Mushroom Data Classification
CS5950: Machine Learning
Final Project
By Chris Carlson and Ben Mechling

MUSHROOM DATA

The mushroom data was obtained from a UCI data repository (https://archive.ics.uci.edu/ml/datasets/Mushroom). The data set has 8124 rows and a total of 23 columns. All of the data is categorical. Each value is a single letter representing a value specific to a given attribute (column). The first column "class" indicates whether the mushroom is considered to be poisonous (p) or edible (e). The number of possible values for each attribute varies. The table below shows the possible of each attribute and symbol used to represent it. The largest number of possible values is 12.

Attribute	**Values**
class	edible=e, poisonous=p
cap.shape	bell=b, conical=c, convex=x, flat=f, knobbed=k, sunken=s
cap.surface	fibrous=f, grooves=g, scaly=y, smooth=s
cap.color	brown=n,buff=b,cinnamon=c,gray=g,green=r, pink=p, purple=u, red=e, white=w, yellow=y
bruises	bruises=t, no=f
odor	almond=a, anise=l, creosote=c, fishy=y, foul=f, musty=m, none=n, pungent=p, spicy=s
gill.attachment	attached=a, descending=d, free=f, notched=n
gill.spacing	close=c, crowded=w, distant=d
gill.size	broad=b, narrow=n
gill.color	black=k, brown=n, buff=b, chocolate=h, gray=g, green=r, orange=o, pink=p, purple=u, red=e, white=w, yellow=y
stalk.shape	enlarging=e, tapering=t
stalk.root	bulbous=b, club=c, cup=u, equal=e, rhizomorphs=z, rooted=r, missing=?
stalk.surface.above.ring	fibrous=f, scaly=y, silky=k, smooth=s
stalk.surface.below.ring	fibrous=f, scaly=y, silky=k, smooth=s
stalk.color.above.ring	brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y
stalk.color.below.ring	brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y
veil.type	partial=p, universal=u
veil.color	brown=n, orange=o, white=w, yellow=y
ring.number	none=n, one=o, two=t
ring.type	cobwebby=c, evanescent=e, flaring=f, large=l, none=n, pendant=p, sheathing=s, zone=z
spore.print.color	black=k, brown=n, buff=b, chocolate=h, green=r, orange=o, purple=u, white=w, yellow=y
population	abundant=a,clustered=c,numerous=n,scattered=s,several=v,solitary=y
habitat	grasses=g, leaves=l, meadows=m, paths=p, urban=u, waste=w, woods=d

Data Observations
The attributes values for mushrooms tend to be heavily grouped by the class, making classification somewhat easy.

Class
The number of elements in each class (edible, poisonous) is roughly equal.

Edible	Poisonous
4208	3916

Odor
The odor class is the most distinctive attribute. The table below shows the frequency of each value in the feature for each class over the entire data set. Not all the features were this segregated, but some clear division were found in other features as well.

Class	a	c	f	l	m	n (none)	p	s	y
Edible	400	0	0	400	0	3408	0	0	0
Poisonous	0	192	2160	0	36	120	256	576	576

Spore Print Color
The spore print color is also a strong classifier. Although a comparison of all of its values is not as segregated as odor, if only records where odor = n, were evaluated, then approximately 600 records remained that were ambiguous.

Veil Type
The veil type attribute is supposed to have two possible values, but in the data only one value exists. All 8124 rows have value "p" (partial). For Bayesian and regression models this column was excluded from the data.

CLASSIFICATION

Directly Available Methods
Since the data is entirely categorical, numerical methods cannot be directly applied. Methods that can be directly applied are **naive Bayesian classification** (NBC), **classification trees**, and **random forests**.

Indirect Methods
While implementing a Bayesian classifier, it was observed that the naive Bayesian approach computes a numeric probability estimate value for each element of a test vector. If a dataset was created using these estimates, then numeric methods could be applied as well. Using an estimate of the log-likelihood that the vector is edible given the value of the feature, numeric training and test sets were built. **Linear discriminate analysis** (LDA), **quadratic discriminate analysis** (QDA), and **logistic regression** (Log.Reg.) were applied.

Naive Bayesian Classification
By Ben Mechling

The naïve Bayesian classifier computes an estimate probability that a test vector belongs to a class for each class. In the mushroom data there are only two classes: edible and poisonous. The estimate probabilities are computed using a modified form of Bayes theorem. Bayes' theorem is stated as products.

$$p(c|v) = p(v|c) * p(c) / p(v)$$

As is commonly done for classification, the form is converted to sums of logs and the denominator p(v) is discarded.

$$\log(p(c|v)) = \log(p(c)) + \mathrm{sum}(\log(p(v_i|c)))$$

To find probabilities in a formal sense, the number of occurrences is divided by the total number of elements.

$$p(x) = \mathrm{freq}(x)/n$$
$$\log(p(x)) = \log(\mathrm{freq}(x)) - \log(n)$$

For this application, however, the size of the set only scales the results. Experimentally, probability estimates were better by ignoring the -log(n) term in computations.

$$\mathrm{estimate}(c|v) = \log(\mathrm{freq}(c)) + \mathrm{sum}(\log(\mathrm{freq}(v_i|c)))$$

Thus, the estimate value use in the implementation uses log frequencies rather than proper probabilities.

The freq(v_i|c) value of is the number of occurrences of the given feature value in a given class. A heuristic was used to replace log(freq(v_i|c)) the value never occurred in class c. Naturally the arithmetic results in negative infinity, which is difficult to use in computation. In such cases, the -log(freq(v_i)), the negative log frequency over all classes used as a more reasonable penalty. With this approach if the edible class never have value v_i, but the number of occurrences was large and all poisonous, a large penalty would be given to the estimate probability of the test being edible. On the other hand, if the number of occurrences was small and all were poisonous, then a smaller penalty was given.

*For the mushroom data, features values had a tendency exclusively used by one class. For Bayesian classification, the experimental results improved when the penalty was large. Some of the best results were found for a penalty of -log(freq(v_i)) * 100. The author theorizes that these large penalties push that results strongly to one side or the other and that because the data is very segregated this strong push tends to be accurate. In the general scope of this assignment, the * 100 multiplier was not used.*

Indirect Numerical Classification

By Ben Mechling

The computation process for NBC computes sum(log(freq(v_i|c))) for each class. Effectively the NBC model has two training data sets: one set with edible mushrooms and one set with poisonous mushrooms. In order to run numerical methods, a single numerical training data needed to be created. Thus for a given value v_i a single and useful number was needed. In experimentation, the best form found was a subtraction to the two values.

$$\mathrm{numericalData}(v_i) = \mathrm{freq}(v_i|\mathrm{edible}) - \mathrm{freq}(v_i|\mathrm{poisonous})$$

This form seemed theoretically reasonable given the form for the odds' of a value.

$$\mathrm{odds}(x) = p(x)/(1-p(x))$$

$$\mathrm{logit}(x) = \log(\mathrm{odds}) = \log(p(x)) - \log(1-p(x))$$

For this implementation, proper probabilities were not used, thus 1-p(x) is not a good estimate. The estimate of p(poisonous) should reasonable inverse for p(edible).

estimate logit(v_i) = log(edible) – log(poisonous)

For practical use, the value will be positive if the data supports edible and negative if the data supports poisonous.

For each training set of original data, a version of the mushroom data was created as numerical data. From this numerical data, train and test data sets were partitioned in the same manner as the original data. Using the standard R modeling functions on the training data, LDA, QDA, and logistic regression models were built and predictions were made over the numerical test data.

Bayesian Based Classification Results
Using 5-fold cross validation on NBC, LDA, QDA, and logistic regression gave the following results.

Regular Penalty	**Actually Edible**	**Actually Poisonous**	**Heavy Penalty**	**Actually Edible**	**Actually Poisonous**
Predicted Edible	4206	78	Predicted Edible	4206	3
Predicted Poisonous	0	3838	Predicted Poisonous	2	3913

LDA Classifier

Regular Penalty	**Actually Edible**	**Actually Poisonous**	**Heavy Penalty**	**Actually Edible**	**Actually Poisonous**
Predicted Edible	4193	62	Predicted Edible	4208	40
Predicted Poisonous	15	3854	Predicted Poisonous	0	3876

QDA Classifier

Regular Penalty	**Actually Edible**	**Actually Poisonous**	**Heavy Penalty**	**Actually Edible**	**Actually Poisonous**
Predicted Edible	4208	0	Predicted Edible	4208	22
Predicted Poisonous	0	3916	Predicted Poisonous	0	3894

Logistic Regression Classifier

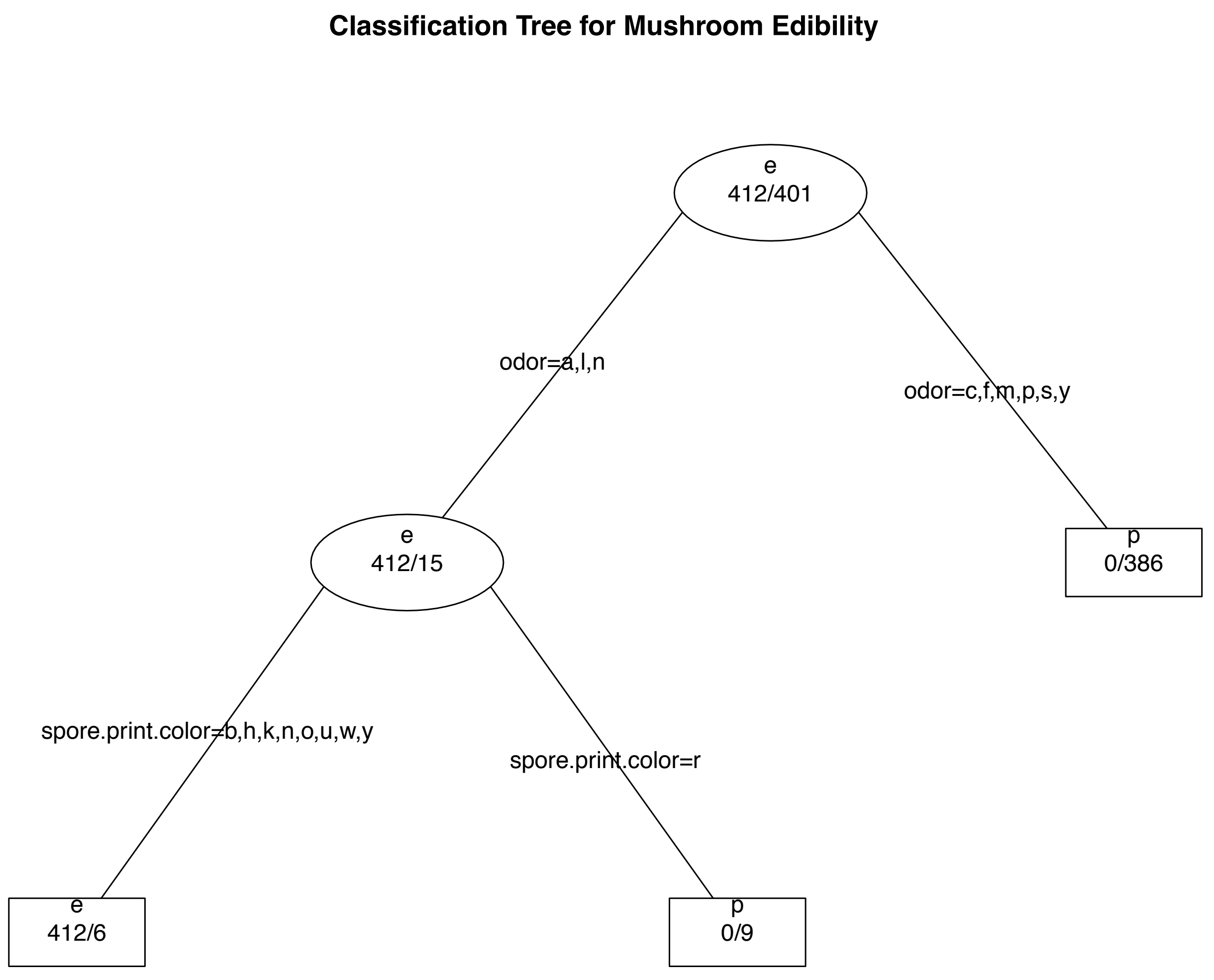
Regular Penalty	**Actually Edible**	**Actually Poisonous**	**Heavy Penalty**	**Actually Edible**	**Actually Poisonous**
Predicted Edible	4208	0	Predicted Edible	4208	0
Predicted Poisonous	0	3916	Predicted Poisonous	0	3916

A number of minor modifications to the algorithm were attempted. Overall, NBC (without the heavy artificial weights) was the worst. LDA and QDA were generally better the NBC, but one of the two was not always better than the other. Logistic regression was by far the best of the classifiers. on the quantitative data.

Decision Tree Classification

By Chris Carlson

The Decision tree classifier was constructed in R using the 'rpart' package. Due to the nature of the data the decision tree classifier was a simple and effective classifier to build. As mentioned in the introduction, the data can be split using only two feature vectors, odor and spore-print-color, with excellent results. The sample plot shown below is representative of the trees rpart builds. In fact, without imposing additional constraints, the rpart package always splits this data on these two features regardless of how the data subsets are arranged.



Random Forest Classification
By Chris Carlson

The Random Forest classifier was produced using the r package called ‘randomForest’. The results of random forest classification were much better than the simple decision tree, which is expected. Typical test error rates for random forest models were less than 0.01, which is the second best result we achieved - the best being logistic regression on Bayesian probabilities. The random forest classifier was tested with as many as 500 trees and as few as 150 trees, and the number of variables available per split was also adjusted from between 3 up to 8. Overall adjusting these characteristics didn’t affect test error rates significantly. Shown below are confusion matrices produced by the random forest algorithm in a typical k-folds cross validation.

```
Confusion Matrix averages from each iteration:
   e      p
e 846.75 0.75
p 0     777.25


   e    p
e 836.5 1
p 0.25 787


   e     p
e 839.25 1
p 0.25 784.25


    e    p
e 844.5 1.25
p 0     779


    e     p
e 840.25 0.75
p 0.25  783.75



Average Confusion Matrix from 5-folds cross validation:
   e      p
e 841.45 0.95
p 0.15  782.25
```

CROSS VALIDATION

K-Folds Cross Validation

K-Folds Cross Validation was used for all model types, though the specific implementation differs between the tree based approaches and the Bayesian approaches. For the Bayesian approaches cross validation was computed precisely according to the method description; the data is dived into k equal size subsets, and for k iterations one section of the data is held out as testing data, and the remaining data is used to build a model which is subsequently tested on the test data. The results of all iterations are reported individually and an average across iterations is reported. For the decision tree methods cross validation was implemented slightly differently; in this approach a model was generated for each of the training folds individually, and each of these models was tested on the test data. The results of each model were averaged to determine the overall test error for the iteration.