CPS 844 Project Report

Members:

- Michael Widianto (501033366)
- Chris Fontein (500189282)

Introduction

The dataset is a description of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota family taken from the UCI Machine Learning Repository [1]. The dataset consists of 8124 instances (2480 instances contain missing values), along with 23 categorical attributes. Furthermore, the class distribution is quite balanced, 51.8% of the samples belong to the edible class, while 48.2% belong to the poisonous class. The purpose of this dataset is to identify whether the mushroom is edible or poisonous based on its attribute values.

Task 1 - Preprocessing Data

To ensure that the dataset can be used by the classifiers and return proper results, it needs to go through several pre-processing steps before any further use.

Firstly, the “?” values in some instances indicate a missing value. These instances are removed from the dataset, resulting in 5644 instances. As a result, the dataset becomes moderately balanced with around 61% of edible class and 39% of poisonous class. Column names were also added as they are not found in the dataset. Additionally, all of the attribute values are acronyms of the actual value, each of the values are mapped to their corresponding value. For instance, one of the values for the attribute cap-shape is “b”, which corresponds to “bell”.

Several attributes have duplicate values such as color names. For example, the attribute veil-color and cap-color both contain “brown” value. To differentiate common values, a prefix is added on all values to make them unique. This step is important to prevent any duplicate attribute names for the dummy variables in the following step.

Finally, as some classifiers implementations don’t support categorical values, they need to be converted to dummy values which only takes the value 0 or 1 to indicate the presence or absence of some attribute. For instance, the target attribute’s value is edible and poisonous, it will be converted to 2 attributes, edible_yes and edible_no (poisonous). However, depending on the model, the target attribute will be exempted from the conversion.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
0	p	x	s	n	t	p	f	c	n	k	e	e	s	s	w	w	p	w	o	p	k	s	u
1	e	x	s	y	t	a	f	c	b	k	e	c	s	s	w	w	p	w	o	p	n	n	g
2	e	b	s	w	t	l	f	c	b	n	e	c	s	s	w	w	p	w	o	p	n	n	m
3	p	x	y	w	t	p	f	c	n	n	e	e	s	s	w	w	p	w	o	p	k	s	u
4	e	x	s	g	f	n	f	w	b	k	t	e	s	s	w	w	p	w	o	e	n	a	g
...	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..
8119	e	k	s	n	f	n	a	c	b	y	e	?	s	s	o	o	p	o	o	p	b	c	l
8120	e	x	s	n	f	n	a	c	b	y	e	?	s	s	o	o	p	n	o	p	b	v	l
8121	e	f	s	n	f	n	a	c	b	n	e	?	s	s	o	o	p	o	o	p	b	c	l
8122	p	k	y	n	f	y	f	c	n	b	t	?	s	k	w	w	p	w	o	e	w	v	l
8123	e	x	s	n	f	n	a	c	b	y	e	?	s	s	o	o	p	o	o	p	o	c	l

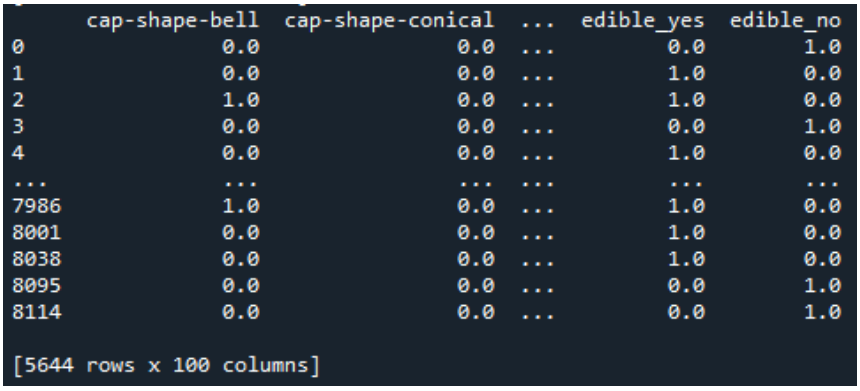
[8124 rows x 23 columns]

Figure 1 - Dataset before pre-processing

	cap-shape	cap-surface	cap-color	...	population	habitat	class
0	convex	smooth	brown	...	scattered	urban	poisonous
1	convex	smooth	yellow	...	numerous	grasses	edible
2	bell	smooth	white	...	numerous	meadows	edible
3	convex	scaly	white	...	scattered	urban	poisonous
4	convex	smooth	gray	...	abundant	grasses	edible
...	...	...	...	...	...	...	...
7986	bell	scaly	brown	...	solitary	paths	edible
8001	convex	scaly	brown	...	solitary	paths	edible
8038	convex	scaly	gray	...	solitary	paths	edible
8095	convex	scaly	cinnamon	...	clustered	woods	poisonous
8114	flat	scaly	cinnamon	...	clustered	woods	poisonous

[5644 rows x 23 columns]

Figure 2 - Dataset during pre-processing



	cap-shape-bell	cap-shape-conical	...	edible_yes	edible_no
0	0.0	0.0	...	0.0	1.0
1	0.0	0.0	...	1.0	0.0
2	1.0	0.0	...	1.0	0.0
3	0.0	0.0	...	0.0	1.0
4	0.0	0.0	...	1.0	0.0
...	...	...	...	...	...
7986	1.0	0.0	...	1.0	0.0
8001	0.0	0.0	...	1.0	0.0
8038	0.0	0.0	...	1.0	0.0
8095	0.0	0.0	...	0.0	1.0
8114	0.0	0.0	...	0.0	1.0

[5644 rows x 100 columns]

Figure 3 - Dataset after pre-processing

Task 2 - Classification Algorithm

Classification is the data mining task chosen for this project. Six different algorithms are used to test and compare the accuracy on the mushroom dataset. The comparison includes models from the base classifier group such as Naive Bayesian, Decision Tree, KNeighbours, and Support Vector Machine (SVM). Additionally, we will also study the performance of ensemble classifier models such as Random Forest and Gradient Boosting. On top of that, each model undergoes stratified k fold cross validation where k = 10 to check its performance under unseen instances.

Cross-validation is applied to each model to ensure that it will still be reliable when tested with unseen data [2]. The cross validation method used is StratifiedKFold cross validation with k=10. The results show that the cross validation score for base and ensemble models remains close to the test accuracy, indicating that the model is reliable and will perform well on unseen data.

The performance of the models can be seen below:

Models	Test Accuracy	AUC Score	Cross-validation Score
Decision Tree (criterion=gini)	97.34%	0.99	93.84%
Bernoulli Naïve Bayesian	92.78%	1	92.00%
K Neighbors (k=5)	100%	1	98.71%
SVC (kernel=linear)	100%	1	97.52%
Random Forest (n=100)	94.77%	1	91.23%
Gradient Boosting (n=100)	100%	1	97.52%

The Decision Tree model uses GINI as the splitting criterion, it also uses the “best” split strategy and a maximum tree depth of 3. These params resulted in a 97.34% accuracy with 0.99 AUC score. Increasing the maximum tree depth will improve the accuracy, but it will make the model more complex [2]. Additionally, using entropy as a splitting criterion is not favourable as it decreases the accuracy to 95.70%.

There are many algorithms to select for the Naive Bayesian model such as Gaussian, Multinomial, Complement, Bernoulli, Categorical, and each option has their own assumptions on the data type. Since the dataset is transformed from categorical to binary values in the pre-processing step, Bernoulli Naive Bayesian model is selected as it assumes that the data is assumed to be binary-valued type [3].

KNeighbors uses 5 neighbours and brute force search algorithm, allowing the model to have an accuracy of 100% and 1 AUC score. Increasing the number of neighbors reduces the accuracy insignificantly, for instance, using 10 neighbors only reduces the accuracy to 99.11%. Other algorithm options are ball tree and kDtree, but according to sklearn KNeighborsClassifier class brute-force search is the most appropriate algorithm for the train data. However, this option

doesn't really affect the accuracy as all algorithms output an accuracy of 100%, it only affects the computation time.

According to the sklearn library, there are 3 options to implement support vector machines such as SVC, NuSVC, and LinearSVC. Compared to LinearSVC, SVC is more practical when the dataset is under tens of thousands instances [4]. Additionally, NuSVC takes longer computation time and outputs lower accuracy than SVC (around 80%), which is why SVC is the preferred choice for this dataset. Other than that, linear kernel is used because it returns the best score result when compared to other kernels such as rbf, poly, and sigmoid using GridSearchCV mod method for model selection.

Among the base classifiers, SVM and K Neighbors show perfect accuracy and AUC score on the mushroom dataset. This means that those models are good at identifying positive instances and minimizing false negatives, as well as avoiding false positives and more accurate at identifying negative instances [5]. Meanwhile, the model with the lowest accuracy compared to other models is the Bernoulli Naive Bayesian. Nonetheless, those values are still above 90%, which means the Naive Bayesian is not a bad model for this dataset.

Random Forest modelling, the first of the ensemble models we examine, creates a number of decision trees and decides on the classification based on the classification with the plurality of their results. This is an attempt to negate the risk of overfitting that might occur in a single tree. For our Random Forest we created 100 trees and provided the same parameters as was provided for our single Decision Tree. The results of this model for accuracy (93.18%), AUC(1), and cross-validation score(91%) are similar to the base decision tree's results (95.7%, 0.99, 91.08%) indicating that our Decision Tree is not overfit. Adjusting the number of trees did not

appear to have a meaningful impact on the results of the forest, but did affect the time to compute.

Number of Trees	Test Accuracy	AUC Score	Cross-validation Score
10	96.19%	1	91.13%
50	94.60%	1	90.92%
100	94.77%	1	91.23%
500	97.25%	1	91.87%
1000	97.17%	1	92.88%
5000	98.32%	1	93.18%

Our other ensemble model, Gradient Boosting, also uses multiple Decision Trees. This time in conjunction with a loss function. Trees are produced to reduce the error of the previous tree based on the loss function. The results of these new trees are then incorporated into the final model. Once again using 100 trees and setting our learning rate (which affects the contribution of each tree to the model) to 0.1, the model produced had results more in line with the non-Decision Tree base classifiers. Of note, if we change the number of trees, the results quickly stabilize indicating that not all the trees were necessary. The number of trees does increase the amount of time to compute results.

Further, adjusting the learning rate can influence the results of our validation. We see raising it too high will drastically reduce its accuracy.The learn rate appears to affect the number of trees required to stabilize. The smaller the learn rate, the more trees are required to get to that point. Before reaching some upper threshold, the learn rate appeared to produce the same end result. As such there was no need to use an incredibly low rate and large number of trees. Number of

trees seemed to be the largest determiner of run time, so tradeoffs would be made between time and accuracy. As such a learn rate of 0.1 and 100 trees were used to build the model. In running the model with different settings, learn rate probably could have increased and the number of trees decreased without affecting the predictive ability of the model, but the settings we used provided a good buffer to handle the data quickly and produce an accurate model.

Learn Rate	Trees	Accuracy	Cross Validation	AUC
0.01	10	62.13%	61.80%	0.99
	50	98.27%	94.49%	1
	100	98.27%	93.98%	1
	500	100%	97.52%	1
0.05	10	98.27%	94.49%	1
	50	100%	98.35%	1
	100	100%	97.52%	1
	500	100%	97.52%	1
0.1	10	98.27%	93.98%	1
	50	100%	97.52%	1
	100	100%	97.52%	1
	500	100%	97.52%	1
0.5	10	100%	99.36%	1
	50	100%	98.30%	1
	100	100%	98.30%	1
	500	100%	98.30%	1
1	10	100%	97.52%	1
	50	100%	97.52%	1
	100	100%	97.52%	1
	500	100%	97.52%	1
5	10	10.27%	70.20%	0.14
	50	10.27%	69.44%	0.14
	100	10.27%	69.44%	0.14
	500	10.27%	69.44%	0.14

Task 3 - Feature Selection and Accuracy Comparison

There are two methods used for feature selection of the mushroom dataset. First is to remove features with low variance, second is to select the best features based on univariate statistical tests. These two methods use the VarianceThreshold and SelectPercentile class respectively from scikit-learn library.

VarianceThreshold removes features with variance that does not meet the given threshold [6]. The minimum threshold is 0 and the maximum is dependent on the dataset, and after some trial and errors, the maximum threshold for the mushroom dataset is 0.249. As a result, the threshold has been set to the middle point, which is 0.125. From this threshold, only 40 out of 99 features are selected, which can be seen in the following list.

['cap-shape-convex', 'cap-shape-flat', 'cap-surface-fibrous', 'cap-surface-scaly', 'cap-surface-smooth', 'cap-color-brown', 'cap-color-gray', 'cap-color-white', 'cap-color-yellow', 'bruises-False', 'bruises-True', 'odor-foul', 'odor-odor-none', 'gill-spacing-close', 'gill-spacing-crowded', 'gill-color-brown', 'gill-color-pink', 'gill-color-white', 'stalk-shape-enlarging', 'stalk-shape-tapering', 'stalk-root-bulbous', 'stalk-root-equal', 'stalk-surface-above-ring-silky', 'stalk-surface-above-ring-smooth', 'stalk-surface-below-ring-silky', 'stalk-surface-below-ring-smooth', 'stalk-color-above-ring-pink', 'stalk-color-above-ring-white', 'stalk-color-below-ring-pink', 'stalk-color-below-ring-white', 'ring-type-large', 'ring-type-pendant', 'spore-print-color-black', 'spore-print-color-brown', 'spore-print-color-chocolate', 'population-scattered', 'population-several', 'population-solitary', 'habitat-grasses', 'habitat-woods']

On the other hand, the SelectPercentile method assigns scores to each feature and selects the top n percent features with the highest score [6]. Similarly, the percentile for the mushroom dataset is set to the middle point, which is 50%. The top 49 features from this method are shown below (order does not indicate highest score).

['cap-color-brown', 'cap-color-buff', 'cap-color-red', 'cap-color-yellow', 'bruises-False', 'bruises-True', 'odor-almond', 'odor-anise', 'odor-creosote', 'odor-foul', 'odor-odor-none', 'odor-pungent', 'gill-spacing-close', 'gill-spacing-crowded', 'gill-size-broad', 'gill-size-narrow', 'gill-color-brown', 'gill-color-chocolate', 'gill-color-gray', 'gill-color-purple', 'stalk-shape-enlarging', 'stalk-shape-tapering', 'stalk-root-bulbous', 'stalk-root-club', 'stalk-root-equal', 'stalk-surface-above-ring-silky', 'stalk-surface-above-ring-smooth', 'stalk-surface-below-ring-silky', 'stalk-surface-below-ring-smooth', 'stalk-color-above-ring-brown', 'stalk-color-above-ring-buff', 'stalk-color-above-ring-gray', 'stalk-color-above-ring-white', 'stalk-color-below-ring-brown', 'stalk-color-below-ring-buff', 'stalk-color-below-ring-gray', 'stalk-color-below-ring-white', 'ring-type-evanescent', 'ring-type-large', 'ring-type-pendant', 'spore-print-color-black', 'spore-print-color-brown', 'spore-print-color-chocolate', 'population-abundant', 'population-numerous', 'population-several', 'habitat-paths', 'habitat-urban', 'habitat-woods']

Surprisingly, the selected features with the specified params improves the accuracy of some models, while also reducing and maintaining the accuracy of other models. The comparison can be seen in the following table:

		Test Accuracy	AUC Score	Cross-validation Score
Decision Tree	No feature selection	97.34%	0.99	93.84%
	feature selection (VarianceThreshold)	93.58%	0.98	90.29%
	feature selection (SelectPercentile)	96.81%	0.99	96.58%
Bernoulli Naïve Bayes	No feature selection	92.78%	1	92.00%
	feature selection (VarianceThreshold)	89.73%	0.96	90.51%
	feature selection (SelectPercentile)	91.98%	1	92.44%
K Neighbors	No feature selection	100%	1	98.76%
	feature selection (VarianceThreshold)	99.60%	1	96.62%

	feature selection (SelectPercentile)	100%	1	98.26%
SVC	No feature selection	100%	1	97.52%
	feature selection (VarianceThreshold)	99.69%	1	97.52%
	feature selection (SelectPercentile)	100%	1	97.66%
Random Forest	No feature selection	94.77%	1	91.23%
	feature selection (VarianceThreshold)	91.32%	1	90.19%
	feature selection (SelectPercentile)	98.32%	1	93.96%
Gradient Boosting	No feature selection	100%	1	97.52%
	feature selection (VarianceThreshold)	99.87	1	97.45%
	feature selection (SelectPercentile)	100%	1	97.66%

For the most part, SelectPercentile maintains the accuracy of K-Neighbors, SVM, AND Gradient Boosting, but it does reduce the accuracy of Decision Tree and Naive Bayes by around 1%. Random Forest is the only model that has benefited from the feature selection as shown by the increase of accuracy. On the other hand, VarianceThreshold only minimally reduces the accuracy of all models.

Similarly, the cross-validation score of each model is either reduced or increased for some models using the attributes from both feature selection methods. Nonetheless, they remain close to the test accuracy, indicating the model will remain stable with unseen data. AUC score of most models remains the same, except for DecisionTree and Naive Bayes that have a reduction when implementing VarianceThreshold.

As some models are not able to maintain their accuracy using the selected features, we need to find the best value of threshold or percentile that will either maintain or minimize the accuracy reduction, and possibly increase the accuracy as well. The table below shows the accuracy of all models on different variations of threshold and percentile.

Models	Metrics	Feature Selection						
		None	VarianceThreshold (threshold)			SelectPercentile (percentile)		
			0.06	0.125	0.18	25%	50%	75%
Decision Tree	Accuracy	97.34%	96.81%	93.58%	93.58%	94.38%	96.81%	97.34%
	AUC Score	0.99	0.98	0.98	0.98	0.98	0.98	0.99
Bernoulli Naive Bayes	Accuracy	92.78	89.73%	89.73%	88.88%	89.55%	91.98%	92.83%
	AUC Score	1	0.99	0.96	0.97	0.99	1	1
K Neighbors	Accuracy	100%	99.96%	99.60%	96.99%	99.65%	98.72%	100%
	AUC Score	1	1	1	0.99	1	1	1
SVC	Accuracy	100%	97.80%	97.52%	95.82%	99.65%	100%	100%
	AUC Score	1	1	1	1	1	1	1
Random Forest	Accuracy	94.77%	**95.48%**	91.32%	92.83%	**94.33%**	**98.32%**	**98.32%**
	AUC Score	1	1	1	0.99	0.99	1	1
Gradient Boosting	Accuracy	100%	100%	99.87%	99.03%	99.65%	100.00%	100%
	AUC Score	1	1	1	1	1	1	1

For VarianceThreshold, most models show a similar pattern where increasing the threshold results in a lower accuracy. Oppositely, reducing the threshold outputs a greater accuracy, but for most models the accuracy will not exceed its initial accuracy (without feature selection). However, this does not apply to the Random Forest model, as the accuracy increases when the threshold is at 0.06 and 0,18, indicating that threshold returns the preferred features for Random Forest to make it more accurate.

SelectPercentile shows a normal pattern where increasing the percentile increases the accuracy as well. Interestly, among all models only SVM and Gradient Boosting are able to maintain their accuracy using only half of the attributes, indicating a superior precision compared to other models. Similarly, Random Forest is able to improve its accuracy to 98.67% when the percentile

is 50%. Based on the accuracy values, features selected when percentile is between 50-75% provides the best accuracy, as the percentile will eventually reduce it to 93.18%.

Conclusion

Based on the performance of all models, Gradient Boosting is the best model for the mushroom dataset due to its capability to maintain the accuracy at 100% and AUC of 1 with fewer features. Meanwhile, Bernoulli Naive Bayes is not the best model when compared to others as shown by the smallest value of accuracy and AUC on different parameters.

References

[1] Mushroom [Dataset]. (1981). UCI Machine Learning Repository. https://doi.org/10.24432/C5959T.

[2] Ding, C. (2024). *lec3_classification_1* [Lecture slides]. Lecture presented in the course CPS844: Data Mining, Toronto Metropolitan University.

[3] scikit-learn developers. (2024). Bernoulli Naive Bayes. scikit-learn. https://scikit-learn.org/stable/modules/naive_bayes.html#bernoulli-naive-bayes

[4] scikit-learn developers. (2024). SVC. scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC

[5] Ding, C. (2024). *lec3_classification_3* [Lecture slides]. Lecture presented in the course CPS844: Data Mining, Toronto Metropolitan University.

[6] scikit-learn developers. (2024). Feature Selection. Scikit-learn. https://scikit-learn.org/stable/modules/feature_selection.html#feature-selection