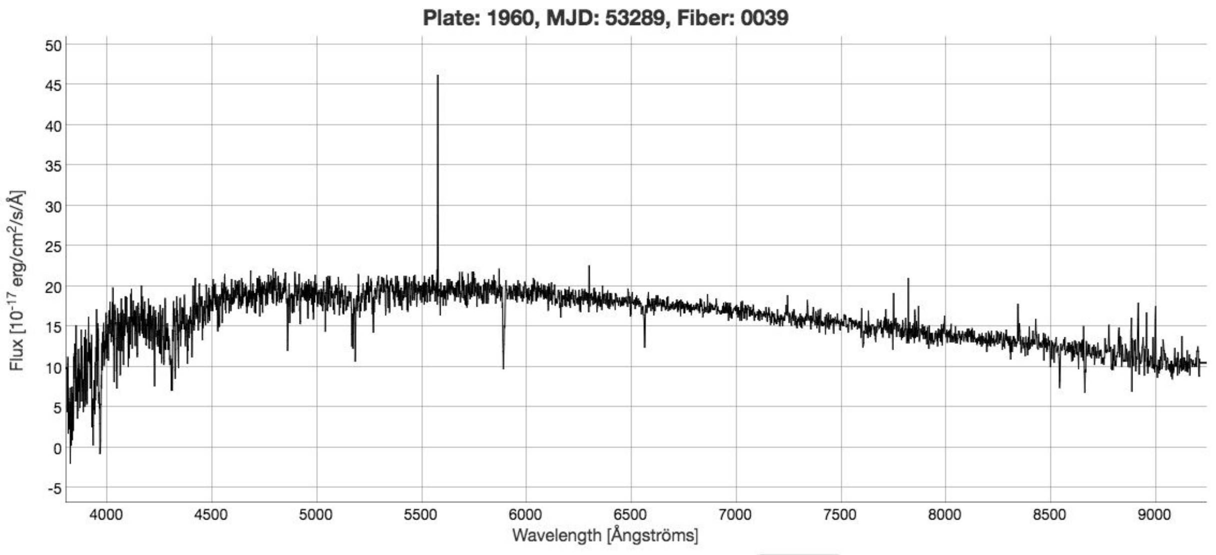


Figure 1: Example of a F9 Star Spectrum FITS Image

The FITS format is the standard data format for astronomy and is highly dimensional. The collected spectra FITS files contained much more data than just the given spectra of star which required preprocessing to extract the relevant data.

B. Initial Preprocessing

The data initially is separated into many individual FITS image files (see Figure 1) that each contain the optical spectra of a given star along with other data relevant to the star. A series of Bash and Python scripts were developed to extract the relevant data from the FITS files and combine them into a single CSV file. Due to the nature of spectroscopy and differences in the wavelengths taken, effort must be taken to make sure that when extracting the data information is not assumed or lost. Outside of the mild difficulty in ultimately generating the final agglomerated data file, the data itself is highly accurate and numerous which should be very useful in generating classifiers (Yongheng, 2006).

The data pulled from the FITS files is very clean and accurate. For this paper's particular experiment, star classes were truncated to simplify star spectral types (ie: from F9 to F) to ultimately create 7 classes of star to classify. Unusual or rare star types like Carbon stars, were similarly removed entirely as to not skew the data.

C. Principal Component Analysis

After the initial preprocessing, each spectral record contains approximately 5000 attributes. Even with a small dataset most algorithms would be crippled by an attribute size this large. Therefore *Principal Component Analysis* was performed to reduce the attribute size. Weka was used to perform the PCA and output the transformed datasets.

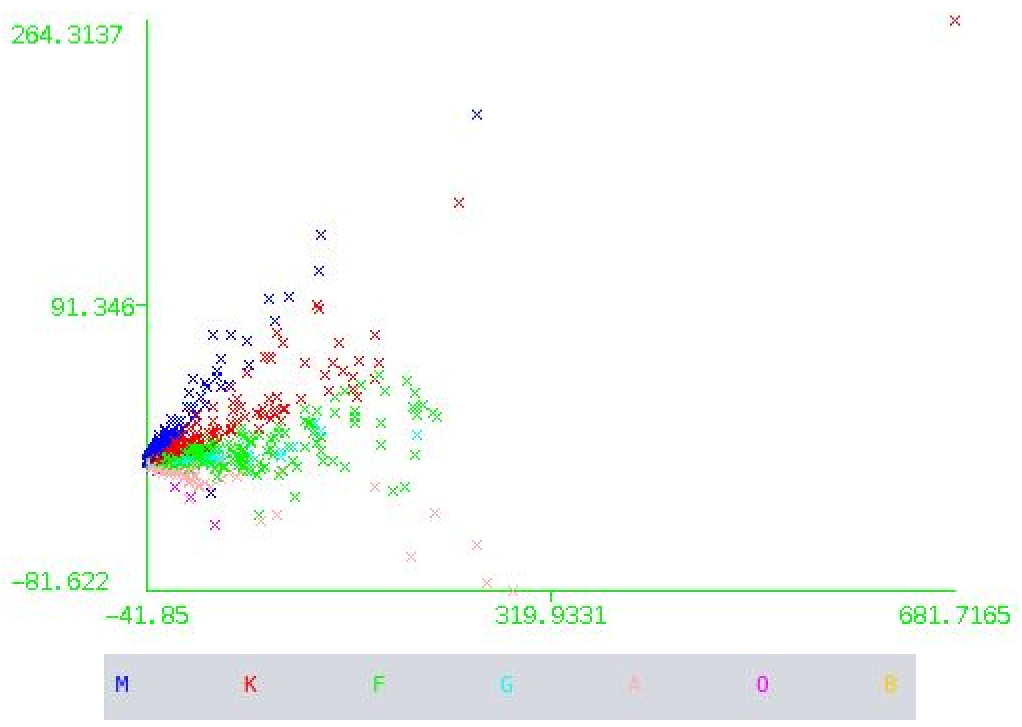


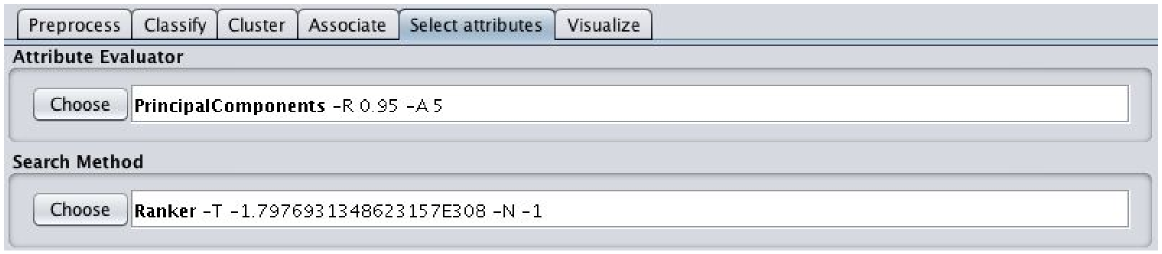
FIGURE 2: TESTING DATA AFTER PCA (1000 RECORDS)

As can be seen from Figure 2, through PCA, the approximately 5000 attribute records were reduced to two attributes. These two attributes work together to show a clear distinction between different classes. One problem however that this PCA analysis introduces is that Weka can only process up to a certain file size before problems with heap size become an issue. Because of this problem, the original test dataset had to be reduced from 10,000 records to just the 1000 seen in Figure 2.

This process can be done with both the Weka GUI or the command line and both processes will be briefly explained.

1. THE WEKA GUI

Through the Weka GUI, once a dataset has been loaded into the Weka Explorer interface, it is trivial to



PrincipalComponents -R 0.95 -A 5

Ranker -T -1.7976931348623157E308 -N -1

FIGURE 3: SELECT ATTRIBUTES TAB WITH PCA OPTIONS SELECTED

run PCA by selecting the “Select attributes” tab and choosing “PrincipleComponents” under “Attribute Evaluator”. By default the GUI will change the “Search Method” to the “Ranker” method. In the case that the data in question has classes (as in this experiment), the drop down, which by default selects the final attribute in the dataset, needs to be changed (if necessary) to the class attribute. If this is not selected properly, the class attribute could be used when the PCA calculates its eigenvalues and eigenvectors. Additionally, if the default values that Weka has implemented are unsatisfactory (such as the variance covered in the analysis), the white box that contains “PrincipleComponents” is “clickable” and values can be easily changed. Finally, the “Start” button can be pressed to begin the analysis.

When the analysis is complete, the result will appear in the “Result List” on the same tab. Each of these results can be expanded to view the transformed data in a graph format or save it to an arff file.

There are several limitations to this process however, because the output is very limited to the data itself and requires much more effort to apply the same reduction matrices to another dataset (for later testing). Due to the extra memory constraints that the GUI itself imposes on a computer, it also limits the size of the data files that can be analyzed compared to the command line. The Weka GUI is recommended for quickly reviewing small datasets for visual inspection.

2. The Weka command line

Alternatively, the Weka command line, while less user-friendly, is preferable for larger datasets and batch processing. The following example command can be used to apply PCA to a (small) training dataset (`trainingData.arff`) and generating an output file (`combinedTrainingData_PCA.arff`) of the transformed data.

```
java -Xmx4000m -cp ./path/to/weka.jar
weka.filters.supervised.attribute.AttributeSelection -c 1
-i trainingData.arff -o combinedTrainingData_PCA.arff
-E "weka.attributeSelection.PrincipalComponents" -R 0.95 -A 5"
-S "weka.attributeSelection.Ranker -T -1.7976931348623157E308 -N -1"
```

Running this command with the "-h" flag will produce an explanation for each flag and value.

Batch processing in Weka is the processes of applying a computationally complicated "Attribute Selection" on a (smaller) training data set and then applying the same resulting "Filter" onto a larger test set to be used for testing later. This is beneficial because applying a filter is computationally less expensive than performing PCA on the larger dataset initially. As a general warning, PCA produces different eigenvalues and eigenvectors for each dataset so if batch processing is not used to generate a pair of testing and training datasets, the resulting reduced attributes will not be equivalent and cannot be compared.

```
java -Xmx4000m -cp ./path/to/weka.jar
weka.filters.supervised.attribute.AttributeSelection -c 1 -b
-i trainingData.arff -o combinedTrainingData_PCA.arff
-r testingData.arff -s combinedTestingData_PCA.arff
-E "weka.attributeSelection.PrincipalComponents -R 0.95 -A 5"
-S "weka.attributeSelection.Ranker -T -1.7976931348623157E308 -N -1"
```

As stated previously, the command line interface is preferable for batch processing. Outside of heap space concerns, this is also because of how easy it is to convert a simple PCA command to a batch process.

D. Artificial Neural Net

1. Single Perceptron Implementation

Initially an oversimplified neural "net" implementation generated a simple model of the data with a single perceptron. In this implementation, a single perceptron is given each of the (two) attributes produced by the PCA as well as a bias term of 1 (Kinsman). The perceptron is trained to find a single, chosen stellar class; it either outputs a positive value (+1) if the linear combination of attributes and weights indicates that given class and a negative value (-1) if it indicates any other class. This single, chosen stellar class will be referred to as the *selected class*.

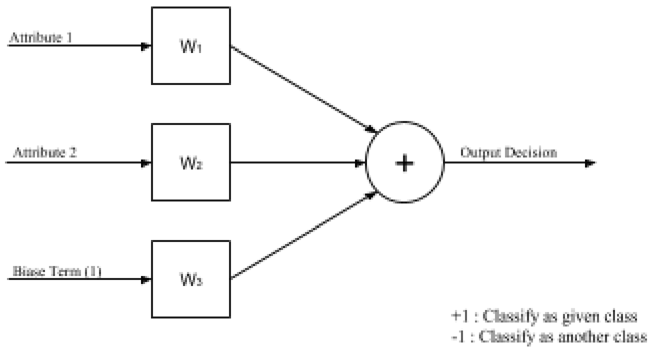


FIGURE 4: SINGLE PERCEPTRON USED FOR CLASSIFICATION
This particular perceptron is the building block of a simple feed-forward neural net (Kinsman).

The results of the single perceptron implementation were surprisingly accurate for such a simple classifier. It had accuracies (where accuracy is the percentage of correctly classified stars over the total number of classifications made) ranging from 11.5% to 92.2%. The entire set of accuracies for 100 iterations of running the algorithm for each individual chosen class is seen in Figure 5.

	O	B	A	F	G	K	M
ACCURACY	92.2	72.8	92.8	68.0	74.7	37.1	11.5

FIGURE 5: AVERAGE ACCURACY FOR 100 ITERATIONS OF ANN CLASSIFIER

Since this implementation of a perceptron can only answer whether or not the supplied record is of the *selected class*, a set of confusions matrices would provide little new data. Instead the classification accuracy for all 100 iterations (per *selected class*) is seen below in Figure 6.

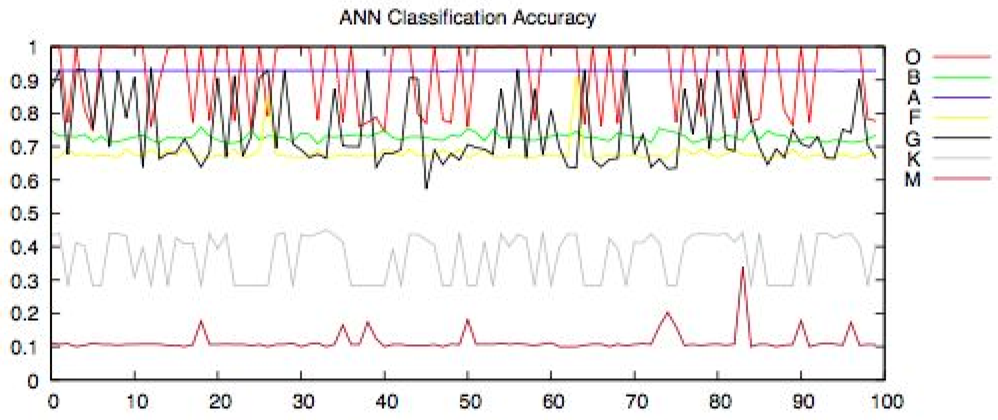


FIGURE 6: ACCURACY FOR 100 ITERATIONS OF ANN CLASSIFIER

E. DECISION TREE

A simple decision tree trainer implementation was used to dynamically generate the decision tree classifier. Unlike the Artificial Neural Net attempts, the decision tree is able to train on all of the star types at the same time.

1. GINI INDEX AND BINARY SPLITS

The decision tree trainer used the Gini Index as a measure of purity in determining the best split to make between each of the two attributes. This split was also explicitly binary in an attempt to reduce complexity of the branching factor in the resultant decision tree. The resulting decision tree was dynamically generated with a wrapper classifier Java file such that a test training set could easily (and automatically) be tested.

The results of the decision tree classifier were approximately the same as the single perceptron implementation. It had an an accuracy of approximately 70.5%

		Predicted						
		O	**B**	**A**	**F**	**G**	**K**	**M**
Actual	**O**	1	0	4	0	0	1	0
	B	0	0	1	0	0	0	0
	A	0	0	56	13	3	0	0
	F	0	0	30	201	39	34	3
	G	0	0	2	39	22	1	0
	K	0	0	14	41	2	175	36
	M	0	0	5	1	0	26	251

Figure 7: Confusion Matrix For Decision Tree

F. k-Nearest Neighbors

K-Nearest Neighbors (kNN) is a preferable method of classification when there are few attributes and the minimum set of records to accurately classify is known. In this case, we are only dealing with two attributes and the training set of approximately 500 records appears to be a good average set of stellar data. K-Nearest Neighbors is also particularly useful to scientists because of its distinct lack of training time and ease of use.

As with many classifiers, the success of kNN is highly dependent on its distance metric. Multiple distance metrics were attempted with a wide range of k values to determine the best combination to use. In particular, this model used different iterations of the general Minkowski Distance formula to determine distance between data records.

$$L_p = \left(\sum_{k=1}^{n} (|x_k - y_k|)^{p} \right)^{1/p}$$

Formula 1: The Generalized Minkowski Norm

P values of 2 to 6 were tried in combination with K values between 1 and 14. Each of these different combinations produced different accuracy ratings. This can be visualized below in Figure N.

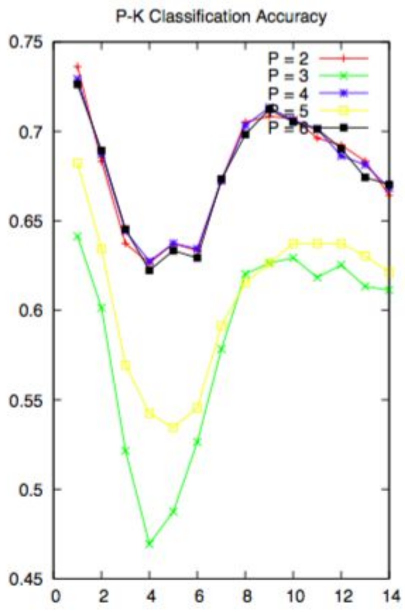


FIGURE 8: COMPARISON OF DIFFERENT MINKOWSKI FORMULAS (DIFFERENT COMBINATIONS OF P AND K)

As can be seen, the best combination of P and K values is a P=4 and K=9 combination. This value was used in the final classification test and resulted in an accuracy of just over 71.3%. The following confusion matrix represents a run with the test data.

		PREDICTED						
		O	**B**	**A**	**F**	**G**	**K**	**M**
ACTUAL	**O**	0	0	5	0	0	1	0
	B	0	0	1	0	0	0	0
	A	0	0	24	47	1	0	0
	F	0	0	7	253	30	17	0
	G	0	0	7	45	18	0	0
	K	0	0	1	72	8	174	13
	M	0	0	1	1	1	35	245

FIGURE 9: CONFUSION MATRIX FOR kNN MODEL

IV. DISCUSSION

As stated earlier, the limitation of dataset size caused by Weka's performance limitations greatly reduced the opportunities for larger training and test datasets. This being said, the results were still very promising.

Each of the classification models explored in this research paper resulted in at least approximately 70% accuracy (excluding some outliers in the ANN experiment) in determining the correct classification for a brand new set of stars. Seeing as each of these attempted classifiers are extremely simple, it is not hard to assume that better accuracy can be attained by extending them.

These results however are likely skewed due to a problem in the testing data. While the training data was mostly even in each of the classes, the testing data was unexpectedly skewed such that there were very few points being to be classified as spectral type O and B. This is in part why the artificial neural network experiment showed excessively poor accuracy. It is likely (but not proven in this paper) that the classifiers are capable of classifying these classes much better than they are represented here however because the training data was relatively equal in each of the star classes.

An interesting detail to note is in each of the confusion matrices, typically misclassification occurred between classes "similar" to each other. For example, when cross-referencing Table 1 with Figure 9, it is clear that many class F stars were misclassified as class G stars (and vice versa) likely because there share the same exact prominent spectral lines (Neutral H and Ionized Ca).

This experiment had a greater focus on processing the data to make a seemingly impossible classification with over 5000 variables not only possible, but straightforward. With the entire preprocessing and PCA process learnt and finalized, as well as the frameworks for three promising classification algorithms, there are many possibilities for future changes.

V. Future Work

Larger and better balanced datasets would be more than beneficial in any future work. As this experiment is continued, the data will be better pulled apart to generate better trainers and testing data.

As mentioned earlier, there is plenty of work that can be attempted toward making more robust classification algorithms. Seeing as a single perceptron is able to accurately determine the class of a star more than 70% of the time (for most star types) it is likely that more complicated neural networks would be able to much more accurately determine a star's type. It is also further possible to surpass the ANN's initial shortcoming of needing to a *selected class* rather than simply classifying for any particular class. Similarly the decision tree could be extended to attempting different purity measurements such as entropy and misclassification error. An attempt at multiway splits might also be useful in this case because of the number of resultant classes. Comparing these different methods might result in a more robust classifier. In the case of the k-Nearest Neighbors algorithm, only different forms of the Minkowski General Norm were tested. Other distance metrics such as the cosine similarity or the mahalanobis distance might prove more successful at classification.

Ideally this project would be available to outside researchers, so while the entire preprocessing has been largely automated, more complete documentation would be ideal for future experiments.

VI. Conclusion

This paper has introduced multiple methods of classification which can to a surprising degree classify stars based solely on their spectroscopic data. The larger focus of this paper however was the extensive preprocessing that allowed these classifiers to be successful. The original record size of more than 5000 attributes would likely never have been classified so easily or quickly. Instead through some initial preprocessing and then extensive use of Principal Components Analysis, the records were reduced to just two attributes each. The success of this attribute reduction can be seen in Figure 2. With this transformation, it is easy to see correlations in the data between different spectral classes even visually.

The results of this paper clearly indicate that it is possible to automatically classify stars with relative certainty. While the classifiers introduced in this paper are not accurate enough, they are reasonable indicators that more robust classifiers of their kind could generate accurate classifications in the future.

VII. Bibliography

[1] Yongheng Zhao, Yanxia Zhang. “Comparison of decision tree methods for finding active objects.” *Science Direct.* 2007.

[2] Jian-qiao, Xue, Li Qi-bin Zhao, Yong-heng. “Automatic classification of stellar spectra using the SOFM method. Beijing AstronomicaI Observatory, Chinese Academy of Sciences, Beijing. 2001.

[3] Rodrıguez, Alejandra, Bernardino Arcay, Carlos Dafonte, Minia Manteiga, Iciar Carricajo. “Automated knowledge-based analysis and classification of stellar spectra using fuzzy reasoning. ”Department of Information and Communications Technologies, Campus de Elvina, University of A Coruna, A Coruna, Spain. 2004.

[4] Lepine, Sebastien and Michael M. Shara. “A catalog of northern stars with annual proper motions larger than 0”.15.” Department of Astrophysics, Division of Physical Sciences, American Museum of Natural History, 2004.

[5] Klusch, M. “HNS - a hybrid neural system and its use for the classification of stars”. Institut fur Informatik und Praktische Mathemistik. 1993.

[6] Rodriguez, Alejandra, Carlos Dafonte, Bemardino Arcay, Minia Manteiga and Iciar Camcajo. “Expert Systems and Artificial Neural Networks applied to Stellar Optical Spectroscopy: A Comparative Analysis.” University of A Corun .2004.

[7] Zhang, J.N., A. L. Luo, L. P. Tu. “A Stratified Approach for Automatic Stellar Spectra Classification.” Chinese Academy of Sciences. 2008.

[8] Bin, Jiang, Pan Jing Chang, Yi Zhen Ping. “A Data Mining Application in Stellar Spectra”. School of Information Engineering, Shandong University at Weihai. 2008.

[9] Kinsman, Thomas. "Lecture 21b ANN,SVM,..." Artificial Neural Networks - Deep Learning. Rochester Institute of Technology, Rochester. 25 Apr. 2016. Lecture.

[10] "Implementing a Neural Network from Scratch in Python – An Introduction."WildML. N.p., 03 Sept. 2015. Web. 25 Apr. 2016.