Mushroom Edibility Classification

Wong Zhao Wu

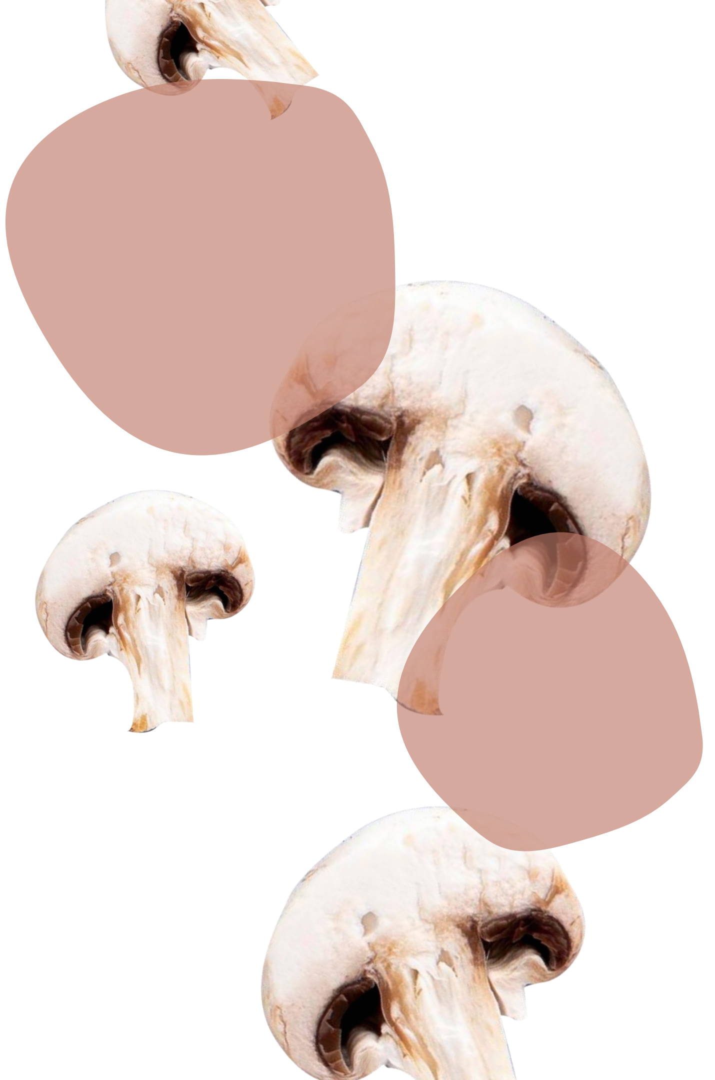


Table of contents

Introduction

- The mushroom dataset includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family (pp. 500-525).
- The task given is a binary classification problem whereby, given the features of mushrooms, we are to classify the mushrooms into **p=Poisonous** or **e=edible**.

Modelling Objective

Build a **Simple** and **Interpretable** Model to Perform **Binary Classification** on the Edibility of Mushroom from *Agarcius and Lepiota Family*.

01

EDA

Understands the dataset and flag out flaws in the dataset.

Data Exploration

Columns	Descriptions	Remarks
is-edible	poisonous=p, edible=e	**Target Variable** with **Balanced Label**
cap-shape	bell=b,conical=c,convex=x,flat=f, knobbed=k,sunken=s	Low Observation Count for Minority Class
cap-surface	fibrous=f,grooves=g,scaly=y,smooth=s	Low Observation Count for Minority Class
cap-color	brown=n,buff=b,cinnamon=c,gray=g,green=r,pink=p,purple=u,red=e,white=w,yellow=y	Low Observation Count for Minority Class
bruises	bruises=t,no=f	**Binary Column** (2 Unique Values)
odor	almond=a,anise=l,creosote=c,fishy=y,foul=f, musty=m,none=n,pungent=p,spicy=s	Low Observation Count for Minority Class
gill-attachment	attached=a,descending=d,free=f,notched=n	**Binary Column** (2 Unique Values)
gill-spacing	close=c,crowded=w,distant=d	**Binary Column** (2 Unique Values)
gill-size	broad=b,narrow=n	**Binary Column** (2 Unique Values)
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	Low Observation Count for Minority Class
stalk-shape	enlarging=e,tapering=t	**Binary Column** (2 Unique Values)
stalk-root	bulbous=b,club=c,cup=u,equal=e, rhizomorphs=z,rooted=r,**missing=?**	**30.5 %** (2480 Rows) of **Missing Values**
stalk-surface-above-ring	fibrous=f,scaly=y,silky=k,smooth=s	Low Observation Count for Minority Class
stalk-surface-below-ring	fibrous=f,scaly=y,silky=k,smooth=s	Low Observation Count for Minority Class
stalk-color-above-ring	brown=n,buff=b,cinnamon=c,gray=g,orange=o,pink=p,red=e,white=w,yellow=y	Low Observation Count for Minority Class
stalk-color-below-ring	brown=n,buff=b,cinnamon=c,gray=g,orange=o,pink=p,red=e,white=w,yellow=y	Low Observation Count for Minority Class
veil-type	partial=p,universal=u	**Constant Value Column** 'veil-type= p'
veil-color	brown=n,orange=o,white=w,yellow=y	Low Observation Count for Minority Class
ring-number	none=n,one=o,two=t	**Discrete Numerical Columns**
ring-type	cobwebby=c,evanescent=e,flaring=f,large=l,none=n,pendant=p,sheathing=s,zone=z	Low Observation Count for Minority Class
spore-print-color	black=k,brown=n,buff=b,chocolate=h,green=r,orange=o,purple=u,white=w,yellow=y	Low Observation Count for Minority Class
population	abundant=a,clustered=c,numerous=n,scattered=s,several=v,solitary=y	Low Observation Count for Minority Class
habitat	grasses=g,leaves=l,meadows=m,paths=p,urban=u,waste=w,woods=d	

EDA Summary

Constant Column

veil-type shall be dropped as it does not bring any information of target variable

Missing Values

Data Cleaning / Imputation is needed to treet missing values for stalk-root

Binary Columns

Perform dummy encoding towards binary columns and One-Hot Encoding to other nominal categorical columns

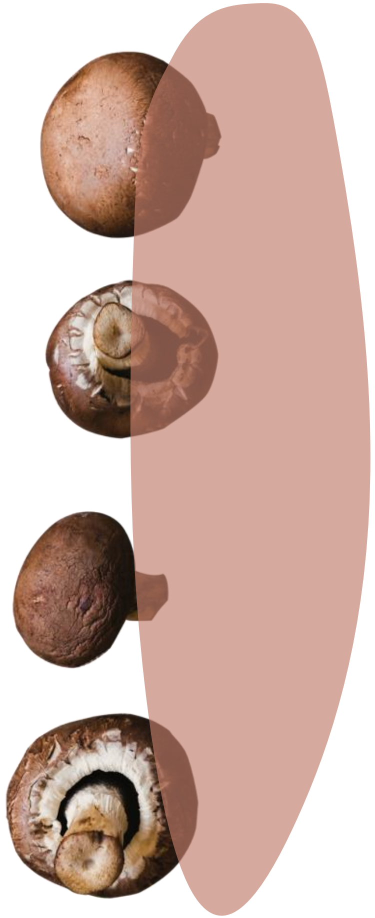
High Cardinality

High cardinality and low observation count in minority classes prompts us to perform feature selection to reduce dimension of dataset.

02

Data Preprocessing

Preprocess dataset into a format that is digestible by model.



Preprocessing Steps

01 Drop Constant Value Column

02 Drop Missing Value Column

03 Feature Selection (Cremer's V Correlation)

04 Feature Encoding (Dummies Encoding + One-Hot Encoding)

05 Recursive Feature Elimination (RFE)

Drop Constant Value Column

veil-type column is dropped as it have **constant value of "p"** which **does not bring any information about the target variable**.

Data Cleaning and Imputation

Since there are around 30.2% of missing values observed for *stalk-root* feature, we can tryout the following approaches:

1. Drop the Entire *stalk-root* Column
2. Impute with Central Tendency(most-frequent = "b")
3. Impute with Advanced Algorithm in SKLearn (e.g. IterativeImputer, KNNImputer)

We will **go with the first approach** since it is the **simplest solution** that **does not change the underlying distribution of dataset.** We can evaluate the decision based on the model's performance later.

Feature Selection

Although we have a total of 107 features after feature encoding, some of the feature might not be useful for modelling as it have too little occurrence, or they are just noises that does not bring any information about the target variable.

For that, feature selection is needed to investigate more on the strong and weak features and how we could perform some feature engineering before we start our modelling.

**All investigation and inference is made with the training set to minimize any data leakage which leads to biased result during model evaluation.*

Cramer's V Correlation Matrix

Cramer's V is a statistical test to calculate correlation in tables which have more than 2x2 rows and columns. It is used as post-test to determine strengths of association after chi-square has determined significance.

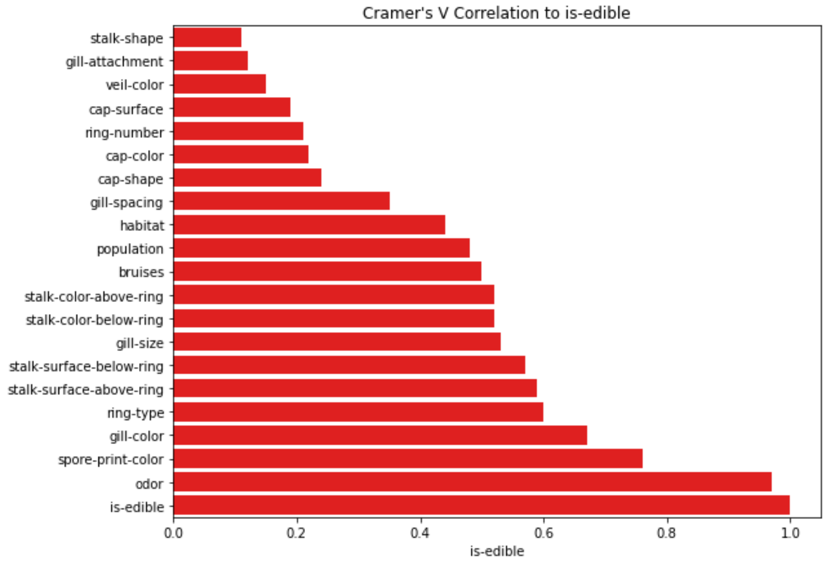
$$V = \sqrt{\frac{\chi^2/n}{k-1}}$$

χ^2 : chi-square

k : number of rows or columns in the contingency table

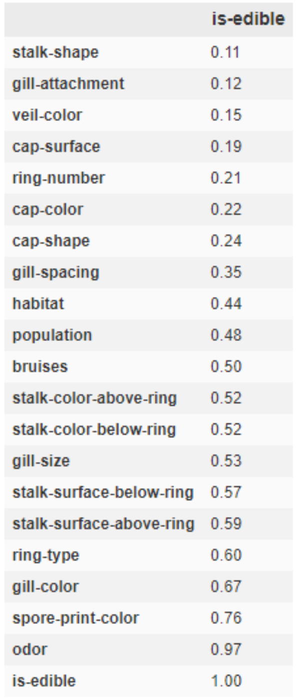
n : Number of observations

$$(Weak Association) 0 < V < 1 (Strong Association)$$



Reference : Cramer's V correlation matrix

Cramer's V Correlation Matrix



	is-edible
stalk-shape	0.11
gill-attachment	0.12
veil-color	0.15
cap-surface	0.19
ring-number	0.21
cap-color	0.22
cap-shape	0.24
gill-spacing	0.35
habitat	0.44
population	0.48
bruises	0.50
stalk-color-above-ring	0.52
stalk-color-below-ring	0.52
gill-size	0.53
stalk-surface-below-ring	0.57
stalk-surface-above-ring	0.59
ring-type	0.60
gill-color	0.67
spore-print-color	0.76
odor	0.97
is-edible	1.00

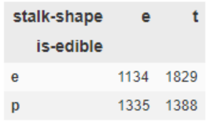
From the Cremer's V statistical test, the following are the observations:

1. *'Odor'* seems to be a strong measure as the association between odor with the target variable, *'is-edible'* is very high.
2. There seems to be little association between *['stalk-shape', 'gill-attachment', 'veil-color', 'cap-surface', 'ring-number', 'cap-color', 'cap-shape']* with target variable, *"is-edible".*

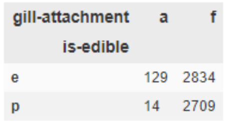
This might be due to the present of **some minority classes**(values with little observation) or there are **simply no association.**

To test out the hypothesis above, we print out the **contingency table** for features with low Cremer's V score and perform some basic visualisation.

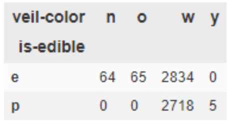
Cramer's V Correlation Matrix



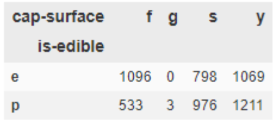
stalk-shape	e	t
is-edible		
e	1134	1829
p	1335	1388



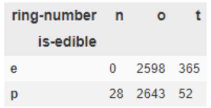
gill-attachment	a	f
is-edible		
e	129	2834
p	14	2709



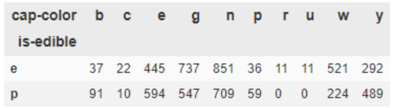
veil-color	n	o	w	y
is-edible				
e	64	65	2834	0
p	0	0	2718	5



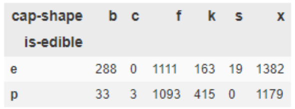
cap-surface	f	g	s	y
is-edible				
e	1096	0	798	1069
p	533	3	976	1211



ring-number	n	o	t
is-edible			
e	0	2598	365
p	28	2643	52



cap-color	b	c	e	g	n	p	r	u	w	y
is-edible										
e	37	22	445	737	851	36	11	11	521	292
p	91	10	594	547	709	59	0	0	224	489



cap-shape	b	c	f	k	s	x
is-edible						
e	288	0	1111	163	19	1382
p	33	3	1093	415	0	1179

By analysing the printed cross-tab the following are the observations:

1. For the features that are flagged as weak association to the target variable, some of the values seems to be a strong split and can make up a strong feature once One-Hot encoded (e.g. gill-attachment, veil-color, ring-number, cap-shape)
2. However, there are also features where all the values are ambiguous in classifying the target variable.(*e.g. stalk-shape, cap-surface*)

Although there might be some hidden relationship when we take account of combination for more than one features, we first attempt to **drop *["stalk-shape", "cap-surface"]* features** from our dataset and revisit the decision after modelling.

Feature Encoding
(Dummies Encoding + One-Hot Encoding)

Dummies Encoding

Encode binary categorical columns to avoid high correlation between encoded features.

One-Hot Encoding

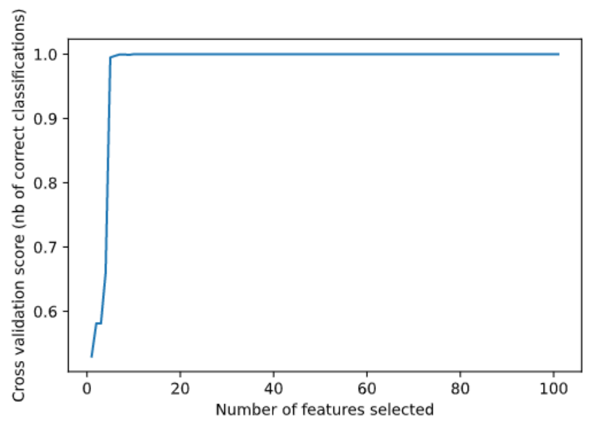
Encode Nominal Categorical Columns into One-Hot features.

Recursive Feature Elimination

As we have 101 features after feature encoding, to get a simpler model, we make use of Recursive Feature Elimination with Support Vector Machine running linear kernel to perform feature selection for us.

By ranking the features through the score generated by LinearSVM, which can find the best linear split with maximum margin, RFE **reduced the number of features recommended into just 10 features**, which we will be using the selected features for modelling later on.

```
Optimal number of features : 10
```



03

Modelling

Train and Evaluate Models based on Modelling Objective.

Modelling

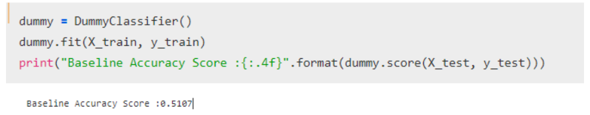
After we have performed feature selection to limit down the number of features from 107 features to just 10 features, we are ready to make use of some Sci-kit Learn model to see whether can we find some **simple yet interpretable model** that can help us classify whether is a mushroom poisonous.

We will first begin by exploring several simple statistical model with default parameters and make decision based on the evaluation outcome and also the interpretability.

Baseline Model & Model Selection

Dummy Classifier

We use Dummy Classifier by predicting the most dominant classes : Poisonous = False and evaluate the score of the baseline predictor as reference point for model selection.

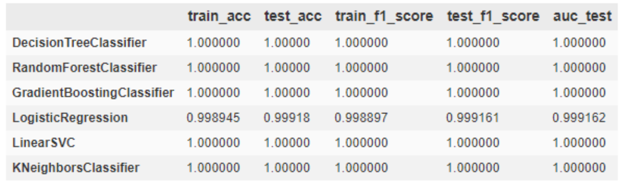


```
dummy = DummyClassifier()
dummy.fit(X_train, y_train)
print("Baseline Accuracy Score :{:.4f}".format(dummy.score(X_test, y_test)))
```

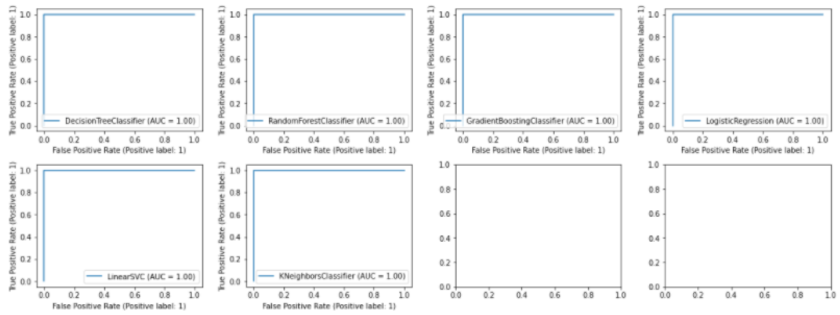
```
Baseline Accuracy Score :0.5107
```

Model Selection

We have built a custom function to train and evaluate the performance of multiple models using accuracy_score, f1_score as well as auc_roc curve as metrics to evaluate the performance.



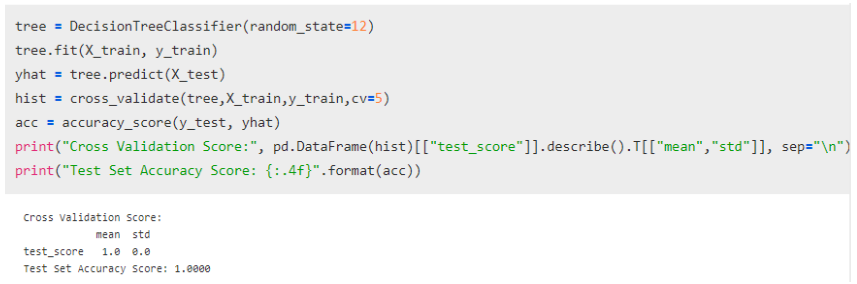
	train_acc	test_acc	train_f1_score	test_f1_score	auc_test
DecisionTreeClassifier	1.000000	1.00000	1.000000	1.000000	1.000000
RandomForestClassifier	1.000000	1.00000	1.000000	1.000000	1.000000
GradientBoostingClassifier	1.000000	1.00000	1.000000	1.000000	1.000000
LogisticRegression	0.998945	0.99918	0.998897	0.999161	0.999162
LinearSVC	1.000000	1.00000	1.000000	1.000000	1.000000
KNeighborsClassifier	1.000000	1.00000	1.000000	1.000000	1.000000



From the accuracy report, we noticed that all the models performs quite well out-of-the-box(without hyperparameter tuning) with accuracy and AUC of 1.0 for most model except Logistic Regression.

As the goal of modelling is to find the **simplest model, with high interpretability**, we continue by looking into **DecisionTreeClassifier** and perform some hyperparameter tuning to see can we get a simpler model that is still able to generalize to the sample data well.

Model Evaluation & Hyperparameter Tuning

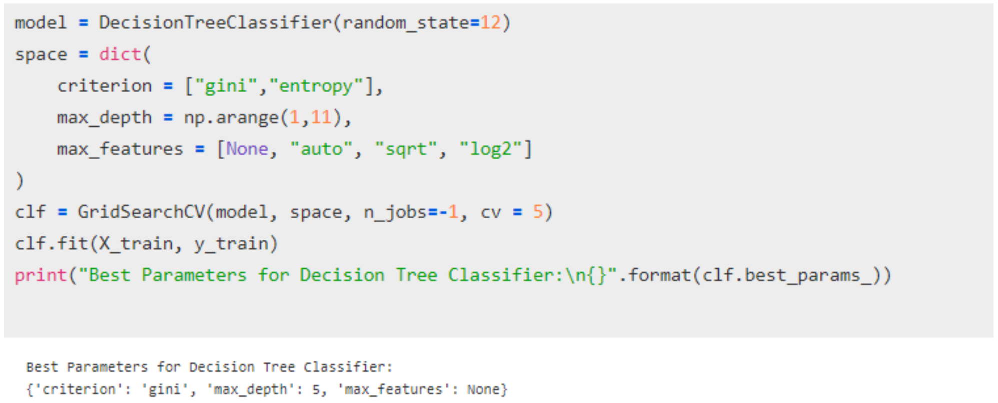


```
tree = DecisionTreeClassifier(random_state=12)
tree.fit(X_train, y_train)
yhat = tree.predict(X_test)
hist = cross_validate(tree,X_train,y_train,cv=5)
acc = accuracy_score(y_test, yhat)
print("Cross Validation Score:", pd.DataFrame(hist)[["test_score"]].describe().T[["mean","std"]], sep="\n")
print("Test Set Accuracy Score: {:.4f}".format(acc))
```

```
Cross Validation Score:
            mean  std
test_score   1.0  0.0
Test Set Accuracy Score: 1.0000
```

With the goal of simplifying the model, we perform hyperparameter tuning towards the max_depth of model to reduce the complexity of model.

From the evaluation metrics, we observed accuracy of 1.0 during K-Fold Cross Validation and also for Hold-Out Test Set, which has not been seen by the algorithm.
Hence, we continue by carrying out hyperparameter tuning to further simplify the model for better interpretability.



```
model = DecisionTreeClassifier(random_state=12)
space = dict(
    criterion = ["gini","entropy"],
    max_depth = np.arange(1,11),
    max_features = [None, "auto", "sqrt", "log2"]
)
clf = GridSearchCV(model, space, n_jobs=-1, cv = 5)
clf.fit(X_train, y_train)
print("Best Parameters for Decision Tree Classifier:\n{}".format(clf.best_params_))
```

```
Best Parameters for Decision Tree Classifier:
{'criterion': 'gini', 'max_depth': 5, 'max_features': None}
```

04

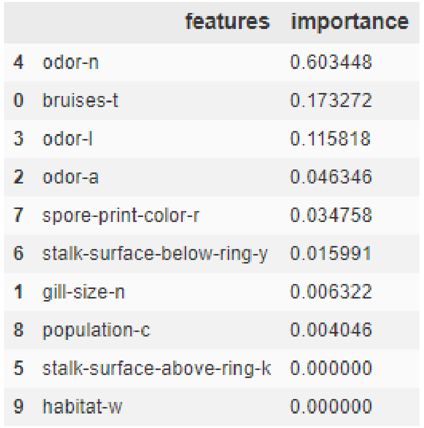
Model Interpretation

Interpret the model and its decision rule.

Feature Importance & Visualizing Decision Tree

By using the best parameters generated from Grid Search Cross-validation, we train our final model and gain some insights by visualising the decision rule of the decision tree.

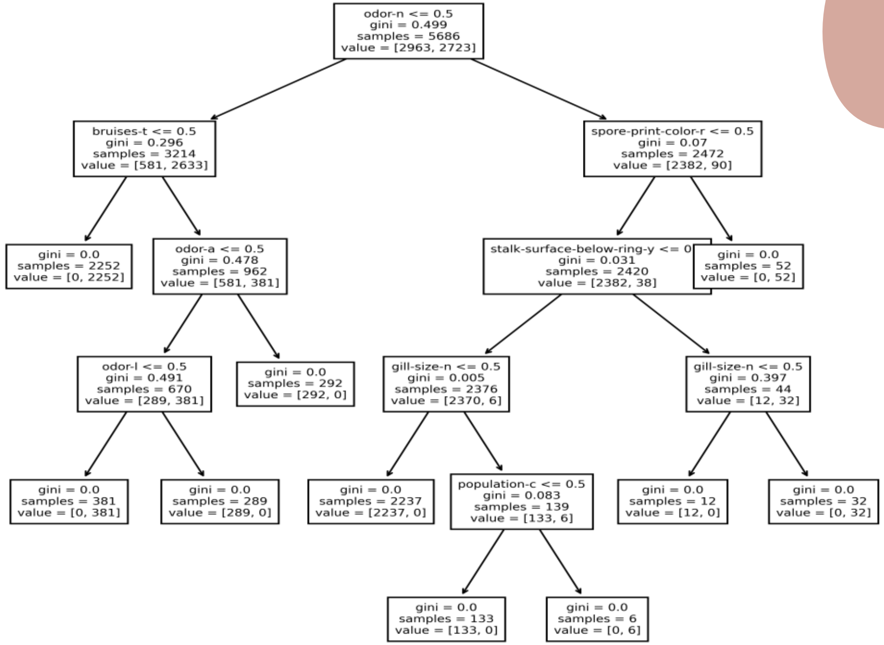
Feature Importance



	features	importance
4	odor-n	0.603448
0	bruises-t	0.173272
3	odor-l	0.115818
2	odor-a	0.046346
7	spore-print-color-r	0.034758
6	stalk-surface-below-ring-y	0.015991
1	gill-size-n	0.006322
8	population-c	0.004046
5	stalk-surface-above-ring-k	0.000000
9	habitat-w	0.000000

From the feature importance we can tell that out of the 10 selected features, only 8 of them are used in the decision rule and odor-n which represents No-odor, is the best-feature with highest feature importance. Meanwhile, "Black Stalk Surface Above Ring, Woods Habitat" is not utilise based on the decision rule.

Visualising Decision Tree


odor-n <= 0.5 gini = 0.499 samples = 5686 value = [2963, 2723]
bruises-t <= 0.5 gini = 0.296 samples = 3214 value = [581, 2633]
spore-print-color-r <= 0.5 gini = 0.07 samples = 2472 value = [2382, 90]
gini = 0.0 samples = 2252 value = [0, 2252]
odor-a <= 0.5 gini = 0.478 samples = 962 value = [581, 381]
stalk-surface-below-ring-y <= 0 gini = 0.031 samples = 2420 value = [2382, 38]
gini = 0.0 samples = 52 value = [0, 52]
odor-l <= 0.5 gini = 0.491 samples = 670 value = [289, 381]
gini = 0.0 samples = 292 value = [292, 0]
gill-size-n <= 0.5 gini = 0.005 samples = 2376 value = [2370, 6]
gill-size-n <= 0.5 gini = 0.397 samples = 44 value = [12, 32]
gini = 0.0 samples = 381 value = [0, 381]
gini = 0.0 samples = 289 value = [289, 0]
gini = 0.0 samples = 2237 value = [2237, 0]
population-c <= 0.5 gini = 0.083 samples = 139 value = [133, 6]
gini = 0.0 samples = 12 value = [12, 0]
gini = 0.0 samples = 32 value = [0, 32]
gini = 0.0 samples = 133 value = [133, 0]
gini = 0.0 samples = 6 value = [0, 6]

Hence, we can conclude that by using only 8 true or false values, we can effectively classify whether is a mushroom poisonous or edible.

Conclusion

From dataset with 22 categorical columns and more than 100 dummies features, we managed to **spot and rectify the errors** (Single-Valued Columns, Null Values, etc) from the dataset and **perform feature selection** to limit down the number of features into just 10 before building the final model which only utilise only 8 features to **achieve perfect classification** based on the evaluation score from cross-validation and hold-out test set.

Thanks!

Personal Learning Reflection

Through the mushroom classification problem, I've gained more hands-on exposure in terms of **Feature Selection** and **Feature Elimination** which is something new to me as in the past, I am more obsessed with chasing the scores at cost of model complexity. Since we can get perfect score with any modern classification models effortlessly with the mushroom dataset, **Model Interpretation** becomes the highlight since a black-box model might not be as useful than a **Simple Decision Map** for an adventurer collecting mushroom in the woods!

Written By : Wong Zhao Wu
Last Modified : 22 May 2021