An Empirical Replication of CNM06: Supervised Model Comparisons

Servin Wayne Vartanian SVARTANI@UCSD.EDU
Department of Cognitive Science,
University of California: San Diego

Abstract

This research paper is my mini, and slightly different, recreation of the CNM06 paper by Niculescu-Mizil. I compare the performance of three Machine Learning algorithms (and their kernel variations), using the best hyperparameters, within 10 trials, with 4 different datasets. I also deviate slightly from the original CNM06 paper by instead of taking a sample of 5000 per trial for training/testing, I split each dataset in 4 different partitions per trial. This still yields in a training set of over 5000 data samples per iteration. Lastly, the three different algorithms I'll be using are Decision Trees, SVM, and Random Forests.

Keywords: SVM, Decision Tree, Random Forest, Binary Classification, CNM06, Supervised Models, Hyperparameters.

1. Introduction

This project is based on the STATLOG trials, now known as CNM06, by Rich Caruana, and Alexandru Niculescu-Mizil performed and published 2006. Their study was revolutionary, as it thoroughly analyzed the performance metrics of 10 different algorithms, with each one having two to three different kernel and solver variations. It showed how each algorithm performed under various datasets, when a random sample size of 5000 attributes were chosen to do cross validation on. Overall, their paper presents their findings very well and for anyone interested in finding more information on their paper after reading my report may visit the link in my references.

As stated before, my project is based on that paper, but it will be a smaller-scale analysis of their methods and findings. My project also does things a little differently, as I thought it would make an interesting analysis to do so. Firstly, my experiment ran for a total of 10 trials, over the four datasets I used for my analysis: "Dry Bean Dataset" (BEAN), "Adult Dataset" (ADULT), "Skin Segmentation Dataset" (SKIN), and "Letter Recognition dataset" (LETTER). The three algorithms I used on each dataset, per trial, were Decision Trees (DT), Support Vector Machines (SVM), and Random Forests (RF). Since the primary focus of my analysis was the actual performance of each dataset, rather than a deep analysis of their classifications themselves, I decided that one metric, like mean or accuracy, would not suffice to show which algorithm performed better. Thus I used 5 metrics, Accuracy, Recall, Precision, F1-Score, and Weighted Mean to perform a thorough and diverse analysis of their performance. With all that being said, I'd like to note something my analysis does differently than CNM06. I didn't take 5,000 samples from each dataset and use those for my training and testing. My approach was to split my dataset into 4 different partitions per trial, per algorithm, per dataset. What this means is that every time my test ran, it would shuffle it's dataset, split it into 80/20, 70/30, 60/40, and 50/50 partitions. Then it would perform all my algorithms on each partition, and all that would yield 1 trial, for 1 dataset. I made sure all my datasets were over 10,000 attributes so none of my training sets were smaller than 5,000 datasets. By the end of

this, My code had run 10 trials, over 4 datasets, each being split 4 different ways, for 7 algorithms (3 truely different algorithms, but for each algorithm I tested 2-3 different kernel and solver variations). This ended up yielding a massive total of 480 trials performed and analyzed across 5 scoring metrics!

2. Methodology

2.1. Learning Algorithms

As stated before, there were three learning algorithms used in this report, but I ran each algorithm using different kernels for its optimal hyperparameters: Random Forests (with "Entropy" and "Gini" criterion), SVM's (with "Linear", "Rbf", and "Sigmoid" kernels), and Decision Trees (with "Gini" and "Entropy" criterion as well). Each and every model along with all its variations was tested using a 10 fold cross validation. Other parameters were copied over from CNM06 paper (like C values for SVM).

Decision Trees: I used two criterion variations of Decision Trees to run my analysis: Entropy, and Gini. I used a max depth parameter list of [1, 4, 8, 16] to run Grid Search Cross Validation on. Doing so allowed me to find the best hyperparameters per model type and actually plug them into the algorithm for analysis.

SVMs: I used the following kernels for my SVM algorithms: Linear, Rbf, and Sigmoid. For each variation of the algorithm, I used the same C values as CNM06: [0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 2], and the following gamma values: [0.000001, 0.00001,0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000]. I then utilized Grid Search Cross Validation to let python find their best hyperparameters for each variation of the algorithm.

Random Forests: For Random Forests, I also used the Entropy and Gini criterion. For the maximum depth of each algorithm, I used the depth list of [1, 2, 4, 6, 8, 12, 16, 20] in my Grid Search Cross Validation. From there, I derive the best parameters and move on to running the algorithm.

2.2. Performance Metrics

The performance metrics I used for my report are Accuracy, Precision, Recall, F1-Score, and their Weighted Means. Accuracy is the first metric to be collected, and as a result returns how efficiently the algorithm can separate our binary classification data into their positive and negative classes:

$$Accuracy = \frac{TP + TN}{(TP + FP) + (TN + FN)}$$

Where TP denotes True Positives, TN denotes True Negatives, FP denotes False Positives, and FN denotes False Negatives. However, it's important to note that Accuracy can sometimes not be the best indicator of an algorithm's performance because imbalanced datasets with extreme class imbalance will be very easily classifiable. To combat this effect, I've also used my other three metrics, Precision, Recall, and F1-Score. In fact, according to the CNM06 paper, Precision is one of the classifiers that essentially ignore the distribution of a dataset. It does this by instead focusing on calculating its positive classifications only, and in doing so advertently creates a negatively classified points (aka points that are not labeled as positive) , which gives a more accurate reading (Caruana, 2006).

$$Precision = \frac{TP}{TP + FP}$$

Moving onto Recall, it can be explained as the sensitivity metric. Recall returns the quotient between the number of positive instances found, and the actual number of positive class points in existence. Much like precision, this is

also resilient to extreme class imbalance in datasets.

$$Recall = \frac{TP}{TP + FN}$$

F1-Score is the last metric I would like to introduce, and it's a function of Precision and Recall that incorporates both these metrics, returning the balance between them.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Thus we can see that using any one of these metrics alone can give us false pretenses on how well our algorithms are running. But since I employ all 4, and take their means to create a 5th metric, I can state our results will be very empirically sound.

2.3. Data Sets

The four datasets that I used for my algorithms were the "Dry Bean Dataset" (BEAN), "Adult Dataset" (ADULT), "Skin Segmentation Dataset" (SKIN), and "Letter Recognition dataset" (LETTER). One Hot Encoding had to be utilized for the BEAN, ADULT, and LETTER datasets, while the SKIN dataset already had a binary classification column, so that one was left untouched. There were also extensive data cleaning methods performed on ADULT, from assigning variables to string entries, dropping any rows with NaN values, assigning functions to universalize attribute names into groups and then assign entries in groups as numerical values, and lastly assigning correct column names per dataset. The target variable for ADULT was its "Income" column, where entries of ">50k" were mapped to be a 0, and "<=50k" were labeled to be a 1. For SKIN, the target variable was an already listed "Y" column where it stored binary information on if that row of data was "skin colored" (1) or not (2). Next, the target variable for the BEAN dataset was it's "Bean Area" column, where values under the median of the area column were labeled as a 0, and values over it were labeled as a 1. Lastly, for the LETTER dataset, the target variable was the "Letter" column, where any letter from A-M was labeled as a 1 and any letter from N-Z was labeled as 0. With that being said, Table 1 shows the attribute number, ATTR#, test and training sizes, and %POS per dataset.

Table 1. Description of Problems

	BEAN	ADULT	SKIN	LETTER
ATTR#	16	14	3	16
Train Size 80/20	10,888	39,073	16,000	40,688
Test Size 80/20	2,722	9,768	4,000	10,172
Train Size 70/30	9,527	34,189	14,000	35,602
Test Size 70/30	4,083	14,652	6,000	15,204
Train Size 60/40	8,166	29,305	12,000	30,516
Test Size 60/40	5,444	19,536	8,000	20,344
Train Size 50/50	6,805	24,421	10,000	25,430
Test Size 50/50	6,805	24,421	10,000	25,430
%POS	50%	24%	50%	29%

3. Difference In Training Framework

At the beginning of my report I noted that the way I spit and trained my data was different from CNM06 and I'd like to address that here. In CNM06, their method randomly takes 5000 attributes from each dataset, and trains a portion of it, (say 4000) and tests the rest (1000) per iteration. Doing this created a method where bigger algorithms would not skew the performance metrics positively due to more data being present.

However, that's not what I did in my research. What I decided to do was per iteration, split my dataset into 4 different chunks. 80/20, 70/30, 60/40 and 50/50 where the first percent is for training and the second is for testing. Doing so, along with the 10 trials I am already doing, essentially increased my findings eight-fold, and gave me much more data to work with. I believe this set up gave my results much more statistical power when it came to comparing algorithm performance. This would also allow me to see what effect, if any, an increased training size would have on the performance metrics, which I logged in a heatmap titled "Figure 1" in the Appendix.

4. Performance

4.1 Performance Per Problem

Table 2 depicts the resulting *test* set performances per algorithm-problem combination. Table 4, shown in the Appendix, shows the *training* set performance under the same conditions. In both tables, the rows present the mean performance (weighted average across the four metrics and splits) for all algorithms (each with their kernel and criterion variations). While the columns represent each dataset the algorithms were performed on. A "Mean" column is also added to these tables where it computes the mean performance per algorithm across the four datasets. A bold score is present per column which indicates which algorithm performed best per dataset. Asterisks (*) can also be seen next to scores whose metric performance were not drastically different from the top scoring one.

Table 2. Performance Per Problem (Test)

Model	BEAN	ADULT	LETTER	SKIN	MEAN
DT (Ent)	0.993*	0.767	0.742	0.987*	0.872
DT (Gini)	0.993*	0.767	0.741	0.986*	0.871
SVM (Lin)	0.992*	0.778*	0.734	0.954	0.865
SVM (Rbf)	0.988*	**0.779**	**0.942**	**0.999**	**0.927**
SVM (Sig)	0.938	0.769	0.695	0.918	0.831
RF (Gini)	**0.993**	0.775*	0.817	0.986*	0.893*
RF (Ent)	0.993*	0.776*	0.816	0.985*	0.892*

Analysing the actual performances, we see in Table 2 that the SVM algorithm with an RBF kernel performed the best for nearly all datasets with varying positivity balances. The RF algorithm (with both Entropy and Gini variations) came in close behind where SVM was first, except in the BEAN dataset where their performances were switched, and RF was first with SVM being second. The two different versions of Random Forests had a negligible difference of +-0.001, while the performance variations across SVM kernels were more drastic. Decision Trees, while not performing *badly*, came in last in nearly all trials and datasets. Their algorithm variations were extremely similar if not identical across the board. An important thing to note however,

is that we do not see better performance for datasets with larger sample sizes, which was something the CNM06 paper stated they avoided by sampling 5000 datasets for all algorithms. In my research, I found that datasets as big as 50,000 attributes were classified better *and* worse with different algorithms than datasets much smaller than that. Thus I can state that there seems to be no proof that higher dataset size impacts algorithm performance positively.

4.2 Performance Per Metric

Similar to the previous section, Table 3 in this section analyzes the performance of each algorithm, but per dataframe. Instead, it shows the performance of the algorithms across all trials, splits, and datasets in the 4 metrics described above. PREC denotes Precision, RCL denotes Recall, F1 denotes F1-Score, and ACC denotes Accuracy.

Table 3. Performance Per Metric (Test)

Model	PREC	RCL	F1	ACC	MEAN
DT (Ent)	0.848	0.858	0.862	0.887	0.864
DT (Gini)	0.879	0.858	0.863	0.888	0.872
SVM (Lin)	0.862	0.851	0.859	0.879	0.863
SVM (Rbf)	**0.929**	**0.915**	**0.921**	**0.941**	**0.927**
SVM (Sig)	0.835	0.815	0.823	0.849	0.831
RF (Gini)	0.898	0.881*	0.885	0.908*	0.893*
RF (Ent)	0.897	0.881*	0.885	0.908*	0.893*

Once again, we see the SVM algorithm with the Rbf kernel sweeps the floor across all 4 metrics, scoring high in the 90th percentiles of each category. Random Forests, once again are a very close second, scoring in the high 80's and low 90's as well, with Decision Trees coming in last here as well. Lastly, in the appendix will be a Table 5, which has logged the same analysis except using the *training* sets instead of the testing.

5. Discussion

To conclude this report, we see that SVM in general dwarfs all other algorithms and in nearly every case, Random Forests are a close second tailing SVM's ever so slightly in all four metrics, and Decision trees come last in metric and problem performance. We can also see that as the training sample gets smaller, the testing performance gets slightly worse per 10% decrease in training size. (see Figure 1 in Appendix). Overall this report's findings seem not to deviate much from CNM06's publication, where their RF's outperformed their SVMs with a range of 0.02-0.05 in performance metrics on average. I suspect this is due to the countless other parameters they were able to employ per model which I couldn't do for this project, but seeing as how their SVM and RF drastically outperformed their DT, it's nice to see my research follow a similar pattern where SVM and RF are racing on top while DTs seem to be at the bottom of the list in performance metrics. Since SVM was best performing, I've also gone ahead and made heatmaps analyzing their hyperparameter performance per dataset (Figures 2-5 in Appendix).

6. Bonus

My report met all the requirements, and exceeded them. It employed 10 trials (essentially doubling my calculation times and statistical power), as well as employed an entirely different data splitting system which gave 4 times more results than just the single

5000 split method. By the end of my collection efforts I had run through 480 trials (over 8 times the data collected than needed)! Lastly, I did more secondary results than needed by including my heatmaps. So, I do hope that extra trials, metrics, and secondary results are sufficient for some extra credit!

Acknowledgments

Firstly, I'd like to thank Professor Fleischer for teaching a great and interesting quarter! His teaching methods, thorough lectures, availability on Piazza and OH, and comprehensive homework assignments were the reason I was able to understand Machine Learning and Supervised Algorithms as well as I do today (even if Midterm 2 was a little bit of a curve ball :-) lol). I'd also like to take a moment to thank the best TA I have ever had, Abdullah. He was always there for me in office hours, discussion, and piazza and his friendship, guidance, and helping hand were the primary reason I passed all my homeworks and assignments with flying colors. Thank you!! Lastly I'd like to thank the UCI Machine Learning Repo, and Sklearn for if it wasn't for their open source and free datasets/code packages, assignments like this wouldn't be possible for anyone to recreate!

References

Caruana, Rich., & Niculescu-Mizil, Alexandru. (2006). *An Empirical Comparison of Supervised Learning Algorithms.* Department of Computer Science, Cornell University, Ithaca. https://www.cs.cornell.edu/~caruana/ctp/ct.papers/caruana.icml06.pdf

Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Kohavi R, (1996). *Scaling Up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid, Proceedings of the Second International Conference on Knowledge Discovery and Data Mining.* http://robotics.stanford.edu/~ronnyk/nbtree.pdf

Koklu, M. and Ozkan, I. A., (2020). *Multiclass Classification of Dry Beans Using Computer Vision and Machine Learning Techniques.* Computers and Electronics in Agriculture. https://doi.org/10.1016/j.compag.2020.105507

P. W. Frey and D. J. Slate, (1991). *Letter Recognition Using Holland-style Adaptive Classifiers*. http://rexa.info/paper/20ffa3d5e05b868d49b102b97c2a60037e85f34e

Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011. http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html

Appendix

Note: Since my project went beyond the required data collection methods, this also means that there was a *plethora* of data I had to sift through with 4 different splits, per dataset and algorithm per 1/10 trials. For that reason, I physically can not fit my raw data into any tables big enough on this document. (The raw metrics data per ONE dataset is 71 rows by 16 columns). As such, I will instead provide links to my raw testing data here (Professor said this was ok):

Raw Bean Data:
https://docs.google.com/spreadsheets/d/1XGpjw7vSWOkdwfLucsdEMIGItiJIfM4mivapDdKvIpk/edit?usp=sharing

Raw Skin Data:
https://docs.google.com/spreadsheets/d/1dwnaXgK1HWOb6UxPF2F5tOtTuTn9S6oKlmg7IUE9SLE/edit?usp=sharing

Raw Adult Data:
https://docs.google.com/spreadsheets/d/1jV_WhkNgm2m68jz0_WqVocgQZICR3zpkpjU50PKWRP0/edit?usp=sharing

Raw Letter Data:
https://docs.google.com/spreadsheets/d/1Q3XKkdi5b5YM1mbr5XkM0-ni176siDJFV4tozt47SqM/edit?usp=sharing

Table 4. Performance Per Problem (Train)

Model	BEAN	ADULT	LETTER	SKIN	MEAN
DT (Ent)	1*	0.765	0.748	0.987*	0.875
DT (Gini)	1*	0.765	0.746*	0.987*	0.874
SVM (Lin)	1*	0.774	0.732	0.955	0.865
SVM (Rbf)	0.999*	**0.788**	**0.985**	**0.999**	**0.943**
SVM (Sig)	0.939	0.762	0.675	0.927	0.826
RF (Gini)	**1**	0.776	0.822	0.986*	0.896
RF (Ent)	1*	0.776	0.823	0.986*	0.896

Table 5. Performance Per Metric (Train)

Model	PREC	RCL	F1	ACC	MEAN
DT (Ent)	0.886	0.861	0.863	0.891	0.875
DT (Gini)	0.887	0.861	0.863	0.891	0.876
SVM (Lin)	0.865	0.859	0.857	0.881	0.866
SVM (Rbf)	**0.938**	**0.936**	**0.938**	**0.958**	**0.943**
SVM (Sig)	0.836	0.818	0.819	0.849	0.831
RF (Gini)	0.901*	0.885	0.886	0.912	0.896
RF (Ent)	0.902*	0.884	0.886	0.913	0.896

Table 6. P-Values Per Problem (Testing)

PROBLEM	P-VALUE
BEAN	0.0000162
ADULT	0.0000454
SKIN	0.0038890
LETTER	0.0000049

Table 7. P-Values Per Metric (Testing)

METRIC	P-VALUE
DT (Entropy)	0.0705
DT (Gini)	0.3353
SVIM (Linear)	0.0007
SVM (Rbf)	0.0001
SVM (Sigmoid)	0.0004
RF (Gini)	0.0071
RF (Entropy)	0.0056

Figures

Figure 1. Heatmap of Partition Performance per Model

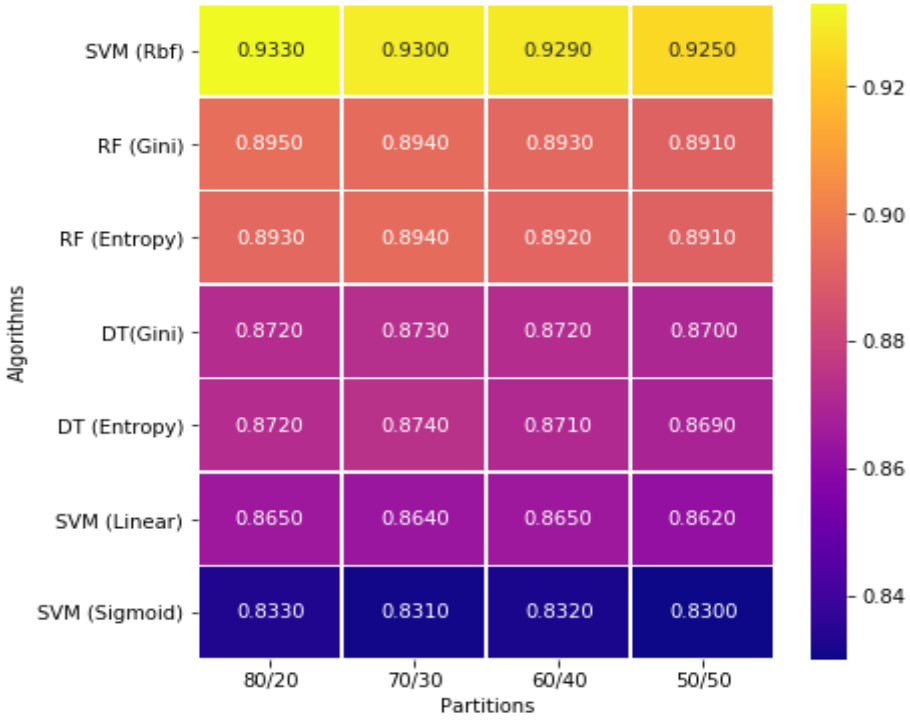

Figure 2. SVM heatmap for Bean Data

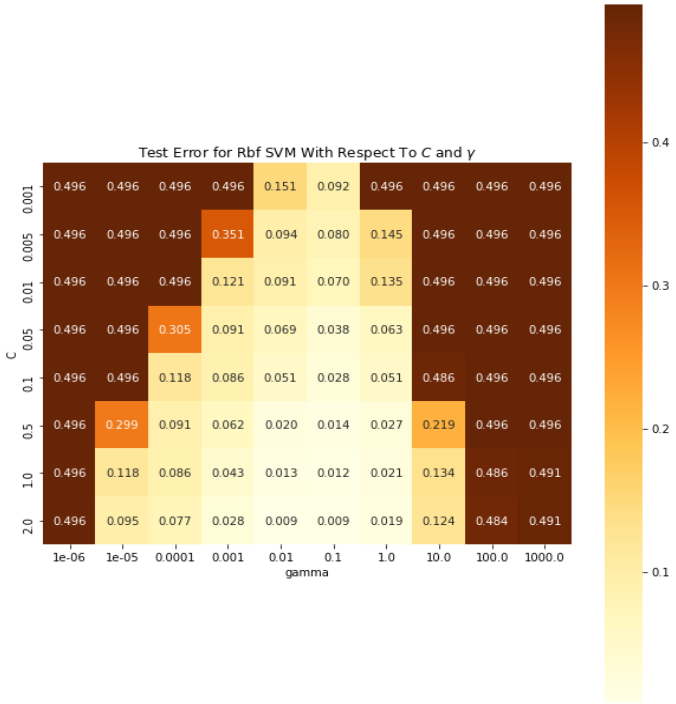

Figure 3. SVM heatmap for Skin Data

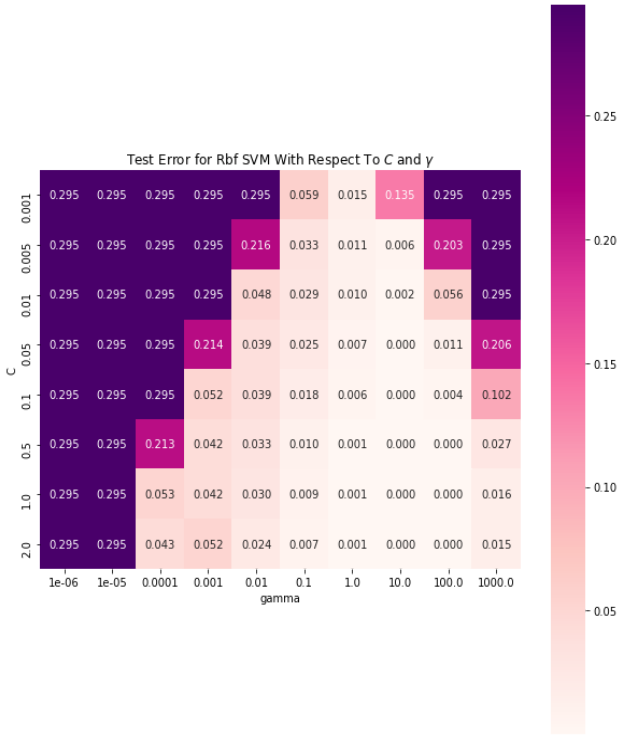

Figure 4. SVM heatmap for Adult Data

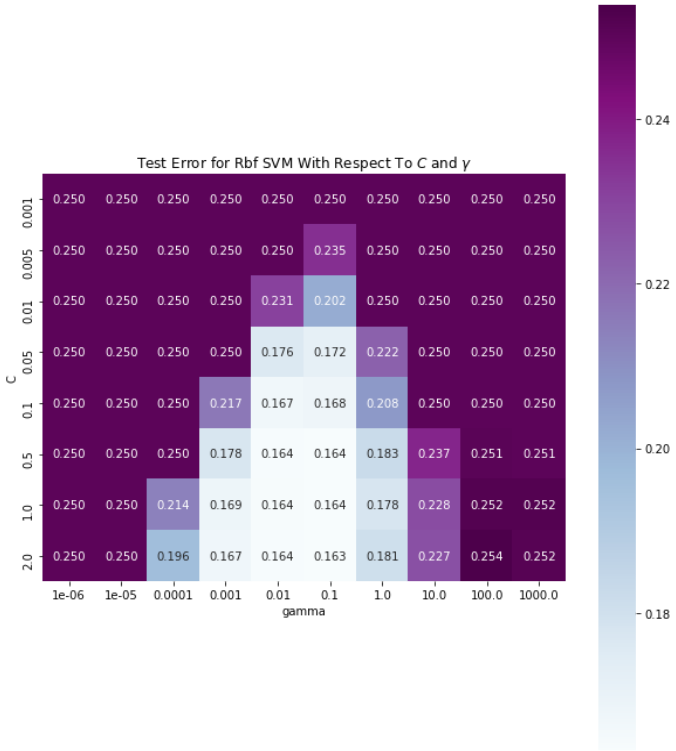

Figure 5. SVM heatmap for Letter Data