Final Project Report: “Shrooms and Tunes”

Sean Maden

CS545, Fall Quarter 2018

12/7/18

Background and Introduction

Dataset Overview

For my final project, I used the UCI Mushroom dataset (“UCI Machine Learning Repository: Mushroom Data Set,” n.d.). This data was first published in a 1987 Ph.D. dissertation at UC Irvine, and it has since been used in a number of publications concerning statistical modeling and machine learning (Iba, Wogulis, & Langley, 1988; Jeffrey Curtis Schlimmer, 1987). The dataset comprises of hypothetical/simulated mushroom samples synthesized from one of 23 known real mushroom species descriptions, from the families *Agaricus* or *Lepiota*. Mushroom species information was obtained and informed by the official field guide from Audubon Society (Knopf, 1981). In its original form, each sample/instance in the data included an edibility classifier that was either “edible”, “poisonous”, or “unknown”, along with 22 features corresponding to species characteristics (eg. cap shape, ring number, etc.). In all current available forms of the dataset that I could find, mushrooms of “unknown” class have already been collapsed into the “poisonous” category. Data variables are entirely discrete/categorical, including numeric information (eg. variable “ring number” is “none”, “one”, or “two”).

Algorithms and Random Decision Forest Description

To analyze the mushrooms dataset, I opted to study the performance of runs applying various hyperparameter configurations of three algorithm classes: random decision forest, support vector machine (SVM), and neural network. Of these, random forest was not discussed extensively in class, so here I will attempt to summarize the method and how it works. Random decision forest is a type of supervised ensemble learning algorithm that uses decision “trees”, or node sets of many nodes each representing individual tests using random feature subsets form the input data. The random component of feature selection is used to avoid overfitting to training data. Randomization can be performed using one of several possible methods, including “bagging”, or a type of random sample selection in bootstrapping with replacement, which was used here. Results from a set of decision trees may be combined to make a prediction in several ways, depending on whether its application is for a classification or regression problem. In the present study, I used the *randomForest* module with mostly default arguments, which resulted in applying the regression strategy. In principle, a random decision forest selects the best feature from random subsets of the underlying feature space in the data. In application, this can result in a better trained model. Feature importance is also readily estimated from random forests by evaluating how trees using a given feature impact forest performance overall.

Methods: Data Preprocessing

Dataset variables, including numeric values, are entirely discrete or categorical. Because many machine learning methods require numeric input data, I converted all variables using the one hot-encoding (or OHE) strategy. This means each variable with more than two possible value levels was converted into a new set of binary features, where each new feature corresponds to one of the original value levels. Features with two levels were binarized as 0 or 1. OHE has the downside of greatly increasing the dimensionality of the dataset, and subsequent risk of overfitting. Here, this limitation is mitigated by the fact the original data has relatively low dimensionality (N = 22 features). OHE resulted in a total of 112 features. After transformation, data was split evenly into equally sized training and test subsets, at random, where each subset included identical sample class frequencies to the original dataset. The OHE-encoded features were themselves assessed to ensure they occurred at similar frequencies in test and training data, and to ensure nonzero frequencies of each value level across subsets. In Experiment 2, I excluded 50% and 75% of the poisonous-classified mushrooms at random, and I then redistributed data into training and test sets while preserving sample group frequency in each. The same randomized data subsets were used across tests of optimal models from each algorithm class.

Methods: Outline of Experiments

The overall aim of my experiments was to evaluate whether certain mushroom features are most important for determining their edibility, and to further compare my findings to prior conclusions drawn from this data (see Appendix). In short, these prior conclusions include that “no simple rule exists” for determining poisonous mushrooms, and, alternatively, that four discrete rules exist based on prior analysis of the data (Appendix and Discussion). To assess these prior conclusions in experiment 1, I initially trained, selected, and assessed several model classes and hyperparameter sets on the data across 3 experiments. In experiment 2, I tested the best-performing model of each class on randomized subsets of the data, excluding either 50% or 75% of poisonous samples. Finally in experiment 3, I discuss to what extent the prior conclusions are supported or not supported by my results.

For experiment 1 I devised, trained, and evaluated hyperparameter sets across runs using three aforementioned supervised algorithm classes. For each algorithm class, I compared runs using at least 4 variations in important hyperparameters (Table 1, below). Outputs for model assessment included performance metrics in test and training data (e.g. error, accuracy, loss, etc.), as well as variable weight and importance

Table 1. Experiment 1 design, final model hyperparameters in runs trained and tested.

Model Type	Parameters Types to Test	Parameter Values for Experiment 1
Random Forest	A. Number of Trees (NT)	1. NT = 10
		2. NT = 50
		3. NT = 100
		4. NT = 500
		5. NT = 1,000
		6. NT = 5,000
SVM	A. Kernel Type (KT);	1. KT = linear, WF = none;
	B. Weight Filter (WF)	2. KT = gaussian, WF = none;
		3. KT = linear, WF = top 50%;
		4. KT = gaussian, WF = top 50%
Neural Network	A. Num. Hidden Nodes (HN), per layer;	1. HN = 5, HL = 1
	B. Num. Hidden Layers (HL)	2. HN = 15, HL = 1
	C. Batch Size (B, equals num. trainset instances, unless otherwise stated)	3. HN = 5, HL = 2
		4. HN = 15, HL = 2
		5. HN = 5, HL = 3
		6. HN = 15, HL = 3
		7. HN = 25, HL = 3
		8. HN = 50, HL = 3
		9. HN = 50, HL = 3
		10. HN = 5, HL = 1, B = 100
		11. HN = 5, HL = 1, B = 1000
		12. HN = 5, HL = 1, B = 5000

for SVM and random forest models (Figs 1-3), and performance over epoch history for neural networks (Figs 4 and 5).

Methods: Coding Strategy

I utilized established and widely used R libraries for model tuning and evaluation. For SVM optimization, I used *e1071* (Meyer et al., 2018). For fitting neural networks, I used *keras* package (Allaire et al., 2018). For fitting random decision forests, I used *randomForest* (Cutler & Wiener, 2018). Performance metric measures were either coded by hand or measured using convenience functions and output returned by the aforementioned modules.

Experiments and Results

Experiment 1 Results

This experiment used the entire UCI Mushrooms dataset, where 50% of all samples were divided into training and test data subsets, while preserving frequencies of classifications (“edible” or “poisonous”) and approximate frequencies of OHE-transformed variables.

Experiment 1 Part 1. Random forest run results

In total, six runs with different hyperparameter sets were tested using the random decision forest algorithm, where each run had from 10 to 5000 decision trees (Table 2). I evaluated the performance parameters returned (overall error and class-specific error) as well as accuracy on predicting sample classes in test data. Interestingly, all runs showed very high performance overall, where the only exception was a slight error rate

(0.001) at the very lowest number of trees tested (N = 10 trees, Table 2). Accuracy in predicting test set classes was perfect (1) across all runs. Notably, the top six variables of greatest importance from each model did vary somewhat among runs from low-to-high tree counts. Certain variables were recurrent in their importance across all runs,

Table 2. Experiment 1 part 1, random forest run results, with n forests ranging from 10-5000. Top six most important variables are indicated for each model fitted.

Run	N Trees	Error Rate	Class Err. ("edible")	Class Err. ("poisonous")	Test Set Acc.	Top Six Variables (by importance)
1	10	0.05	0.001	0	1	odor.n;bruises;stalksurfabovering.s;stalkroot.e;odor.f;pop.v
2	50	0	0	0	1	odor.n;stalksurfbelowring.k;odor.f;gillcol.b;gillsize;ringtype.p
3	100	0	0	0	1	odor.n;gillsize;odor.f;gillcol.b;stalksurfbelowring.k;ringtype.p
4	500	0	0	0	1	odor.n;gillsize;odor.f;gillcol.b;stalksurfabovering.k;ringtype.p
5	1000	0	0	0	1	odor.n;odor.f;gillsize;gillcol.b;stalksurfabovering.k;ringtype.p
6	5000	0	0	0	1	odor.n;gillsize;odor.f;gillcol.b;stalksurfabovering.k;stalksurfbelowring.k

and later runs with the highest tree counts seemed to reach consensus on the top important variables, compared to runs with low tree counts (Figures 1 and 2, and Table 2 last column). For this reason, I favored the final run (with N = 5000 trees) as optimal, even though almost every run performed very well.

Experiment 1 Part 2. SVM model run results

I tested four runs with different hyperparameters using an SVM optimization algorithm, where runs included either linear or polynomial kernel function, and either none or a top-50% weight filter applied to retained model variables. As with the random decision forest results, all runs showed very high performance, including perfect accuracy on the test data in each case. Interestingly, there was improvement on runs with the polynomial kernel, where the run applying the top-50% weight filter showed precision of 1 while the run without a weight filter showed precision of 0.84. It's interesting to note that the top six most important variables (chosen by absolute value, Figure 3) were more similar in runs with the same kernel type, regardless of the presence of a weight filter. Despite these differences, we again see recurrence of important variables across runs that pertain to odor, stalk surface type, and gill size, all of which were also identified as important from random forest run results. Owing to the improvement with the weight filter for the polynormal kernel, but the better overall performance of linear kernel runs, I selected the third run hyperparameters (linear kernel with top-50% weight filter) for inclusion in experiment 2.

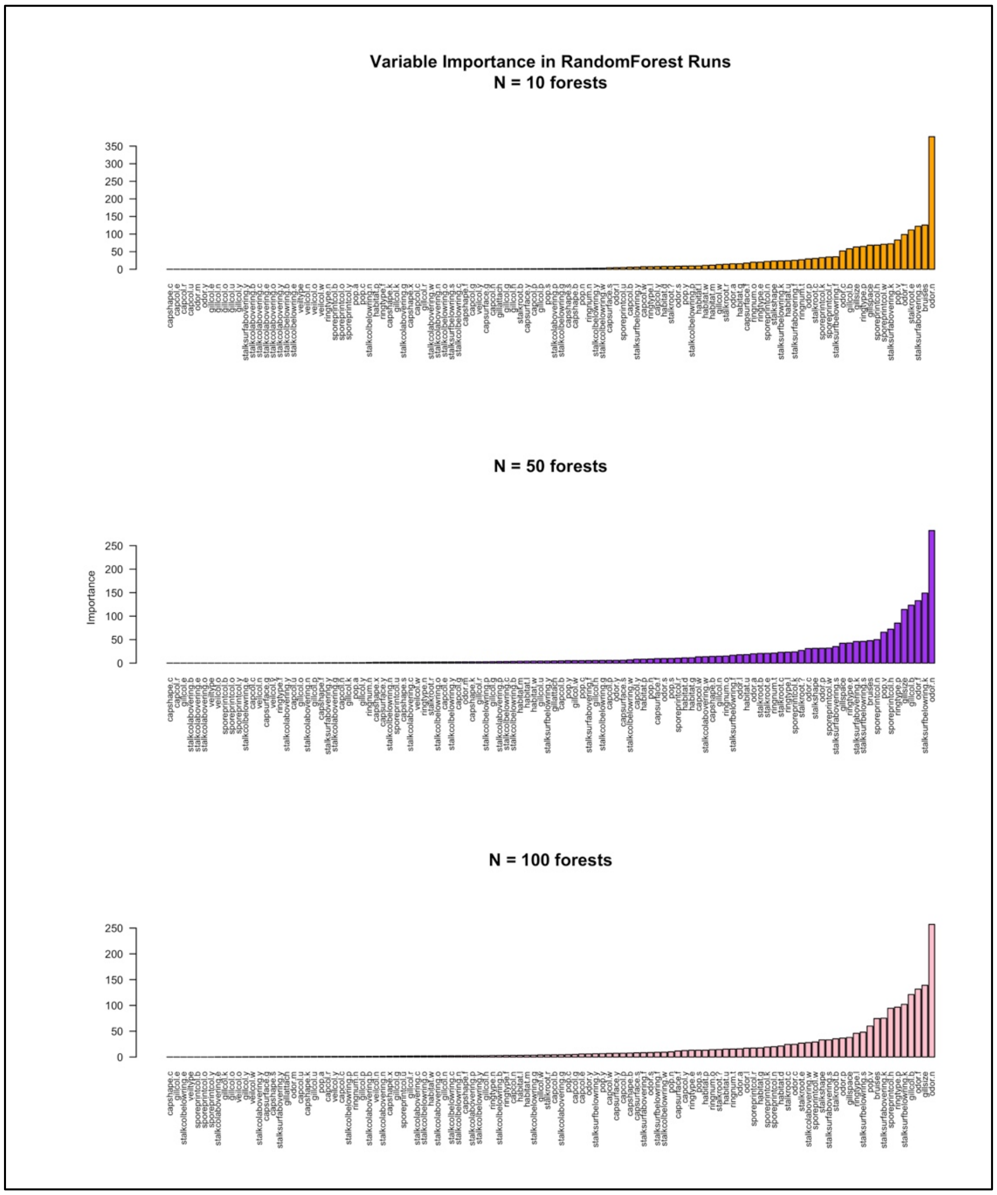

Figure 1. All variables sorted by increasing relative importance, from runs 1 to 3 testing random forest model hyperparameters (num. trees 10 to 100).

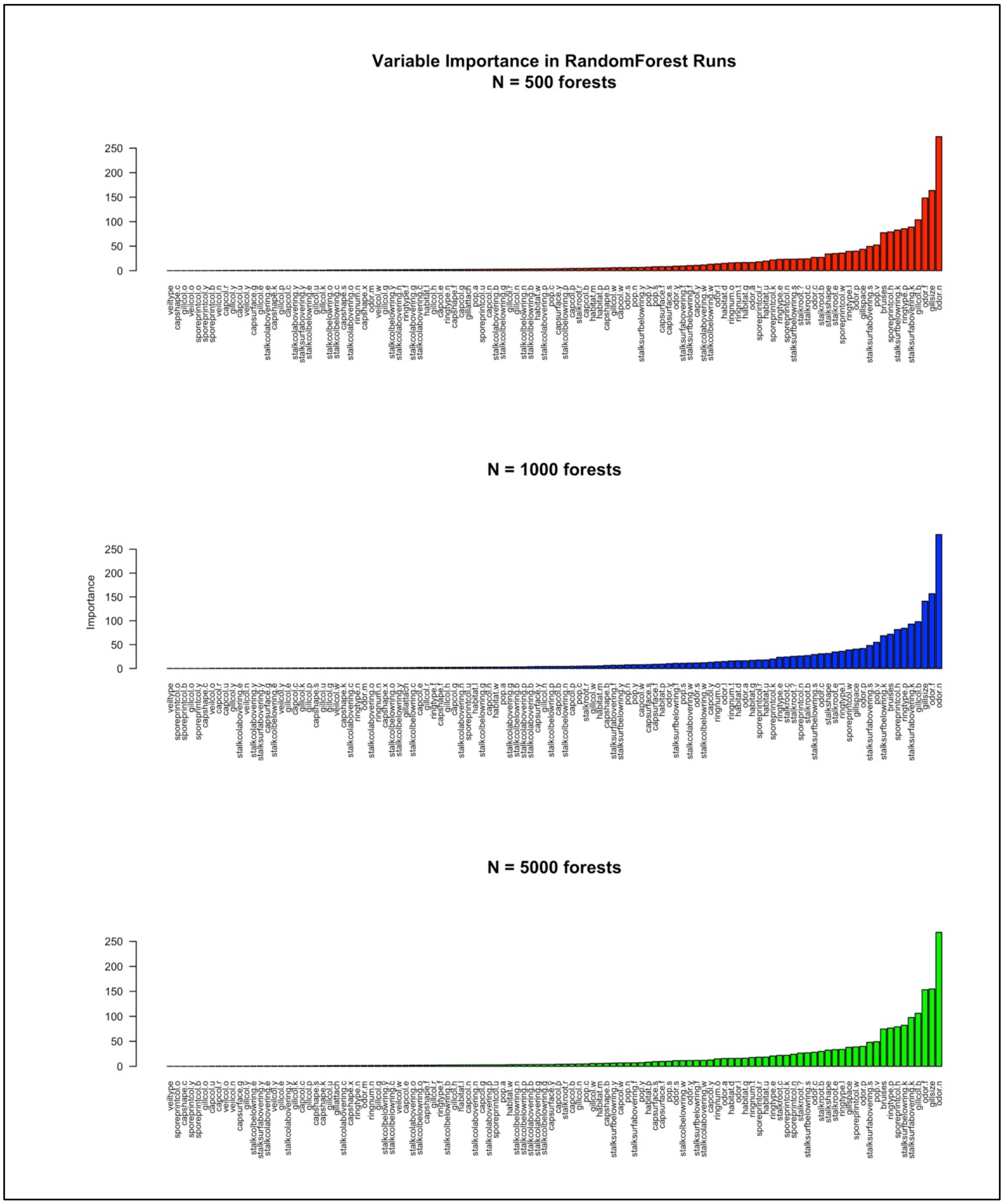

Figure 2. All variables sorted by increasing relative importance, from runs 4 to 6 testing random forest model hyperparameters (num. trees 500 to 5000).

Table 3. Experiment 1 part 2, SVM run results varying kernel type (linear or polynomial) and top weight filter (none or top 50% of variables only). Top six variables were identified by absolute importance.

Run	Kernel	Weight Filt.	Prec.	Recall	Top Six Vars (absolute importance)
1	linear	none	1	1	odor.l;odor.a;odor.n;sporeprintcol.r;odor.c;stalksurfaceabovering.k
2	polynomial	none	0.84	1	odor.n;gillsize;ringtype.p;odor.f;gillcol.b;stalksurfaceabovering.k
3	linear	top50perc	1	1	odor.l;odor.a;odor.n;sporeprintcol.r;odor.c;stalksurfabovering.k
4	polynomial	top50perc	1	1	odor.n;gillsize;ringtype.p;odor.f;gillcol.b;stalksurfaceabovering.k

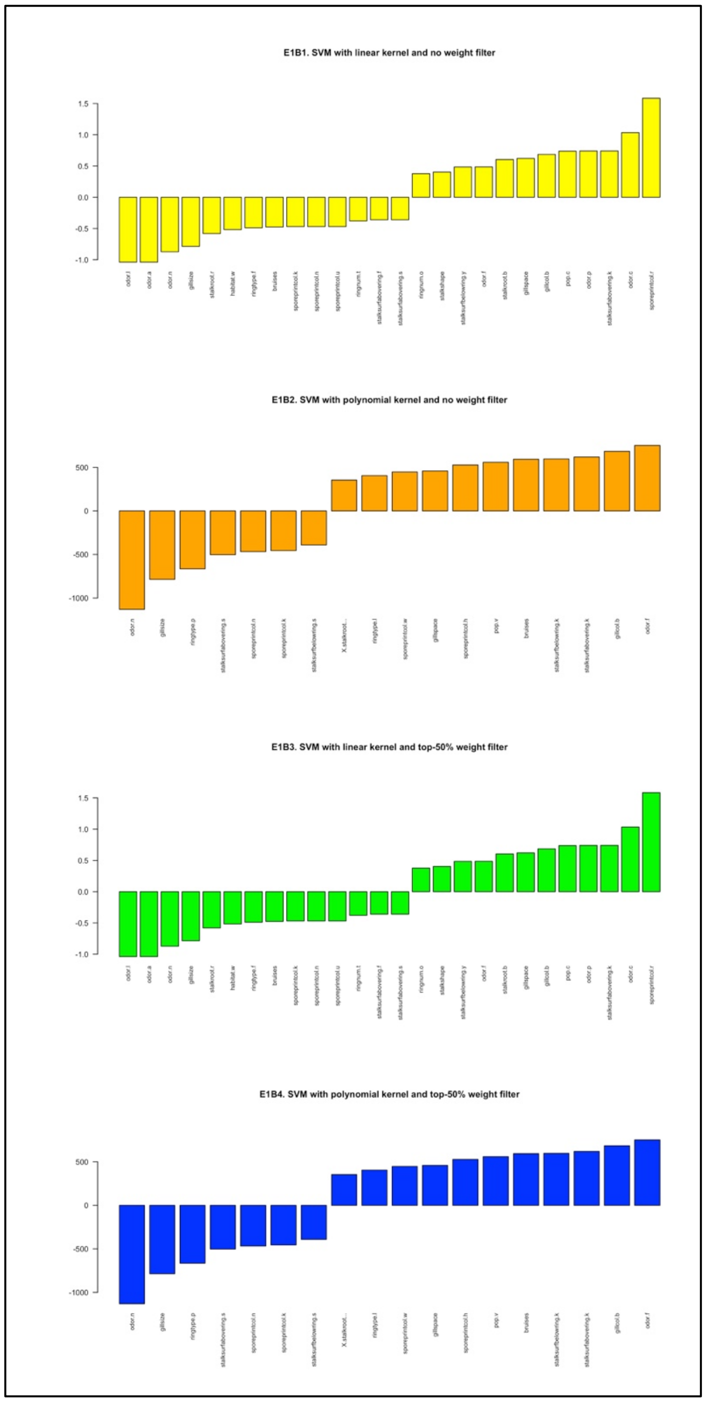

Figure 3. Relative importance, or model-assigned weights, of a subset of variables, across all SVM runs. Only variables >=1 Std. Dev. from the mean of all feature weight values were retained, thus only variables of relatively high absolute importance are shown. Because class categorization was binarized as 1 for poisonous and 0 for edible, important variables with positive weights are considered descriptive for the former category, while important variables with negative weights are considered descriptive for the latter category.

Experiment 1 Part 3. Neural networks model run results

Because the *keras* package enables very rapid computation for neural network fitting, I opted to test a wide variety of hyperparameter sets. I varied the number of hidden nodes (HN), number of hidden layers (HL), and size of training data batch (Tables 1 and 4). In total, I tested 12 hyperparameter sets, where the first 9 included the same batch size (equal to the total size of the training data set) and the remaining 3 varied the batch size, using what appeared to be both the most consistent and highest performing HN+HL hyperparameter set. I fitted each neural network across 50 epochs. Notably, most runs did not show consistent improvement of accuracy on test data across all 50 epochs, and many showed declining accuracy to varying degrees across some or most epochs (Fig 4). This may be indicative of overfitting, and needs to be taken into consideration when evaluating which model showed the best performance relative to others. Table 4 reports the summary evaluations of test data accuracy and loss across the history of each trained model, but models that show moderate or relatively high performance on these metrics did not necessarily show quality across epochs. For instance, run #5 (HN = 5, HL = 3) showed steep decline in test/validation data accuracy, and increase in loss, across epochs (Fig 4). This is evidence of substantial overfitting to the training data set. By contrast, the first run (Fig 5) showed better overall test data accuracy (0.85) and loss (0.47), with more modest decline in test set accuracy over epochs. I selected the run 1 hyperparameters for experiments varying batch size.

Batch size is specified as an argument in the *keras* neural network function, pertaining to the sizes of randomized batches drawn from the training data used in an epoch. It was interesting to note improved summary performance when batch size was reduced to 100 or 1000, where the latter has substantially better accuracy (0.9) and loss (0.21) than run 1. This indicates how it can be helpful to withhold training data samples when training a neural network, so the model is not fitted to all available data at once. I chose the hyperparameters in run 11 (HN = 5, HL = 1, B = 1000) for inclusion in experiment 2.

Table 4. Experiment 1 part 3, neural networks run results, varying number of hidden nodes (HN) and hidden layer (HL) quantity across runs, and varying batch size with HN = 5 and HL = 1. Loss computed as "categorical crossentropy", test set accuracy and loss computed as a summary across epochs using "model %>% evaluate()".

Run	Num. HN	Hum. HL	Batch Size	Test Acc.	Test Loss
1	5	1	nrow_train	0.85	0.47
2	15	1	nrow_train	0.9	0.35
3	5	2	nrow_train	0.78	0.49
4	15	2	nrow_train	0.88	0.3
5	5	3	nrow_train	0.73	0.56
6	15	3	nrow_train	0.92	0.24
7	25	3	nrow_train	0.9	0.32
8	50	3	nrow_train	0.9	0.41
9	50	5	nrow_train	0.91	43
10	5	1	100	0.9	1.52
11	5	1	1000	0.9	0.21
12	5	1	5000	0.83	0.41

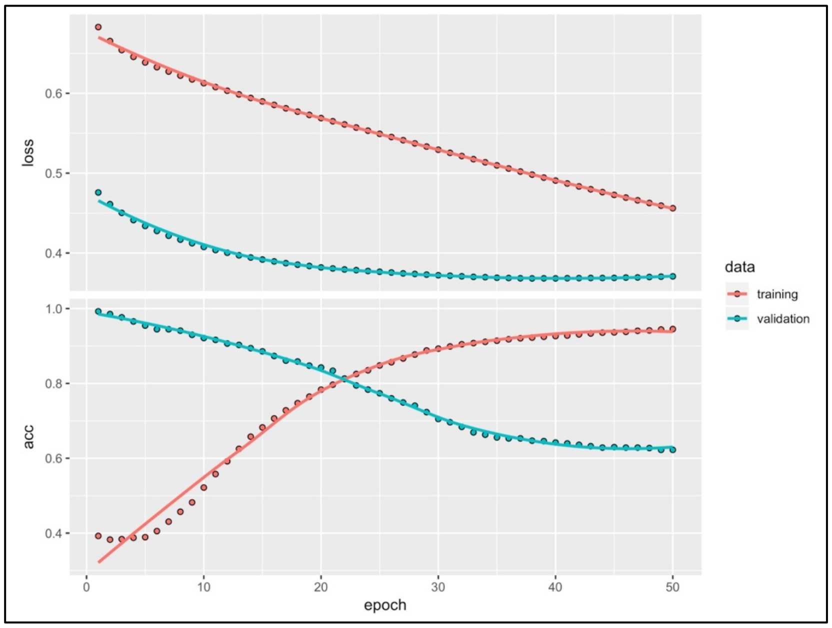

Figure 4. Performance (loss and accuracy in test/validation and training data sets) over 50 epochs for run 5 (HN = 5 nodes, HL = 3 layers). Drastic decline in validation over epochs may indicate overfitting of network to training data subset.

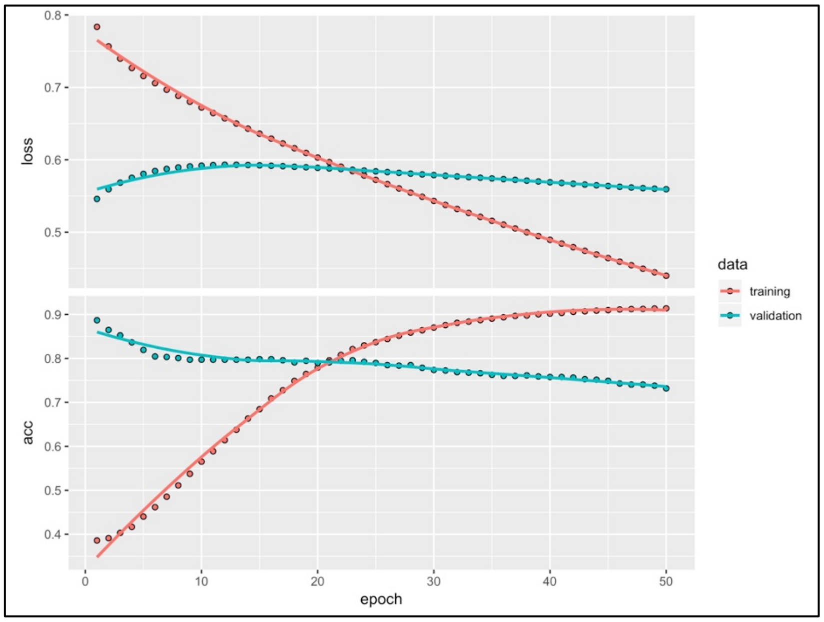

Figure 5. Performance (loss and accuracy in test/validation and training data sets) over 50 epochs for run 1 (HN = 5 nodes, HL = 1 layers). While model accuracy is relatively and consistently high, gradual but steady decline in test set accuracy over epochs indicates some model overfitting.

Experiment 2 Results

For experiment 2, I evaluated the single best performing parameter set from each of the 3 classes of algorithms tested in experiment 1. For random decision forests class, this was the run with N = 5,000 decision trees, for SVM this was the linear kernel with top-50% variable weight filter, and for neural networks, this was the network with 5 hidden nodes, 1 hidden layer, and batch size of 1,000 samples. Each of these three models was trained and evaluated on 2 subsets of the UCI Mushrooms data, either reducing the total number of poisonous samples to 50% or 25% of its initial amount. In both cases, data was re-distributed at random into training and test data, preserving sample class frequency in each.

Interestingly, all 3 models showed consistent performance after 50% of poisonous samples were removed, compared to results using the entire dataset (Table 5), where the neural network showed slight decline in accuracy (from 0.92 to 0.90) and

Table 5. Experiment 2 results, summarizing performances of chosen optimal hyperparameter sets across 3 model classes using either all "poisonous" instances, or from retaining only 50% or 25% of poisonous instances, at random (same sample sets used to evaluate each model type, respectively).

Model Class	Hyperparameters	Performance		
		All	50%	25%
Random Forest	nt=5000	err.rate=0;	err.rate=0;	err.rate=0;
		class.err0=0;	class.err0=0;	class.err0=0;
		class.err1=0;	class.err1=0;	class.err1=0;
		test.acc=1	test.acc=1	test.acc=0.99
SVM	k=linear;	precision=1;	precision=1;	precision=1;
	wfilt=top50	recall=1	recall=1	recall=1
Neural Network	nh=5;	test.acc=0.92;	test.acc=0.90;	test.acc=0.48;
	nl=1;	test.loss=0.23	test.loss=1.25	test.loss=8.25
	nbatch=100			

increase in loss (0.23 to 1.25). When poisonous samples were reduced to 25% of their initial number, the random forest and neural network model both showed diminished performance to varying degrees, while the SVM model maintained perfect precision and recall on test data.

Overall, the neural network was least consistent across data subset experiments, showing very substantial decrease in performance (test summary accuracy = 0.48, summary loss = 8.25) with 25% of poisonous samples included. By contrast, SVM and random forest models were extremely consistent, where SVM showed no decrease in performance while the random forest model showed very slight decline in accuracy (from 1 to 0.99) with no change in error rate.

Experiment 3 Results

The aim of experiment 3 is to test the results of experiments 1 and 2, including any discernible rules about variables and their relative importance, against two prior hypotheses (see Appendix): 1. That no simple rule exists for determining whether a mushroom is poisonous or edible; and 2. That a series of several discrete rules exist. Speaking to just the analysis results covered in experiments 1 and 2, there do seem to be a subset of variables that are relatively more important for discerning mushroom edibility in the UCI Mushrooms dataset, and this set of putative important variables (Table 7, below) is remarkably similar to the rules described in prior hypothesis 2 (see Discussion).

For random forest and SVM model types, it is straightforward to assess each variable's individual importance on the overall model performance, and I have shared to variables that seem to be most important in each case (Figs 1-3, Tables 2 and 3 final columns). For neural networks, assessing variable importance is less straightforward. As a neural network's configuration becomes more complex, so too can the ways in which inputs are disseminated across weights and between hidden layers to ultimately impact activation of the output layer. A very approximated way around this complicated issue is to systematically “remove” variables (e.g. set all sample values for that variable to 0) and assess how the neural network’s performance changes. Table 6 shows the results from removing the top 11 putative important variables, as informed by random forest and SVM results (see Tables 2, 3 and 7), and results from removing 11 variables selected at random. Intriguingly, using the version of the neural network from experiment 2 (HN = 5, HL = 1, B = 1,000) trained on the entire dataset, I did indeed observe a more substantial decrease in model performance when the set of putative important variables were removed, where summary test set accuracy decreased from 0.9 to 0.74, and loss increased from 0.23 to 0.55! This is compared to removing 11 variables at random, which resulted in a test accuracy 0.83 and loss 0.49. These results provide further evidence that the 11 putative important variables from random forest and SVM tests are also important for the performance of the neural network.

Table 6. Impact on neural network performance after removing either the 11 putative important variables, or 11 random variables, where network hyperparameters were the same as the model used in experiment 2.

Var's removed (N = 11)	Train Acc.	Train Loss	Test Acc.	Test Loss
Important variables	**0.73**	**0.55**	**0.74**	**0.55**
Random variables	**0.83**	**0.48**	**0.83**	**0.49**

Ultimately, this set of about 11 important variables largely reflects the rules proposed in primary hypothesis 2 (Table 7 and Appendix). That is, they pertain to similar feature states like odor, cap color, and print spot color in hypothesis 2 sub-rules 1 and 2 (Appendix). For instance, rule 1 shows >98% accuracy predicting poisonous mushrooms for odor levels **not** *anise* or *almond*, and I did in fact observe that odor feature levels *anise*, *almond*, and *none* were important for predicting edible mushrooms, while levels *foul* and *creosote* are important for predicting poisonous mushrooms. Furthermore, samples with green spore print color were important for predicting poisonous mushrooms, in both my results and in rule 2 (Fig 3). There doesn't seem to be complete consensus, however, as the features in rules 3 and 4 of hypothesis 2 did not occur among important variables in my results. It could be these features were marginally significant and just filtered out in my selection of most important variables, but are nonetheless important for the models I tested. Other features like gill size and ring shape do not appear in hypothesis 2 but do appear to be important from my analyses of model consensus.

Discussion and Conclusions

How did Different Models Behave?

Runs using SVM or random forest showed very high performance (low error, high accuracy, etc.) across hyperparameter sets, with some slight performance improvements when random forest tree count was increased or a top-50% SVM variable weight filter was introduced, respectively. Models with chosen hyperparameter sets from these classes also performed very well even when total poisonous instances was reduced to 50% or 25% of the initial sample count.

By contrast, neural network models showed underwhelming performance across Experiment 1 runs, and the best selected model showed very large declines in performance when poisonous samples were reduced, especially when they were reduced to 25% of the original total. Interestingly, the reduction of poisonous samples to 50% showed a modest but not huge decline in performance for the neural network. Assuming unknown-labeled mushrooms to be less informative than mushrooms labeled poisonous for predicting samples of poisonous class, it could be that, by chance, unknown mushroom instances were excluded from the poisonous instances at a higher rate than known poisonous instances with the 50% reduction. Unfortunately, I could not find a form of this dataset that included these original “unknown” labels, and so I could not rigorously test the effect of reducing unknown instances from the poisonous class.

What Can I Eat?

Vitally, the UCI Mushroom dataset is simulated based on real species descriptions, and it should not be considered a substitute for real field data from biological specimens! As such, *conclusions from this analysis should in no way be used to inform any actual decisions on what type of mushrooms it is ok to consume!* In considering this downside, I further find that, while this analysis does highlight some recurrent important mushroom traits pertaining to edibility, hypothesis 1 is still valid insofar as these new hypotheses need to actually be tested with real specimens before attaining any rigorous verdict on whether a simple rule truly exists for determining mushroom edibility.

Table 7. Final most important variables for “poisonous” and “edible” mushroom classification, based on results from all 3 tested model classes.

Category	Variable	Description
Poisonous	odor.f	foul odor
	odor.c	creosote odor
	gillcol.b	buff gill color
	stalksurfaceabovering.k	silky stalk surface, above ring
	stalksurfbelowring.k	silky stalk surface, below ring
	sporeprintcol.r	green spore print color
Edible	odor.n	no odor
	odor.l	anise odor
	odor.a	almond odor
	gillsize	broad gill size (not narrow)
	ringtype.p	pendant ring type

Possible Extensions to Analysis

The problem of how to determine a mushroom's edibility can be reduced to a problem of identifying the most important traits about a mushroom that can inform such a determination. The UCI Mushrooms dataset simulates a traditional kind of discrete biological field data that is often gathered from real specimens in the wild. However, there are a number of other conceivably useful datatypes and experiments whose analysis could yield just as, if not more meaningful information. For instance, it would be fascinating to simply train an adversarial network on just photographs of mushrooms as they might appear in the wild, and to observe the most important traits such a network extracts from these images for edibility classification. Without any other prior human input, the results of this analysis could be compared with important traits in the mushroom field data or simulated field data. Similarly, various biochemical assay profiles from organic specimens and derived compounds could be analyzed, and results of this analysis could pertain more directly to how a real mushroom is perceived and identified by a human in the field. Especially when combined with real field data, these types of approaches could provide compelling additional information.

Conclusions

This analysis studied the relative importance of various mushroom traits in determining their edibility. In it, I have applied a wide array of supervised learning algorithms using a variety of hyperparameter variations in a number of runs. Results from models of different classes ultimately showed a general consensus on the relative importance of a handful of roughly 11 traits for determining a mushroom's edibility (Table 7). Many of these traits and trait states further reflect rules postulated in the prior hypothesis 2, namely those pertaining to odor, gill color, and spore print color (Appendix 2). Overall, neural networks showed lower performance than SVM or random forest approaches, and this may be due to several factors. First, neural networks may not be well suited for one hot-encoded data types, and an alternative codification might be warranted for the UCI Mushrooms data features, which are all discrete or categorical. Second, neural networks may be more greatly and deleteriously impacted by the presence of unknown-classified mushrooms among poisonous mushroom samples. Finally, there may be alternate neural network hyperparameter set that was, and which could perform better and/or more consistently than the models I tested.

Works Cited

Allaire, J. J., Chollet, F., RStudio, Google, Tang, Y., Falbel, D., … Studer, M. (2018). keras: R Interface to “Keras” (Version 2.2.0). Retrieved from https://CRAN.R-project.org/package=keras

Cutler, F. original by L. B. and A., & Wiener, R. port by A. L. and M. (2018). randomForest: Breiman and Cutler’s Random Forests for Classification and Regression (Version 4.6-14). Retrieved from https://CRAN.R-project.org/package=randomForest

Iba, W., Wogulis, J., & Langley, P. (1988). Trading Off Simplicity and Coverage Incremental to Concept Learning. Elsevier; Dept Info and Comp Sci, UC Cali Irvine. Retrieved from https://www.westmont.edu/~iba/pubs/hillary-paper.pdf

Jeffrey Curtis Schlimmer. (1987). Concept Acquisition Through Representational Adjustment. University of California Irvine. Retrieved from https://dl.acm.org/citation.cfm?id=913421

Knopf, A. (1981). *The Audubon Society Field Guide to North American Mushrooms*. New York, NY: G. H. Lincoff.

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., Leisch, F., C++-code), C.-C. C. (libsvm, & C++-code), C.-C. L. (libsvm. (2018). e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien (Version 1.7-0). Retrieved from https://CRAN.R-project.org/package=e1071

Sing, T., Sander, O., Beerenwinkel, N., & Lengauer, T. (2015). ROCR: Visualizing the Performance of Scoring Classifiers (Version 1.0-7). Retrieved from https://CRAN.R-project.org/package=ROCR

UCI Machine Learning Repository: Mushroom Data Set. (n.d.). Retrieved October 31, 2018, from https://archive.ics.uci.edu/ml/datasets/Mushroom

Appendix

1. Prior Conclusion 1: No simple rule for determining the edibility of a mushroom (attributed in data documentation to the conclusions of the Audubon field guide).

2. Prior Conclusion 2: Four specific rules are deducible from prior analysis of the data (described in documentation):

H2_1:	Rule: odor=NOT(almond.OR.anise.OR.none) Prior Result: 120 poisonous cases missed, 98.52% accuracy
H2_2:	Rule: spore-print-color=green Prior Result: 48 cases missed, 99.41% accuracy
H2_3:	Rule: odor=none.AND.stalk-surface-below-ring=scaly.AND. (stalk-color-above-ring=NOT.brown) Prior Result: 8 cases missed, 99.90% accuracy
H2_4:	Rule: habitat=leaves.AND.cap-color=white Prior Result: 100% accuracy