Noble Kennamer 77163775

Ariel Kruger

Abhisaar Sharma

Dylan Cockerham 65243353

CS 273A Final Project

A Survey of Machine Learning methods in Astrophysics

What to talk about:

(a) Problem statement

(b)Citations

(c) Who did what

(d) Experience coding it up

(e) Results and experience getting those results

(f) Conclusion

(g) Bibliography

**Problem Statement**

For our project we worked on two different problems. Both focused on applying and comparing different machine learning methods on astrophysical data. In observational astronomy there are two major types of data photometric and spectroscopic. Photometric data is centered on measuring the flux (you can think of this as the number of photons per area per second hitting a CCD camera) for very broad wavelengths. In the case of the SDSS photometric data was collected for each target (galaxy, star, quasar) with five filters u, g, r, i, z [1]. Each of these filters corresponds to a different range in the electromagnetic spectrum and can be used to infer physical properties of the target. In the case of spectroscopic data you can think of it as photometry using infinitely many filters (in practice a diffraction grating is used). Hence with spectroscopy you have much greater resolution on the light (not necessarily visible) the target is emitting. An example of a spectrum of a typical star is shown in figure 1. There are several advantages and disadvantages to photometry and spectroscopy. The main difference is that photometry is much cheaper to collect while spectroscopy provides much greater information about the object. Because of this, large surveys will typically first take and use photometric data to plan and identify which targets that want to invest the time and resources to collect spectroscopy data on. It is also important to note that the amount of data is growing both in size and complexity. For example the SDSS surveyed over 14,000 square degrees of the sky (out of about 40,000 square degrees), collected data on over 900 million unique objects and over 4 million of these included the spectrums of the objects [2]. In addition to the already large amounts of data more data will be collected from surveys like the Dark Energy Spectroscopic Instrument (DESI),the Large Synoptic Sky Telescope (LSST), and several others [3][4]. The LSST will actually be doing only photometry on tens of billions of objects bringing us close to targeting every galaxy in the observable universe. Thus it is extremely important to develop techniques to process and analyze all this data. This is where machine learning comes.

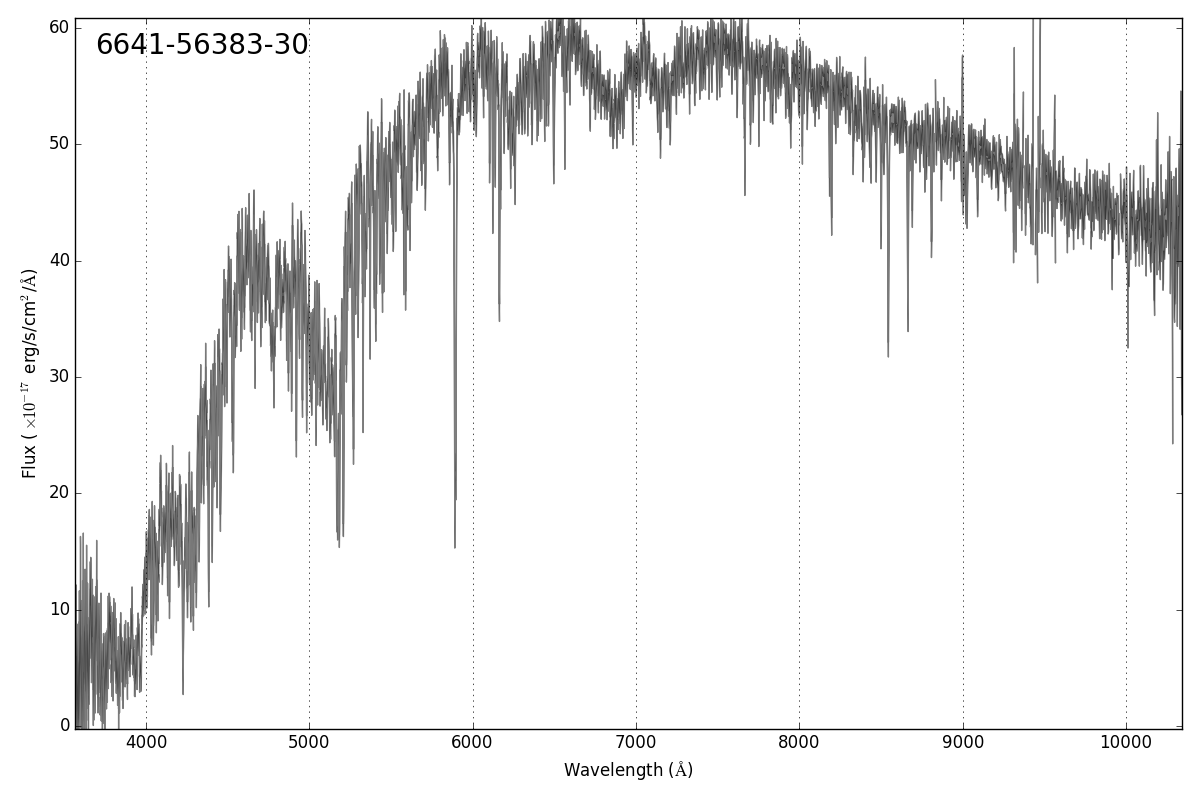


Figure 1: An example spectrum of a typical star. This spectrum comes from the Sloan Digital Sky Survey. The horizontal lines represent the amount of uncertainty of the flux value for a given wavelength.

For our project we worked on two different problems both of which used photometric data. One of the projects we worked was to classify RR Lyrae variable stars versus standard stars. This is a very important problem in astrophysics because RR Lyrae variable stars are used as standard to candles to measure distances (both galactic and extragalactic) [5]. Thus if we can develop reliable methods for discriminating between RR Lyrae stars from standard stars just using photometric data than observational astronomers can use this to select which stars to collect finer data (spectroscopy) on to determine distances (How the distances are measured is based on a relationship between the period of the variable star and its luminosity, which is interesting but not relevant to the project). The data, shown in figure 2, comes from the Sloan Digital Sky Survey (exactly how we obtained the data is discussed in a later section) and consists of four features called colors [6]. These colors are u-g, g-r, r-i, and i-z and they are the difference of magnitudes between two different bands. For example the u-g colors is the magnitude from the g band (filter) subtracted from the u band. The dataset consists of about 93 thousand objects and only about 500 of which are RR Lyrae variable stars. This huge difference between the number of positive example (RR Lyrae stars) and negative examples makes this a very challenging problem. Furthermore the detection of rare events is pervasive throughout observational astronomy so by identifying and developing methods that are good at addressing this issue might generalize to other problems. The detection of exoplanets, strong lensing, collisions, supernova, MACHOS(no one really cares about these anymore) and many more are all examples in astrophysics where detecting rare events is important.

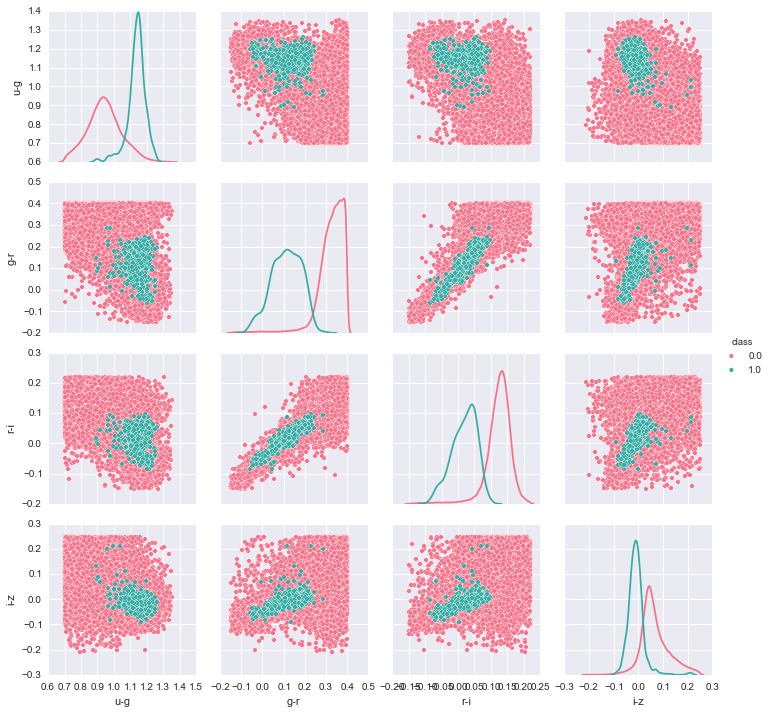


Figure 2: A pair plot of the RR Lyrae Variable stars(green) verse standard stars(pink). Our dataset consists of four features u – g, g – r, r – i, and i – z. The figure shows the relationships between all the features. For example the subfigure in the first row and first column shows u-g on the x-axis and u-g on the y-axis. The figure in the first and row and second column shows the g – r on x-axis and u-g on the y-axis. Note that the figure is symmetric about the main diagonal.

The other problem we worked on was a regression problem where we would like to assign a redshift to a galaxy using only photometric data. This dataset consists of over 60 thousand objects (all galaxies) where the features are the magnitudes of the 5 bands [5]. Hence there are 5 features u, g, r, i, and z. The data is shown in figure 3.

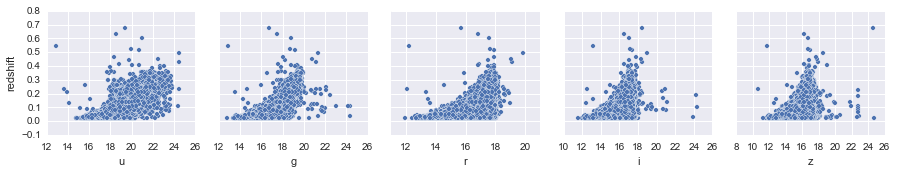
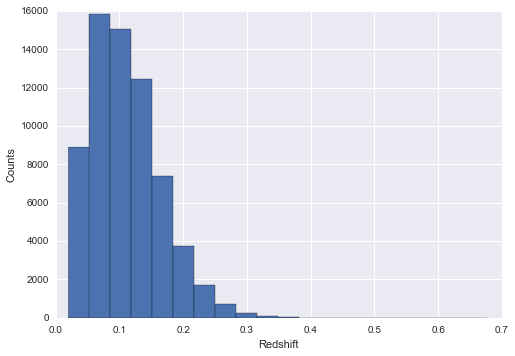


Figure 3: This figure consists of five subfigures. One for each feature of the dataset the feature is on the x-axis of each plot and the redshift is on the y-axis of every plot. Hence the middle plot shows the value of the r magnitude on the x-axis and the redshift on the y-axis with a dot for every galaxy in the dataset.

Figure 4 shows a histogram of the redshifts. As we can see the redshift ranges from about 0.001 to 0.4 with a peak around 0.1.Figure 4: A histogram of the redshifts for every galaxy. As we can the redshift ranges from about 0.001 to about 0.4 with a peak around 0.1.

Redshift is very similar to a Doppler shift in that it is a phenomenon where the wavelength of the light emitted by an object gets distorted based on the motion of the object relative to the lab frame. Since all the object we are looking at are galaxies the redshift is actually from the cosmological expansion of the universe between us and the galaxy and the peculiar motions of the object play a very little role in causing the redshift. Since for our objects the redshift is caused by the expansion of the space-time there is a direct relationship between the objects distance from us and its measurement. By determining the redshift of as many objects as we can scientists can better map the large scale structures of the universe and compute n-point correlation statistics and compare these with predictions from theoretical models of the universe. For our dataset the ground truth comes from determining the redshift from spectrums of the objects since it is a much easier problem when the spectrum is known. But as stated earlier we do not always have the spectrum and future surveys like the LSST will have no spectrums [4]. Thus it is important to develop models that can determine redshift values from only photometry.

**Results**

For the RR Lyrae variable star classification problem we trained several different models and evaluated them across several metrics. For many classification problems the overall accuracy is considered to be one of the most important metrics, however for this problem we believe that the precision the most important metric to consider. The reason for this lies in the fact that the data has only 500 examples of the positive class (RR stars) where it has over 90 thousand examples of standard stars (the negative class). Thus a classifier could predict every object to be a star and receive an accuracy of over 99%. But this would be a poor classifier because for scientific reasons we are most interested in obtaining a strong classifier for RR Lyrae variable stars. Hence our argument for why the precision is the critical metric to look at.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Models** | **precision** | **recall** | **F1** | **ROC\_AUC** | **Accuracy** |
| KNN | 0.697916667 | 0.540322581 | 0.609090909 | 0.769535265 | 0.996306794 |
| D Tree | 0.638655462 | 0.612903226 | 0.625514403 | 0.805523368 | 0.996092072 |
| Naive Bayes | 0.027458924 | 0.983870968 | 0.053426757 | 0.898657701 | 0.814351971 |
| Linear SVM | 0.132890365 | 0.967741935 | 0.23369036 | 0.966968282 | 0.966202869 |
| ANN | 0 | 0 | 0 | 0.5 | 0.994674912 |
| Log Reg | 0.13986014 | 0.967741935 | 0.244399185 | 0.967939701 | 0.96813536 |
| Random Forest | 0.688172043 | 0.516129032 | 0.589861751 | 0.757438491 | 0.996177961 |

Table 1: Shows the results from different models on the RR Lyrae classification problem.

Table 1 shows the results of many of the models we tried for this problem. Note that all the models except for Naïve Bayes achieves an accuracy between 96%-99.6%. However the models differ greatly in the precision and recall. The models doing the best in regards to precision are KNN, Random Forest, and Decision trees they are also achieving the highest F1 score which is the harmonic mean of the precision and recall. It is also interesting to note that Logistic Regression and SVM are doing almost exactly the same. One thing that is quite surprising is how poorly the neural network is performing. The neural network is a two hidden layer network using dropout with an architecture of 4-20-15-2 (from the input layer with 4 nodes to the output layer with 2 nodes). From the results we can see that the ANN is predicting everything to be a standard star resulting in a high accuracy but a 0 precision and recall. We suspect that there is simply not enough data/examples of RR Lyrae stars for the network to learn.

Figure 5 shows the performance of the models in greater detail by displaying the ROC and precision and recall curves of all the models.

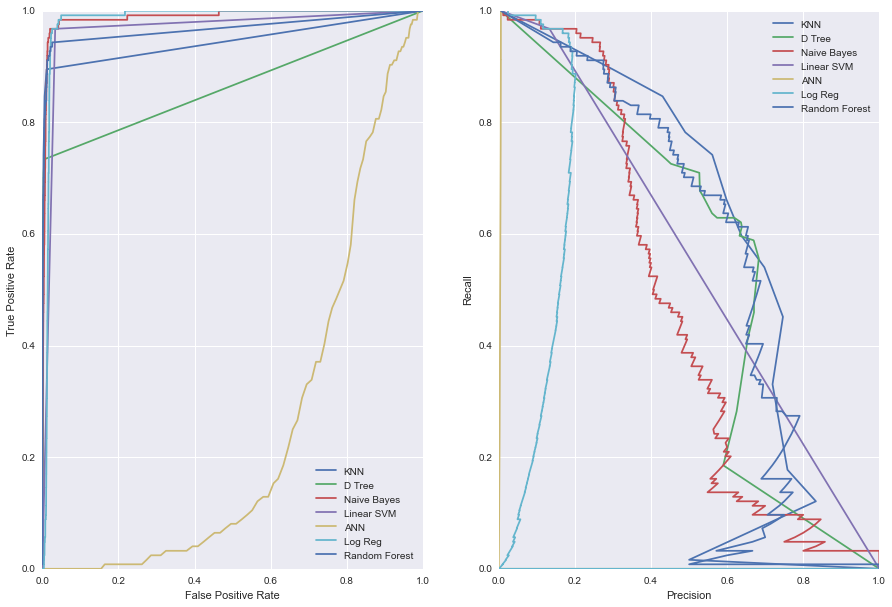


Figure 5: The left plot is the ROC curve for each model and the right plot is precision recall curve for each model. The right plot is much more interesting to look at for this project given that the precision and recall are arguably more important metrics when one is working with rare events.

To try to compensate for the fact that there were drastically more examples of standard stars than RR Lyrae stars we tried two separate approaches. The first being to try to model the RR Lyrae stars with kernel density estimation and then sample from this density new RR Lyrae stars which were then added to the training set but not the test set.

Figure 6 and 7 shows the results of this kde sampling after optimizing for the bandwidth.

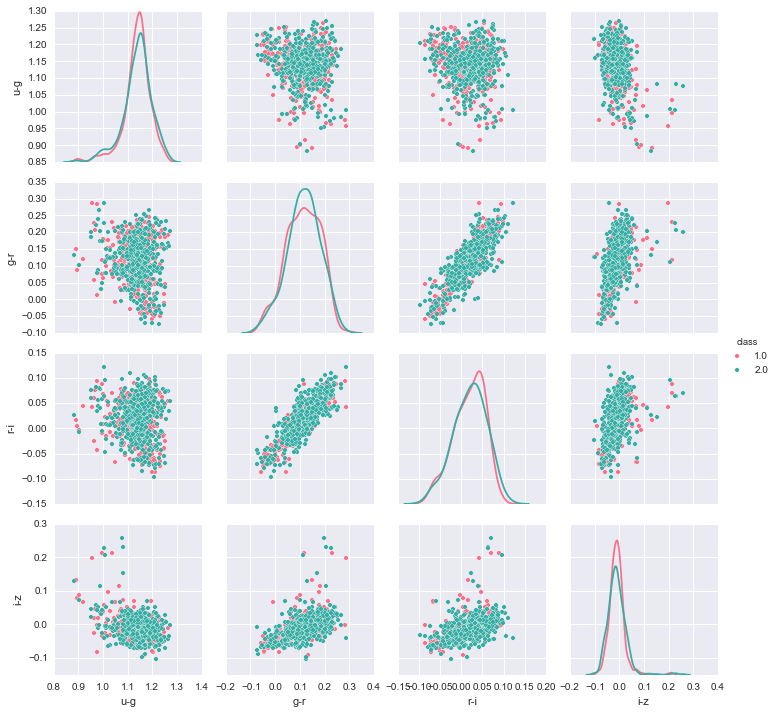


Figure 6: This plot is very similar to figure 2 except that the pink points shows the original RR Lyrae stars and the green points show the sampled RR Lyrae stars using KDE. No standard stars are shown in this figure

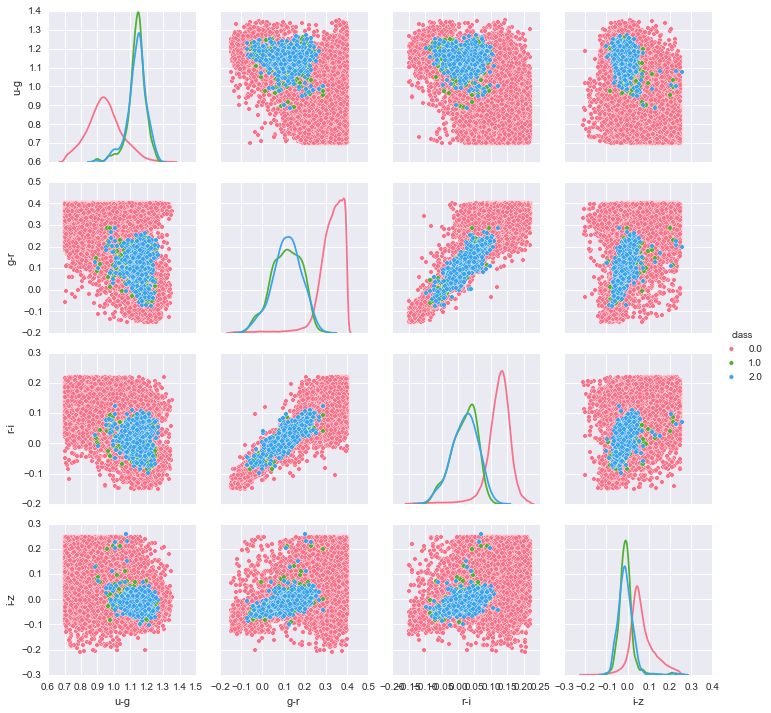


Figure 7: Is the same as figure 6 except with the standard stars added in. The pink points are the standard stars, the green points are the original RR Lyrae stars, and the blue points are the sample RR Lyrae stars from KDE.

From visual inspection of figure 6 and 7 it appears that KDE is doing a good job at modeling the RR Lyrae class. However the quantitative results in figures 8-12 tell a different story.

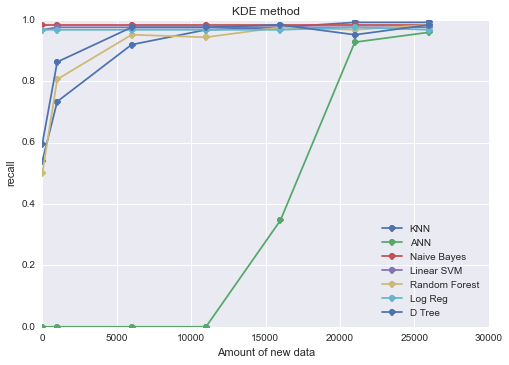


Figure 8: The x-axis is the amount of new data sampled from kde and the y-axis shows the recall.

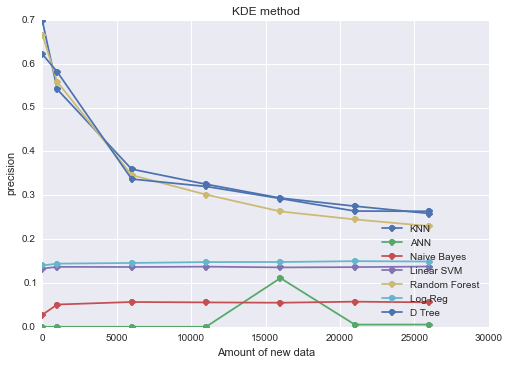


Figure 9: The x-axis is the amount of new data sampled from kde and the y-axis shows the precision.

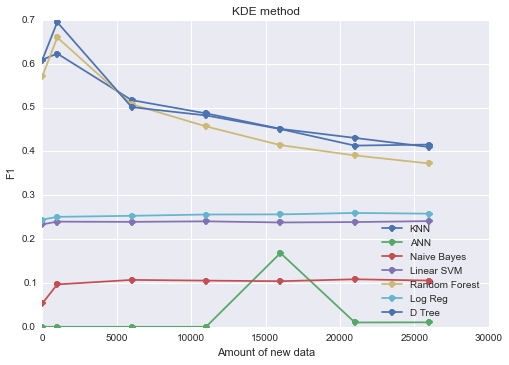


Figure 10: The x-axis is the amount of new data sampled from kde and the y-axis shows the F1 score.

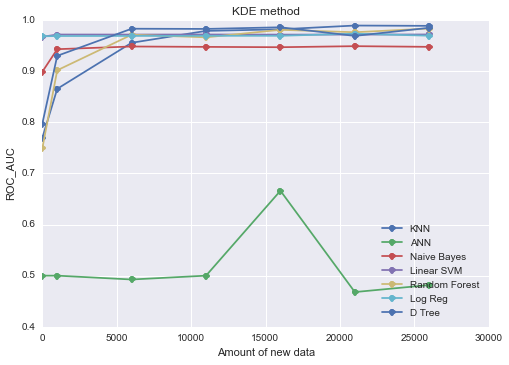


Figure 11: The x-axis is the amount of new data sampled from kde and the y-axis shows the ROC AUC (area under the curve).

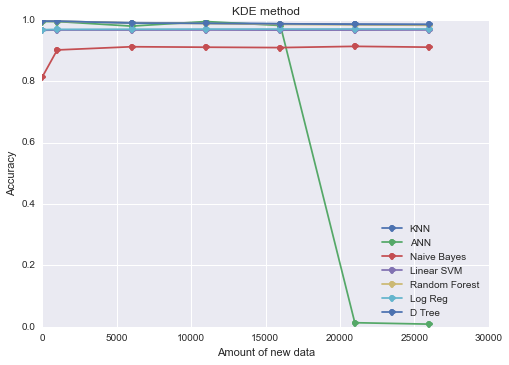


Figure 12: The x-axis is the amount of new data sampled from kde and the y-axis shows the accuracy.

From these plots we can see that the recall for all the models does increase as more data is added up to a certain point. And as would be expected when the recall increases the precision decreases. But when the F1 score also decreases or stays constant meaning that the recall is increasing faster or at the same pace as the decrease of the precision. Thus it seems like the models are learning to predict more RR Lyrae stars but are not actually learning a better representation of them than the initial results. We can also see that the accuracy stays roughly constant except for Naïve Bayes where it improves and the ANN, which I discuss below.

It is also interesting to note that from the plots it appears that the ANN goes from predicting only standard stars to predicting mostly RR Lyrae stars. This can be seen from a precipitous rise in the recall but a precipitous drop in the accuracy. I’m not exactly sure why the ANN is behaving but my first suspicion would be that it is not being initialized in the best possible way. If we could start the ANN off in a better portion of the error space than perhaps it would do a much better job at learning. If more time were available than this would be my first line of investigation for improving the ANN as the outside literature does show that ANN should excel at these classification problems.

It is also entirely possible that even though it appears visually that KDE is doing a good job at modeling the RR class that it is actually and this is what is throwing off the models.

The next method we tried for adding more examples of the RR Lyrae stars was to simply randomly sample from the existing population. This random sample was created by simply randomly picking a RR Lyrae star to “observe” again with replacement. We than randomly put these samples into the training set so as not to skew the ordering. The results of this method are shown in figure 13 through 17.

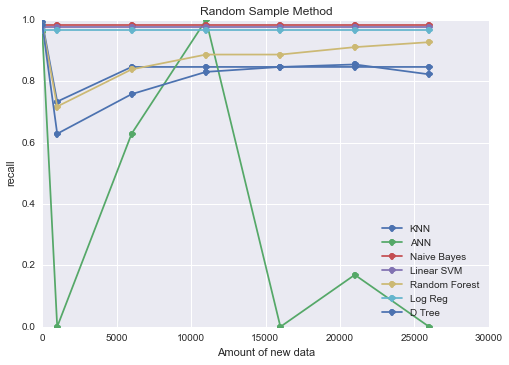


Figure 13: The x-axis is the amount of new data sampled from the random sample approach and the y-axis shows the recall.

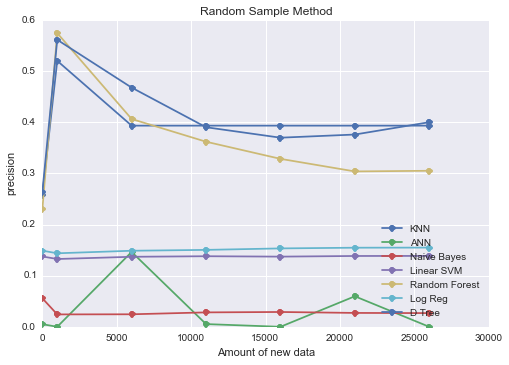


Figure 14: The x-axis is the amount of new data sampled from the random sample approach and the y-axis shows the precision.

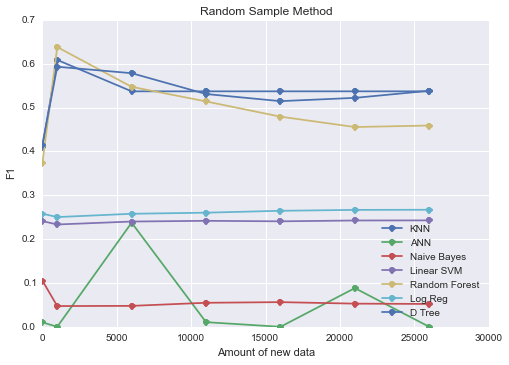


Figure 15: The x-axis is the amount of new data sampled from the random sample approach and the y-axis shows the F1 score.

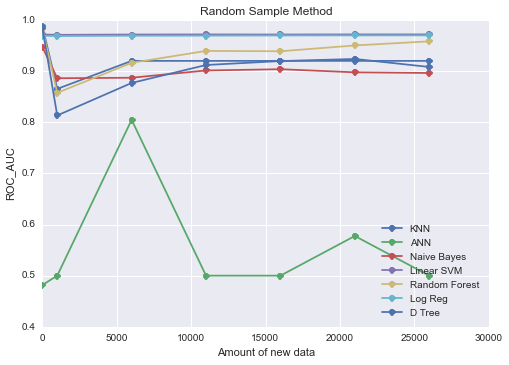


Figure 16: The x-axis is the amount of new data sampled from the random sample approach and the y-axis shows the ROC AUC.

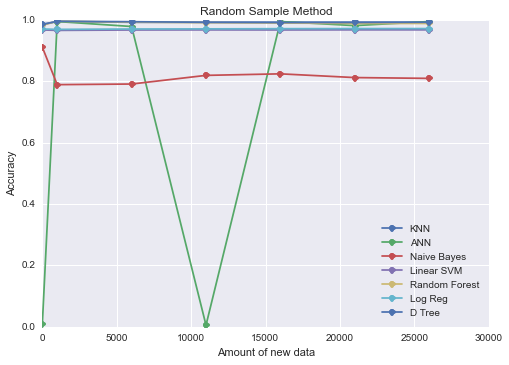


Figure 17: The x-axis is the amount of new data sampled from the random sample approach and the y-axis shows the accuracy.

From the above figures it appears that this approach does slightly better than KDE although a lot of the analysis that applied to that approach still applies here although the results are not changing as drastically as in KDE. Again I’m really not sure what the problem is with the ANN from other tests I’m confident that the implementation is correct.

For the galaxy redshift problem we trained a variety of models with the results shown in table 2. To train we took our dataset of over 60 thousand objects and divided it into a train and test set with a three quarters of the data belonging to the training set. To optimize the models cross validation was used.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **models** | **Explained Variance** | **Mean Absolute Error** | **R2 Score** | **Mean Squared Error** |
| KNN | 0.800698468 | 0.017679692 | 0.800697387 | 0.000580438 |
| Boosted DT | 0.802792119 | 0.017699619 | 0.802757243 | 0.000574439 |
| D Tree | 0.701416943 | 0.022384838 | 0.701377469 | 0.000869692 |
| RF | 0.702886049 | 0.022369561 | 0.702845702 | 0.000865416 |
| Ridge Reg | 0.61555643 | 0.022984036 | 0.615466487 | 0.001119894 |
| LR | 0.365825959 | 0.033775144 | 0.365034666 | 0.001849237 |

Table2: Shows the results from different models trained on the galaxy redshift regression problem.

And figure 18 below shows the true redshift versus the redshift predicted by each model across each band (hence there are 5 subplots one for each band).

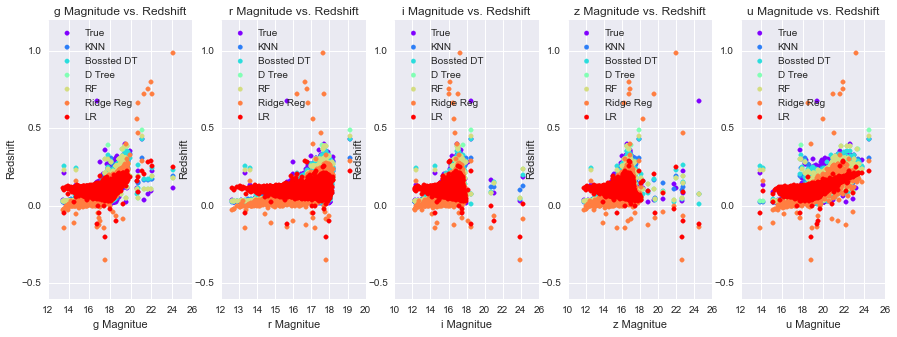


Figure 18: This figure is very similar to figure 3 except instead of only showing the true redshift it shows the predicted redshift for each model.

From the table and the figure above we can see that the two most competitive models were KNN regression and boosted decision tree regression. The worst models were the linear models vanilla linear regression and ridge regression. In between these were decision trees and random forest. But I was actually quite surprised that all the models were able to do fairly well. Although one can see in the figure above that the linear approaches systematically failed to achieve good values for smaller magnitudes.

**Conclusion**

This project served as an introduction to applying machine learning to astronomy. We worked on two different problems a classification problem where we tried to classify RR Lyrae variable stars from standard stars and a regression problem where tried to determine the redshift of galaxies. Both project used photometric data from the Sloan Digital Sky Survey and had their separate challenges. In particular the classification problem had the difficult property where there was much more of one class (Standard stars) than the other. Building a classifier to detect these rare events was a difficult challenge. For both problems we compared a variety of models.

From this report we can see that the nearest neighbor and tree (random, boosted) approaches did very well for both problems. Even though both problems are very different physically and in regards to what they are trying to pick both used photometric data. As stated previously photometry gives you a very coarse representation of the object so perhaps the success of these methods speaks to their ability to work with a feature space where you do not have fine details about the objects (the representation contains little information compared to a spectrum representation).

The linear models also did not do well for either problem indicating that there is quite a bit of complexity in the actual physical relationships that we are observing. And there needs to be more investigation into the neural networks. If more time were permitted a more systematic study would be done of training a variety of network models with different hyper-parameters.

The KDE and random sample approach for adding more example of the RR class did not do as well as we expected and analysis was given for why this might be the case. Adding in more data has shown to be a successful approach for a variety of problems especially with images. Thus we think more investigation needs to be done to conclude whether this approach is viable for this problem. Perhaps it is necessary to incorporate domain expertise about the physical objects and use this build a generative model. It is entirely possible that more clever approach such as this might achieve even better performance that was presented.

We would also like to add that this is only the tip of the iceberg for applying machine learning methods to astronomy. As stated in the introduction the amount of data is growing both in size and complexity and it important to develop models that can process and analyze it. This is very exciting and active area of research for both physicist and data science experts.

**Who Did What** (Everybody write what you did)

Looking at the commit history of the group GitHub can give a quick picture as to who did what. The Github can be found at the following link:

https://github.com/NobleKennamer/astro\_porject

**Noble Kennamer –**

* I organized and collected the two data sets [5].
* I wrote the code for the pipeline, which included making the visualizations for the data; training the models, compute metrics, and displaying the results for the models in data frames and figures. This pipeline allowed anyone in the group to write a model (conforming to a specified a.p.i) and quickly plug the model into the pipeline and immediately visualize the results in comparison with other models. Most of the figures in this report were generated from this pipeline.
* I implemented a Neural Network for the RR Lyrae classification problem. I used TensorFlow to define the graph structure of the network [7].
* I implemented Naïve Bayes for the RR Lyrae classification problem.
* I implemented Linear Regression (gradient descent based implementation) for the galaxy redshift problem
* I optimized several models from the sci-kit learn python library such including KNN, Decision Trees, Random Forest, and SVM for the RR Lyrae classification problem. And Decision Tree, Random Forest, Boosted Decision Tree, KNN, and Ridge regression for the galaxy redshift problem [8][9][10][11][12][13][14] [15].
* Wrote the code to add more RR Lyrae stars to the training set using a KDE approach (using sci-kit learns implementations) and simply selecting a random sample from the original data to duplicate in the original training set [16].
* Helped organize the poster.
* Wrote the introduction, results, analysis, and conclusion of the paper.

**Ariel Kruger –**

I implemented Decision Tree for classification. I used entropy to calculate the information gain of the features and consequently decide what was the best feature to use to build the tree. As the features of our data are real values I used the library panda in order to have the data discretized and to control the maximum depth of the tree.  
 I also tested different parameters in order to optimize and have the best models for Random Forests and Boosted Decision Trees.

**Abhisaar Sharma -**

* Helped make up the common api for the project that different models should follow.
* Wrote abstract classes for the generalized prediction model and data access object.
* Implemented linear, rbf, gaussian, sigmoid and polynomial kernels for the Kernel SVM - compared the performance of each and a custom weight grid search over class weights for the SVM models.
* Implemented a version of SVM which solves for the lagrange multipliers in the dual space using cvxopt - solution dropped since it does not generalize for large datasets (Size - O(N^2)) [17].
* Implemented a version of SVM that solves for lagrange multipliers using SMO algorithm for SVM.
* Optimized SVM code and ran under different python environments like pypy and Cpython [18] [19].
* Contributed to discussions for the project, poster and the report.

**Dylan Cockerham –**

* Designed, organized and did all of the construction work for the poster.
* Contributed to the report and served as the primary proofreader and editor.
* Investigated lasso, elastic net, and Bayesian ridge methods for classification (all of which was ultimately excluded, as various attempts proved these methods to be incompatible with the data).
* Implemented logistic regression method in python.

In addition to our individual contributions our group was in constant contact either meeting in person, communicating through GitHub or through our facebook group. This led to many fruitful discussions on what would be interesting things to try, advice on why something might not be working or how it could work better and also kept everyone in the loop on what others were doing. Ultimately these discussions not only enhanced learning, but we feel led to a stronger project.

**What did we use** (Everybody write about the libraries and other sources you used to do your work)

**Noble Kennamer**

* Throughout the entire project I wrote all my code in python using numpy (matrix library).
* To gather the data I used the library astroML which is a python library written by Jake Vanderplas for downloading astrophysical data from various sources. The data we use come from the SDSS and both problems were actually introduced to me from Jake Vanderplas’s book “Statistics, Data Minding, and Machine Learning in Astronomy: A Practical Python Guide for the Analysis of Survey Data”. The main inspiration from this project came directly from reading his book over the summer [5] [20].
* To build the pipeline I used Matplotlib and Seaborn to create the plots. And functions in the sci-kit learn library to split the data into training and test sets and to compute various metrics for both problems [21] [22][15].
* To Implement the Neural Network I used TensorFlow to define the graph structure of the network and compute the gradients [23][7]. I also read two papers one on dropout which I ended up using for the network by Geoff Hinton and another on the difficulties of training deep neural networks by Yoshua Bengio [24] [25]. Both papers helped me in trying to properly train the network and when regularization is important. I thought these papers were especially poignant for this problem because I was worried about the network not being able to learn RR Lyrae stars based on how few of them we had.
* I implemented Naïve Bayes and Linear Regression (using gradient descent) from scratch.
* Any other model that I worked on optimizing for the respective problems I used an implementation from sci-kit learn.

**Ariel Kruger**

* Just Python and numpy.

**Abhisaar Sharma -**

* All the code was written in python using - common modules like python's abc (abstract base class), numpy, matplotlib and sklearn.
* For svm, an implementaion of sklearn was used [26].
* For optimizing coefficients of lagrange multipliers in the dual svm problem - package cvxopt was used [17].
* For analyzing the running time of the SMO algorithm, pypy and Cpython were used [19] [18].

**Dylan Cockerham –**

* I used Python and numpy, and nothing else that the others didn’t use.

**Experience** (Everybody write about your experience doing this project)

**Noble Kennamer**

I had a great experience working on this project. I learned a lot about applying machine learning methods on real world problems. Not only did I enhance my theoretical knowledge of different machine learning algorithms and methods to evaluate them I also gained practical skills in working with data.

Specifically I learned a lot about how to effectively visualize data and display results in a meaningful way that helped me to make decisions about what I should try next or why a particular model is or is not working. On the visualization side I learned a lot about the matplotlib library, the seaborn library and how to take advantage of pandas dataframes to quickly visualize data.

I learned a lot about TensorFlow (Google’s new Deep Learning library) by going through many of the tutorials and ultimately using it to implement a Neural Network(with dropout) for classification. I’m already applying these newfound skills towards my research by using tensor flow to implement a convolutional network for analyzing spectrum data.

I also increased my knowledge of the sci-kit learn library including what kind of models it offers and how to train them along with what code it provides for evaluating them. I can also use this knowledge in my research by using the sci-kit learn library to quickly try out many basic models and get baselines.

In addition to these practical skills I also learned a lot about many different machine learning models such as decision trees, random forests, boosted decisions trees, nearest neighbor methods, density estimation methods, various enhancements to linear regression, support vector machines and generative methods. Either by implementing them on my own or by using a sci-kit implementation I was forced to read in more detail about these methods to get an idea if they would be effective for the problems faced in this project and how to tune them to get the best performance possible out of them.

Comparing the different models and also the different methods for adding more data in my training set were a big take away for this project. The comparison gave me greater insights into the strengths and weaknesses of the different approaches and is probably one of the most important lessons I learned from this project. The comparison between the different approaches will be elaborated on in the results and conclusion section.

I enjoyed working in a group. This project is closely related to my Ph.D. research and by having the opportunity to share my knowledge of astrophysics and my related research. I was able to get valuable insights from members of the group of what might be interesting approaches to try. I think I gained a lot more by working in a group that I would have otherwise.

**Ariel Kruger –**

I appreciated working with different models and seeing the comparison between the results generated by each one of models. Also implementing a model is always fun and give many insights on how things work internally.

Using sci-kit learn library was awesome. Very easy to use, having a standard interface for all different models which makes easy to apply them to our dataset. Very useful when deciding what model will work the best for a specific dataset.

**Abhisaar Sharma -**

I had a good experience working on this project. I learnt about applying machine learning methods on real world problems and gaining practical skills in working with data. I also learned about some data visualization tools like seaborn library and pandas dataframes to quickly visualize data. I also learnt about the sci-kit learn library including what kind of models it offers and how to train them along with what code it provides for evaluating them.

Most of my work was based on implementing the SVM - during its implementation I realized how important it is for theoretical results to be computationally feasible - optimizing lagrange multipliers for a large dataset is a tough problem in SVM, since our computation is happening in the dual space. It is constrained by memory requirements and forced me to look for a slower but a more memory friendly algorithm calleI had a good experience working on this project. I learnt about applying machine learning methods on real world problems and gaining practical skills in working with data. I also learned about some data visualization tools like seaborn library and pandas dataframes to quickly visualize data. I also learnt about the sci-kit learn library including what kind of models it offers and how to train them along with what code it provides for evaluating them.

**Dylan Cockerham –**

This project was something of a nightmare for me. I was less familiar with the project’s content than the rest of the group, and while I wholeheartedly supported the project I nonetheless found myself scrambling to keep up, which meant that I got to do a lot of learning just to get to the starting point. Then, several of the sections that I undertook proved to be fruitless endeavors, I fact that I was loathe to admit until I had exhausted those options, so I found myself desperately scrambling to find more ways to contribute to the project as the clock ticked down.

Difficulties aside, I learned a great deal about cross validation while working on the project, and I feel much more comfortable with all of the methods of classification that we used for the project than I did earlier in the quarter. It was also an interesting learning experience use python notebooks, as I had only used Python a small bit in the past; I found that the python notebooks were a very useful method for implementing the project, and easy to work with once I’d figured them out. So, even though working on the project caused me no end of grief, I got a lot out of it and I’m thankful to have worked on it.

**Bibliography**

[1] “Measures of Flux and Magnitude | SDSS.” [Online]. Available: http://www.sdss.org/dr12/algorithms/magnitudes/. [Accessed: 07-Dec-2015].

[2] “Scope | SDSS.” [Online]. Available: http://www.sdss.org/dr12/scope/. [Accessed: 07-Dec-2015].

[3] “DESI-brochure.v4 - DESI-brochure.pdf.” [Online]. Available: file:///Users/nobleWkennamer/Downloads/DESI-brochure.pdf. [Accessed: 07-Dec-2015].

[4] “About LSST.” [Online]. Available: http://www.lsst.org/about. [Accessed: 07-Dec-2015].

[5] Z. Ivezic, A. Connolly, J. Vanderplas, and A. Gray, *Statistics, Data Mining, and Machine Learning in Astronomy*, 1st ed. Princeton University Press, 2014.

[6] “SDSS.” [Online]. Available: http://www.sdss.org/. [Accessed: 07-Dec-2015].

[7] “Getting Started — TensorFlow.” [Online]. Available: http://tensorflow.org/get\_started/basic\_usage.md. [Accessed: 17-Nov-2015].

[8] “1.1. Generalized Linear Models — scikit-learn 0.17 documentation.” [Online]. Available: http://scikit-learn.org/stable/modules/linear\_model.html#ridge-regression. [Accessed: 07-Dec-2015].

[9] “3.2.4.3.1. sklearn.ensemble.RandomForestClassifier — scikit-learn 0.17 documentation.” [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. [Accessed: 07-Dec-2015].

[10] “sklearn.neighbors.KNeighborsClassifier — scikit-learn 0.17 documentation.” [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html. [Accessed: 07-Dec-2015].

[11] “sklearn.neighbors.KNeighborsRegressor — scikit-learn 0.17 documentation.” [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html. [Accessed: 07-Dec-2015].

[12] “sklearn.tree.DecisionTreeClassifier — scikit-learn 0.17 documentation.” [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html. [Accessed: 07-Dec-2015].

[13] “sklearn.tree.DecisionTreeRegressor — scikit-learn 0.17 documentation.” [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html. [Accessed: 07-Dec-2015].

[14] “Decision Tree Regression with AdaBoost — scikit-learn 0.17 documentation.” [Online]. Available: http://scikit-learn.org/stable/auto\_examples/ensemble/plot\_adaboost\_regression.html. [Accessed: 07-Dec-2015].

[15] “3.1. Cross-validation: evaluating estimator performance — scikit-learn 0.17 documentation.” [Online]. Available: http://scikit-learn.org/stable/modules/cross\_validation.html. [Accessed: 07-Dec-2015].

[16] “2.8. Density Estimation — scikit-learn 0.17 documentation.” [Online]. Available: http://scikit-learn.org/stable/modules/density.html. [Accessed: 07-Dec-2015].

[17] “Home — CVXOPT.” [Online]. Available: http://cvxopt.org/. [Accessed: 07-Dec-2015].

[18] “Cython: C-Extensions for Python.” [Online]. Available: http://cython.org/. [Accessed: 07-Dec-2015].

[19] “PyPy - Welcome to PyPy.” [Online]. Available: http://pypy.org/. [Accessed: 07-Dec-2015].

[20] “User guide: contents — astroML 0.2 documentation.” [Online]. Available: http://www.astroml.org/user\_guide/index.html. [Accessed: 07-Dec-2015].

[21] “Seaborn: statistical data visualization — seaborn 0.6.0 documentation.” [Online]. Available: http://stanford.edu/~mwaskom/software/seaborn/index.html. [Accessed: 07-Dec-2015].

[22] “API Reference — scikit-learn 0.17 documentation.” [Online]. Available: http://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics. [Accessed: 07-Dec-2015].

[23] “whitepaper2015.pdf.” [Online]. Available: http://download.tensorflow.org/paper/whitepaper2015.pdf. [Accessed: 17-Nov-2015].

[24] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.

[25] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS’10). Society for Artificial Intelligence and Statistics*, 2010.

[26] “sklearn.svm.SVC — scikit-learn 0.17 documentation.” [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html. [Accessed: 07-Dec-2015].

**Appendix**