Mushroom Identifier 3000

Mao Yanyu U2221600B
Ng Jun yu U2222345E
Li Jinglin U2222441E

Table of content

Why Mushrooms?

Personal motivation

Super Mario!

Social impact

1) Yun Nan Mushroom Situation
2) 100 over deaths a year around the world

Commercial Impact

1) Shroomers
2) Small Businesses

Problem Definition:

To investigate the identifiable characteristics of a poisonous mushroom and use ML models to predict whether they are edible.

Exploratory Data Analysis and Visualization

02

Data cleaning -> Visualization -> Analysis

Data Cleaning

1) Through analysis, all the variables are categorical variables with no empty slot

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	ring-number	ring-type	spore-print-color	popu
0	p	x	s	n	t	p	f	c	n	k	...	s	w	w	p	w	o	p	k	
1	e	x	s	y	t	a	f	c	b	k	...	s	w	w	p	w	o	p	n	
2	e	b	s	w	t	l	f	c	b	n	...	s	w	w	p	w	o	p	n	
3	p	x	y	w	t	p	f	c	n	n	...	s	w	w	p	w	o	p	k	
4	e	x	s	g	f	n	f	w	b	k	...	s	w	w	p	w	o	e	n	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
8119	e	k	s	n	f	n	a	c	b	y	...	s	o	o	p	o	o	p	b	
8120	e	x	s	n	f	n	a	c	b	y	...	s	o	o	p	n	o	p	b	
8121	e	f	s	n	f	n	a	c	b	n	...	s	o	o	p	o	o	p	b	
8122	p	k	y	n	f	y	f	c	n	b	...	k	w	w	p	w	o	e	w	
8123	e	x	s	n	f	n	a	c	b	y	...	s	o	o	p	o	o	p	o	

8124 rows × 23 columns

Data Cleaning

2) Using their readme, we converted all the variables to make them readable.

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	ring-number	ring-type
0	p	convex	smooth	brown	have bruises	pungent	free	close	narrow	black	...	smooth	white	white	partial	white	one	pendant
1	e	convex	smooth	yellow	have bruises	almond	free	close	broad	black	...	smooth	white	white	partial	white	one	pendant
2	e	bell	smooth	white	have bruises	anise	free	close	broad	brown	...	smooth	white	white	partial	white	one	pendant
3	p	convex	scaly	white	have bruises	pungent	free	close	narrow	brown	...	smooth	white	white	partial	white	one	pendant
4	e	convex	smooth	grey	no bruises	none	free	crowded	broad	black	...	smooth	white	white	partial	white	one	evanescent
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8119	e	knobbed	smooth	brown	no bruises	none	attached	close	broad	yellow	...	smooth	orange	orange	partial	orange	one	pendant
8120	e	convex	smooth	brown	no bruises	none	attached	close	broad	yellow	...	smooth	orange	orange	partial	brown	one	pendant
8121	e	flat	smooth	brown	no bruises	none	attached	close	broad	brown	...	smooth	orange	orange	partial	orange	one	pendant
8122	p	knobbed	scaly	brown	no bruises	fishy	free	close	narrow	buff	...	silky	white	white	partial	white	one	evanescent
8123	e	convex	smooth	brown	no bruises	none	attached	close	broad	yellow	...	smooth	orange	orange	partial	orange	one	pendant

8124 rows × 23 columns

Data Cleaning

3) Split the data into their classes, Poisonous and Edible

Edible mushrooms

	index	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	...	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	ring-number	ring-type	sp p c
0	1	e	convex	smooth	yellow	have bruises	almond	free	close	broad	...	smooth	white	white	partial	white	one	pendant	br
1	2	e	bell	smooth	white	have bruises	anise	free	close	broad	...	smooth	white	white	partial	white	one	pendant	br
2	4	e	convex	smooth	grey	no bruises	none	free	crowded	broad	...	smooth	white	white	partial	white	one	evanescent	br
3	5	e	convex	scaly	yellow	have bruises	almond	free	close	broad	...	smooth	white	white	partial	white	one	pendant	b
4	6	e	bell	smooth	white	have bruises	almond	free	close	broad	...	smooth	white	white	partial	white	one	pendant	b
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4203	8115	e	convex	smooth	brown	no bruises	none	attached	close	broad	...	smooth	orange	orange	partial	orange	one	pendant	ora
4204	8119	e	knobbed	smooth	brown	no bruises	none	attached	close	broad	...	smooth	orange	orange	partial	orange	one	pendant	
4205	8120	e	convex	smooth	brown	no bruises	none	attached	close	broad	...	smooth	orange	orange	partial	brown	one	pendant	
4206	8121	e	flat	smooth	brown	no bruises	none	attached	close	broad	...	smooth	orange	orange	partial	orange	one	pendant	
4207	8123	e	convex	smooth	brown	no bruises	none	attached	close	broad	...	smooth	orange	orange	partial	orange	one	pendant	ora

4208 rows × 24 columns

Poisonous mushrooms

	index	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	...	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	ring-number	ring-
0	0	p	convex	smooth	brown	have bruises	pungent	free	close	narrow	...	smooth	white	white	partial	white	one	per
1	3	p	convex	scaly	white	have bruises	pungent	free	close	narrow	...	smooth	white	white	partial	white	one	per
2	8	p	convex	scaly	white	have bruises	pungent	free	close	narrow	...	smooth	white	white	partial	white	one	per
3	13	p	convex	scaly	white	have bruises	pungent	free	close	narrow	...	smooth	white	white	partial	white	one	per
4	17	p	convex	smooth	brown	have bruises	pungent	free	close	narrow	...	smooth	white	white	partial	white	one	per
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
3911	8114	p	flat	scaly	cinnamon	no bruises	musty	attached	close	broad	...	scaly	cinnamon	cinnamon	partial	white	null	
3912	8116	p	knobbed	scaly	brown	no bruises	spicy	free	close	narrow	...	silky	pink	white	partial	white	one	evane
3913	8117	p	knobbed	smooth	red	no bruises	fishy	free	close	narrow	...	smooth	pink	white	partial	white	one	evane
3914	8118	p	knobbed	scaly	brown	no bruises	foul	free	close	narrow	...	smooth	pink	white	partial	white	one	evane
3915	8122	p	knobbed	scaly	brown	no bruises	fishy	free	close	narrow	...	silky	white	white	partial	white	one	evane

3916 rows × 24 columns

Analysis - Proportion

1) Calculate the number of poisonous and edible mushrooms for every characteristic

```
#Counting the number of instances for each variable. -filtered into poisonous and non-poisonous mushrooms
p_cap_shape_count = p_mushrooms['cap-shape'].value_counts(dropna=False)
p_cap_color_count = p_mushrooms['cap-color'].value_counts(dropna=False)
p_cap_surface_count = p_mushrooms['cap-surface'].value_counts(dropna=False)
p_bruises_count = p_mushrooms['bruises'].value_counts(dropna=False)
p_odor_count = p_mushrooms['odor'].value_counts(dropna=False)
p_gill_attachment_count = p_mushrooms['gill-attachment'].value_counts(dropna=False)
p_gill_spacing_count = p_mushrooms['gill-spacing'].value_counts(dropna=False)
p_gill_size_count = p_mushrooms['gill-size'].value_counts(dropna=False)
p_gill_color_count = p_mushrooms['gill-color'].value_counts(dropna=False)
p_stalk_shape_count = p_mushrooms['stalk-shape'].value_counts(dropna=False)
p_stalk_root_count = p_mushrooms['stalk-root'].value_counts(dropna=False)
p_stalk_surface_above_ring_count = p_mushrooms['stalk-surface-above-ring'].value_counts(dropna=False)
p_stalk_surface_below_ring_count = p_mushrooms['stalk-surface-below-ring'].value_counts(dropna=False)
p_stalk_color_below_ring_count = p_mushrooms['stalk-color-below-ring'].value_counts(dropna=False)
p_stalk_color_above_ring_count = p_mushrooms['stalk-color-above-ring'].value_counts(dropna=False)
p_veil_type_count = p_mushrooms['veil-type'].value_counts(dropna=False)
p_veil_color_count = p_mushrooms['veil-color'].value_counts(dropna=False)
p_ring_number_count = p_mushrooms['ring-number'].value_counts(dropna=False)
p_ring_type_count = p_mushrooms['ring-type'].value_counts(dropna=False)
p_spore_print_color_count = p_mushrooms['spore-print-color'].value_counts(dropna=False)
p_population_count = p_mushrooms['population'].value_counts(dropna=False)
p_habitat_count = p_mushrooms['habitat'].value_counts(dropna=False)

e_cap_shape_count = e_mushrooms['cap-shape'].value_counts(dropna=False)
e_cap_color_count = e_mushrooms['cap-color'].value_counts(dropna=False)
e_cap_surface_count = e_mushrooms['cap-surface'].value_counts(dropna=False)
e_bruises_count = e_mushrooms['bruises'].value_counts(dropna=False)
e_odor_count = e_mushrooms['odor'].value_counts(dropna=False)
e_gill_attachment_count = e_mushrooms['gill-attachment'].value_counts(dropna=False)
e_gill_spacing_count = e_mushrooms['gill-spacing'].value_counts(dropna=False)
e_gill_size_count = e_mushrooms['gill-size'].value_counts(dropna=False)
e_gill_color_count = e_mushrooms['gill-color'].value_counts(dropna=False)
e_stalk_shape_count = e_mushrooms['stalk-shape'].value_counts(dropna=False)
e_stalk_root_count = e_mushrooms['stalk-root'].value_counts(dropna=False)
e_stalk_surface_above_ring_count = e_mushrooms['stalk-surface-above-ring'].value_counts(dropna=False)
e_stalk_surface_below_ring_count = e_mushrooms['stalk-surface-below-ring'].value_counts(dropna=False)
e_stalk_color_below_ring_count = e_mushrooms['stalk-color-below-ring'].value_counts(dropna=False)
e_stalk_color_above_ring_count = e_mushrooms['stalk-color-above-ring'].value_counts(dropna=False)
e_veil_type_count = e_mushrooms['veil-type'].value_counts(dropna=False)
e_veil_color_count = e_mushrooms['veil-color'].value_counts(dropna=False)
e_ring_number_count = e_mushrooms['ring-number'].value_counts(dropna=False)
e_ring_type_count = e_mushrooms['ring-type'].value_counts(dropna=False)
e_spore_print_color_count = e_mushrooms['spore-print-color'].value_counts(dropna=False)
e_population_count = e_mushrooms['population'].value_counts(dropna=False)
```

Analysis - Proportion

2) Calculate the proportion of poisonous mushrooms to edible mushrooms for every characteristic

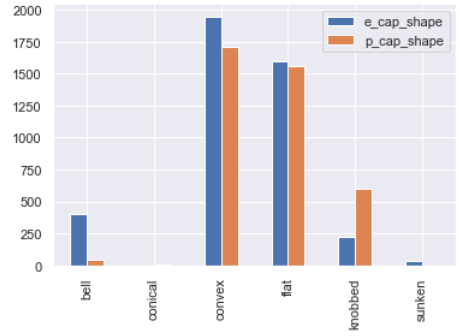
Proportion of Poisonous Mushrooms for every characteristic = $\frac{P_{Mushrooms}}{P_{mushrooms}+E_{Mushrooms}}$

```python
#Defining the 2 functions required for calculating proportion
#calculating the proportion
def calc_proportions(combined_df, characteristic, mushroom_types):
    proportions = {}
    for mushroom_type in mushroom_types:
        proportion = combined_df.loc[mushroom_type, characteristic] / combined_df.loc[mushroom_type].sum()
        proportions[mushroom_type] = proportion
    return proportions
#printing the proportion
def print_proportions(combined_df, characteristic, mushroom_types):
    proportions = calc_proportions(combined_df, characteristic, mushroom_types)
    for mushroom_type, proportion in proportions.items():
        print(f"{mushroom_types[mushroom_type]}: {proportion:.4f}")
```

Analysis - Proportion

3) Use the function to calculate proportion and represent the data on a bar chart

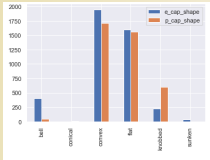
```
#cap-shape
combined = pd.concat([e_cap_shape_count, p_cap_shape_count], axis=1, sort=True).fillna(0)
combined.columns = ['e_cap_shape', 'p_cap_shape']
combined.plot(kind='bar')
plt.show()
#proportion
mushroom_types = {'bell': 'Bell', 'conical': 'Conical', 'convex': 'Convex', 'flat': 'Flat', 'knobbed': 'Knobbed', 'sunken': 'Sur
print_proportions(combined, 'p_cap_shape', mushroom_types)
```



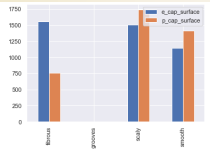

```
Bell: 0.1062
Conical: 1.0000
Convex: 0.4672
Flat: 0.4937
Knobbed: 0.7246
Sunken: 0.0000
```

Analysis - Proportion

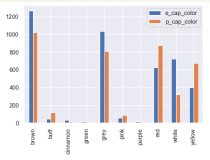
Findings



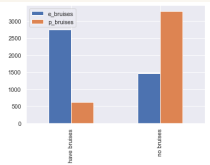
Bell: 0.1862
Conical: 1.0000
Convex: 0.4672
Flat: 0.4937
Knobbed: 0.7246
Sunken: 0.0000



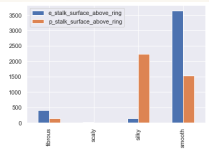
Smooth: 0.5524
Scaly: 0.5364
Fibrous: 0.3276
Grooves: 1.0000



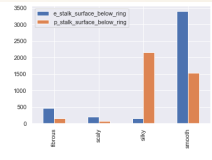
Brown: 0.4466
Yellow: 0.6269
White: 0.3077
Grey: 0.4391
Red: 0.5840
Pink: 0.6111
Buff: 0.7143
Purple: 0.0000
Cinnamon: 0.2727
Green: 0.0000



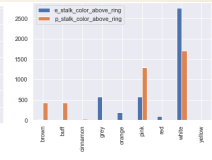
Have bruises: 0.1848
No bruises: 0.6933



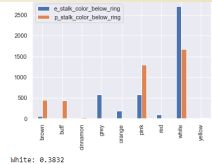
Smooth: 0.2968
Fibrous: 0.2609
Scaly: 0.3333
Silky: 0.9393



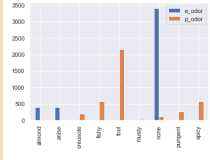
Smooth: 0.3112
Fibrous: 0.2400
Scaly: 0.2676
Silky: 0.9375



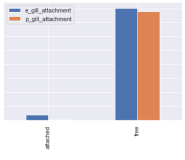
White: 0.3835
Grey: 0.0000
Pink: 0.6923
Brown: 0.0643
Buff: 1.0000
Red: 0.0000
Orange: 0.0000
Cinnamon: 1.0000
Yellow: 1.0000



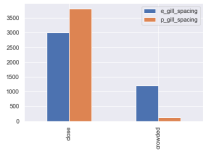
White: 0.3832
Pink: 0.6923
Grey: 0.0000
Buff: 1.0000
Brown: 0.0750
Red: 0.0000
Yellow: 1.0000
Orange: 0.0000
Cinnamon: 1.0000



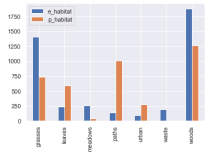
Pungent: 1.0000
Almond: 0.0000
Anise: 0.0000
None: 0.0340
Creosote: 1.0000
Fishy: 1.0000
Spicy: 1.0000
Musty: 1.0000
Foul: 1.0000



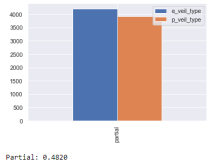
Free: 0.4925
Attached: 0.0857



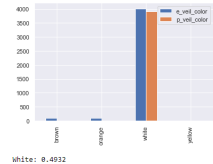
Close: 0.5584
Crowded: 0.0054



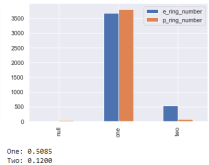
Urban: 0.7391
Grasses: 0.3445
Meadows: 0.1233
Woods: 0.4028
Paths: 0.8811
Waste: 0.0000
Leaves: 0.7115



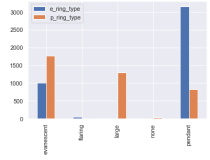
Partial: 0.4820



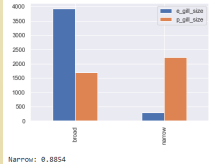
White: 0.4932
Brown: 0.0000
Orange: 0.0000
Yellow: 1.0000



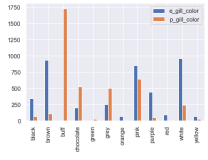
One: 0.5085
Two: 0.1200
Null: 1.0000



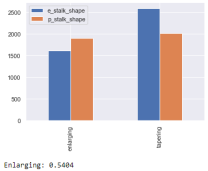
Pendant: 0.2056
Evanescent: 0.6369
Large: 1.0000
Flaring: 0.0000
None: 1.0000



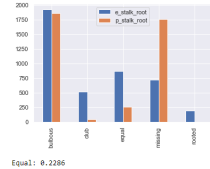
Narrow: 0.8854
Broad: 0.3815



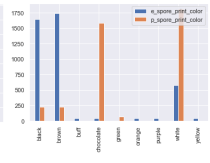
Black: 0.1569
Brown: 0.1069
Grey: 0.6702
Pink: 0.4290
White: 0.2047
Chocolate: 0.7213
Purple: 0.0976
Red: 0.0000
Buff: 1.0000
Green: 1.0000
Yellow: 0.2558
Orange: 0.0000



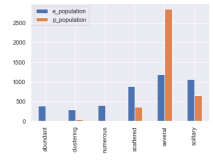
Enlarging: 0.5404
Tapering: 0.4375



Equal: 0.2286
Club: 0.0791
Bulbous: 0.4915
Rooted: 0.0000
Missing: 0.7007



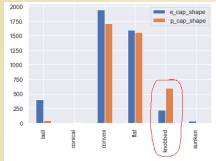
Black: 0.1197
Brown: 0.1138
Purple: 0.0000
Chocolate: 0.9706
White: 0.7588
Green: 1.0000
Orange: 0.0000
Yellow: 0.0000
Buff: 0.0000



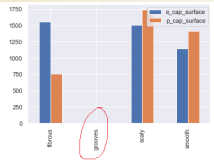
Abundant: 0.0000
Clustering: 0.1529
Numerous: 0.0000
Scattered: 0.2949
Several: 0.7050
Solitary: 0.3785

Analysis - Proportion

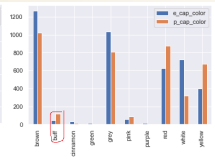
Findings



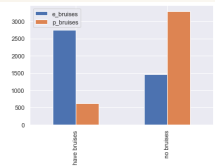
Bell: 0.1062
Conical: 1.0000
Convex: 0.4672
Flat: 0.4937
Knobbed: 0.7246
Sunken: 0.0000



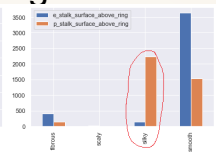
Smooth: 0.5524
Scaly: 0.5364
Fibrous: 0.3276
Grooves: 1.0000



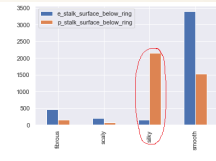
Brown: 0.4466
Yellow: 0.6269
White: 0.3077
Grey: 0.4391
Red: 0.5840
Pink: 0.6111
Buff: 0.7143
Purple: 0.0000
Cinnamon: 0.2727
Green: 0.0000



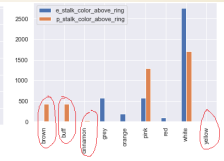
Have bruises: 0.1848
No bruises: 0.6933



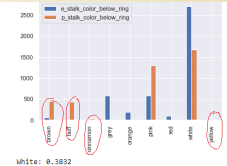
Smooth: 0.2068
Fibrous: 0.2609
Scaly: 0.3333
Silky: 0.9393



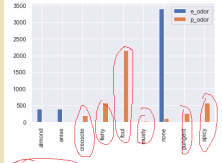
Smooth: 0.3112
Fibrous: 0.2400
Scaly: 0.2676
Silky: 0.9375



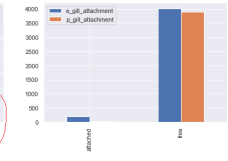
White: 0.3835
Grey: 0.0000
Pink: 0.0923
Brown: 0.0643
Buff: 1.0000
Red: 0.0000
Orange: 0.0000
Cinnamon: 1.0000
Yellow: 1.0000



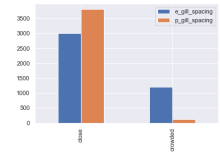
White: 0.3832
Pink: 0.6923
Grey: 0.0000
Buff: 1.0000
Brown: 0.0750
Red: 0.0000
Yellow: 1.0000
Orange: 0.0000
Cinnamon: 1.0000



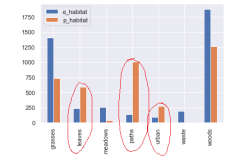
Pungent: 1.0000
Almond: 0.0000
Anise: 0.0000
None: 0.0340
Creosote: 1.0000
Fishy: 1.0000
Spicy: 1.0000
Musty: 1.0000
Foul: 1.0000



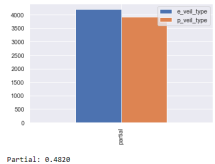
Free: 0.4925
Attached: 0.0857



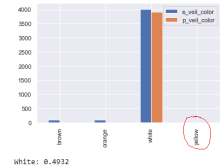
Close: 0.5584
Crowded: 0.8854



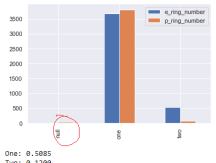
Urban: 0.7391
Grasses: 0.1445
Meadows: 0.1223
Woods: 0.4028
Paths: 0.881
Waste: 0.0000
Leaves: 0.7115



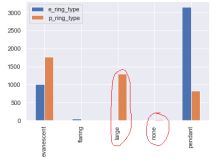
Partial: 0.4820



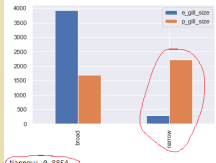
White: 0.4932
Brown: 0.0000
Orange: 0.0000
Yellow: 1.0000



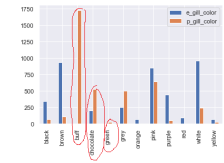
One: 0.5085
Two: 0.1200
Null: 1.0000



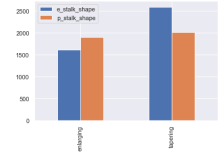
Pendant: 0.2056
Evanescent: 0.6369
Large: 1.0000
Flaring: 0.0000
None: 1.0000



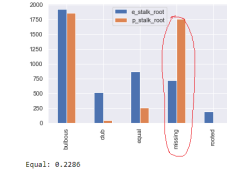
Narrow: 0.8854
Broad: 0.3015



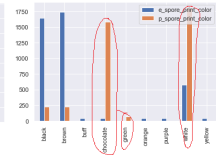
Black: 0.1569
Brown: 0.1069
Grey: 0.6702
Pink: 0.4200
White: 0.2047
Chocolate: 0.7213
Purple: 0.0976
Red: 0.0000
Buff: 1.0000
Green: 1.0000
Yellow: 0.2558
Orange: 0.0000



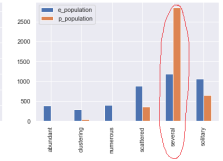
Enlarging: 0.5404
Tapering: 0.4375



Equal: 0.2286
Club: 0.0791
Bulbous: 0.4915
Rooted: 0.0000
Missing: 0.7097



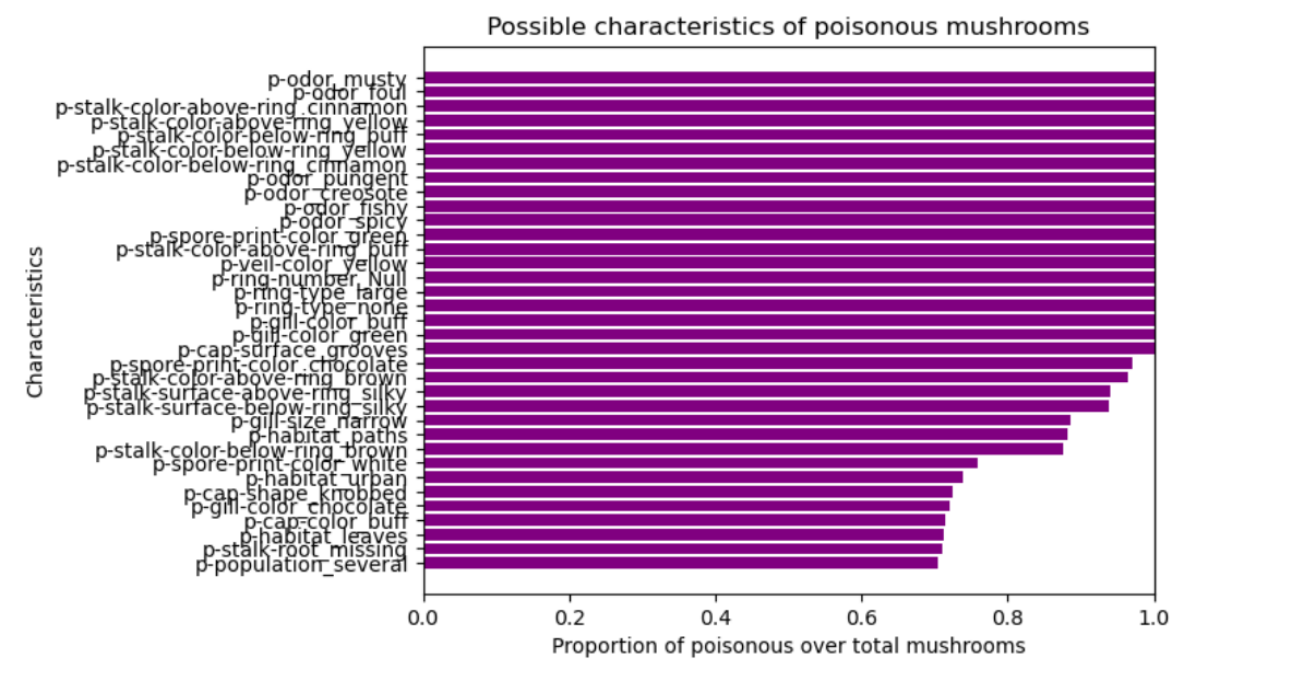
Black: 0.1197
Brown: 0.1138
Purple: 0.0000
Chocolate: 0.9706
White: 0.7588
Green: 1.0000
Orange: 0.0000
Yellow: 0.0000
Buff: 0.0000



Abundant: 0.0000
Clustering: 0.1529
Numerous: 0.0000
Scattered: 0.2949
Several: 0.7050
Solitary: 0.3785

Analysis – Proportion

Findings



KMode

- **Clustering algorithm** that works with **categorical data**.
- Works by **iteratively** assigning data points to clusters based on the **similarity** of their categorical values.
- Updates the cluster centers based on the mode of the categorical values of the data points assigned to that cluster.
- The algorithm **continues iterating** until the cluster assignments **no longer change**.

Analysis - KMode

1) Use One-hot encoding to give a numerical value to the categorical data.

2) Create a Kmode object using the ‘huang’ algorithm and 5 clusters

```
# Select categorical columns to cluster on
cat_cols = dataset.columns
```

```
# Convert categorical variables to numerical values using one-hot encoding
one_hot = pd.get_dummies(dataset[cat_cols])

# Create KModes object
km = KModes(n_clusters=5, init='Huang', n_init=5, verbose=1)

# Fit the data to the model
clusters = km.fit_predict(one_hot)

# Add the clusters to the original dataset
dataset['cluster'] = clusters
```

```
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 1, iteration: 1/100, moves: 2887, cost: 94594.0
Run 1, iteration: 2/100, moves: 163, cost: 94534.0
Run 1, iteration: 3/100, moves: 1, cost: 94534.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
```

Analysis - KMode

3) Apply PCA to reduce the number of dimensions
4) Create a scatter plot of the first 2 principal components.
5) Print the Mode for each attribute for each cluster

```python
# Print the mode (most frequent value) of each attribute for each cluster
for cluster in set(clusters):
    print(f'Cluster {cluster} mode:')
    for column in cat_cols:
        mode = dataset[dataset['cluster'] == cluster][column].mode()[0]
        print(f'\t{column}: {mode}')
```

```python
# Apply PCA to reduce the number of dimensions to 2
pca = PCA(n_components=2)
pca_result = pca.fit_transform(one_hot)

# Define the colors for each cluster
colors = ['red', 'green', 'blue', 'purple', 'orange']

# Create a scatter plot of the first two principal components
for i in range(len(set(clusters))):
    plt.scatter(pca_result[clusters==i, 0], pca_result[clusters==i, 1], c=colors[i], label=f'Cluster {i}')

plt.title('KModes Clustering')
plt.xlabel('PCA 1')
plt.ylabel('PCA 2')
plt.legend()
plt.show()
```

Analysis - KMode

Findings

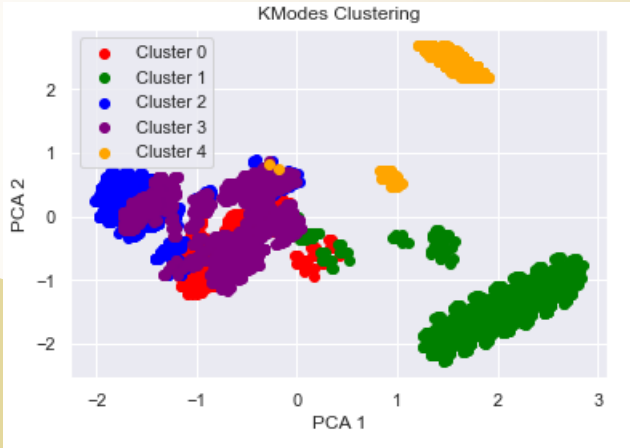
```
Cluster 0 mode:
        class: e
        cap-shape: convex
        cap-surface: fibrous
        cap-color: grey
        bruises: no bruises
        odor: none
        gill-attachment: free
        gill-spacing: crowded
        gill-size: broad
        gill-color: pink
        stalk-shape: tapering
        stalk-root: equal
        stalk-surface-above-ring: smooth
        stalk-surface-below-ring: smooth
        stalk-color-above-ring: white
        stalk-color-below-ring: white
        veil-type: partial
        veil-color: white
        ring-number: one
        ring-type: evanescent
        spore-print-color: black
        population: scattered
        habitat: grasses
```

```
Cluster 1 mode:
        class: p
        cap-shape: flat
        cap-surface: scaly
        cap-color: brown
        bruises: no bruises
        odor: fishy
        gill-attachment: free
        gill-spacing: close
        gill-size: narrow
        gill-color: buff
        stalk-shape: tapering
        stalk-root: missing
        stalk-surface-above-ring: smooth
        stalk-surface-below-ring: smooth
        stalk-color-above-ring: white
        stalk-color-below-ring: white
        veil-type: partial
        veil-color: white
        ring-number: one
        ring-type: evanescent
        spore-print-color: white
        population: several
        habitat: woods
```

```
Cluster 2 mode:
        class: e
        cap-shape: flat
        cap-surface: fibrous
        cap-color: brown
        bruises: have bruises
        odor: none
        gill-attachment: free
        gill-spacing: close
        gill-size: broad
        gill-color: white
        stalk-shape: tapering
        stalk-root: bulbous
        stalk-surface-above-ring: smooth
        stalk-surface-below-ring: smooth
        stalk-color-above-ring: white
        stalk-color-below-ring: white
        veil-type: partial
        veil-color: white
        ring-number: one
        ring-type: pendant
        spore-print-color: brown
        population: several
        habitat: woods
```

```
Cluster 3 mode:
        class: e
        cap-shape: convex
        cap-surface: smooth
        cap-color: white
        bruises: have bruises
        odor: none
        gill-attachment: free
        gill-spacing: close
        gill-size: broad
        gill-color: white
        stalk-shape: enlarging
        stalk-root: bulbous
        stalk-surface-above-ring: smooth
        stalk-surface-below-ring: smooth
        stalk-color-above-ring: white
        stalk-color-below-ring: white
        veil-type: partial
        veil-color: white
        ring-number: one
        ring-type: pendant
        spore-print-color: brown
        population: scattered
        habitat: grasses
```

```
Cluster 4 mode:
        class: p
        cap-shape: convex
        cap-surface: scaly
        cap-color: grey
        bruises: no bruises
        odor: foul
        gill-attachment: free
        gill-spacing: close
        gill-size: broad
        gill-color: chocolate
        stalk-shape: enlarging
        stalk-root: bulbous
        stalk-surface-above-ring: silky
        stalk-surface-below-ring: silky
        stalk-color-above-ring: brown
        stalk-color-below-ring: brown
        veil-type: partial
        veil-color: white
        ring-number: one
        ring-type: large
        spore-print-color: chocolate
        population: solitary
        habitat: woods
```



Analysis - KMode

Findings

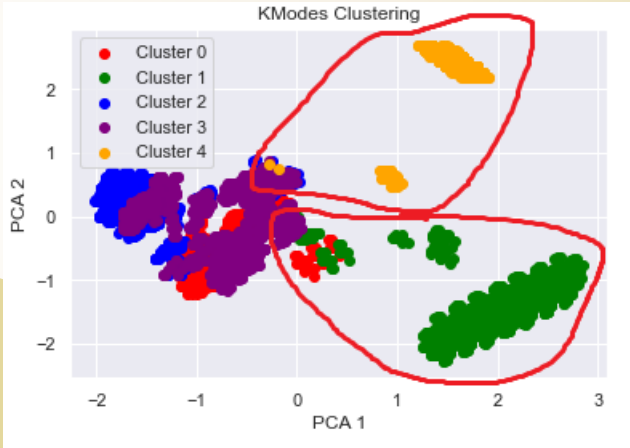
```
Cluster 0 mode:
        class: e
        cap-shape: convex
        cap-surface: fibrous
        cap-color: grey
        bruises: no bruises
        odor: none
        gill-attachment: free
        gill-spacing: crowded
        gill-size: broad
        gill-color: pink
        stalk-shape: tapering
        stalk-root: equal
        stalk-surface-above-ring: smooth
        stalk-surface-below-ring: smooth
        stalk-color-above-ring: white
        stalk-color-below-ring: white
        veil-type: partial
        veil-color: white
        ring-number: one
        ring-type: evanescent
        spore-print-color: black
        population: scattered
        habitat: grasses
```

```
Cluster 1 mode:
        class: p
        cap-shape: flat
        cap-surface: scaly
        cap-color: brown
        bruises: no bruises
        odor: fishy
        gill-attachment: free
        gill-spacing: close
        gill-size: narrow
        gill-color: buff
        stalk-shape: tapering
        stalk-root: missing
        stalk-surface-above-ring: smooth
        stalk-surface-below-ring: smooth
        stalk-color-above-ring: white
        stalk-color-below-ring: white
        veil-type: partial
        veil-color: white
        ring-number: one
        ring-type: evanescent
        spore-print-color: white
        population: several
        habitat: woods
```

```
Cluster 2 mode:
        class: e
        cap-shape: flat
        cap-surface: fibrous
        cap-color: brown
        bruises: have bruises
        odor: none
        gill-attachment: free
        gill-spacing: close
        gill-size: broad
        gill-color: white
        stalk-shape: tapering
        stalk-root: bulbous
        stalk-surface-above-ring: smooth
        stalk-surface-below-ring: smooth
        stalk-color-above-ring: white
        stalk-color-below-ring: white
        veil-type: partial
        veil-color: white
        ring-number: one
        ring-type: pendant
        spore-print-color: brown
        population: several
        habitat: woods
```

```
Cluster 3 mode:
        class: e
        cap-shape: convex
        cap-surface: smooth
        cap-color: white
        bruises: have bruises
        odor: none
        gill-attachment: free
        gill-spacing: close
        gill-size: broad
        gill-color: white
        stalk-shape: enlarging
        stalk-root: bulbous
        stalk-surface-above-ring: smooth
        stalk-surface-below-ring: smooth
        stalk-color-above-ring: white
        stalk-color-below-ring: white
        veil-type: partial
        veil-color: white
        ring-number: one
        ring-type: pendant
        spore-print-color: brown
        population: scattered
        habitat: grasses
```

```
Cluster 4 mode:
        class: p
        cap-shape: convex
        cap-surface: scaly
        cap-color: grey
        bruises: no bruises
        odor: foul
        gill-attachment: free
        gill-spacing: close
        gill-size: broad
        gill-color: chocolate
        stalk-shape: enlarging
        stalk-root: bulbous
        stalk-surface-above-ring: silky
        stalk-surface-below-ring: silky
        stalk-color-above-ring: brown
        stalk-color-below-ring: brown
        veil-type: partial
        veil-color: white
        ring-number: one
        ring-type: large
        spore-print-color: chocolate
        population: solitary
        habitat: woods
```



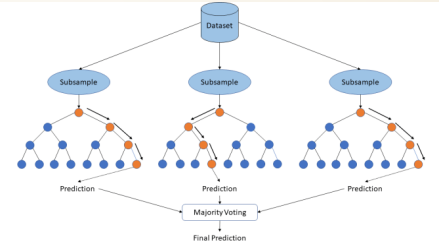
Machine Learning

03

Models Used

Decision Tree

Simple and Interpretable model



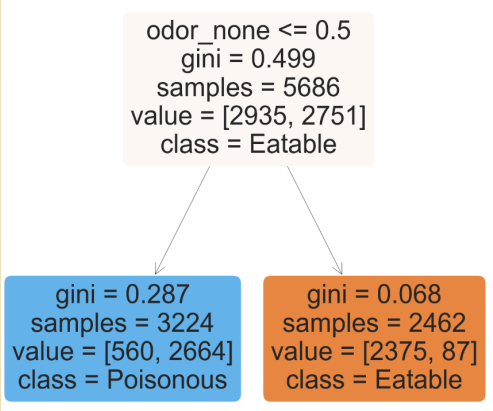
Random Forest

Complex but Accurate model

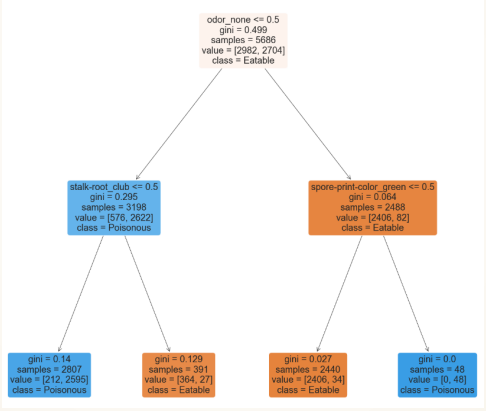
Decision Tree

Decision Tree

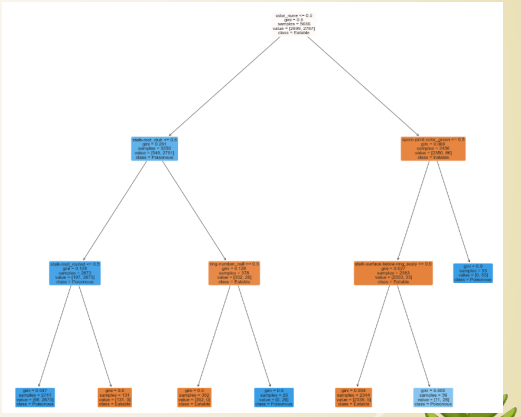
Depth 1



Depth 2

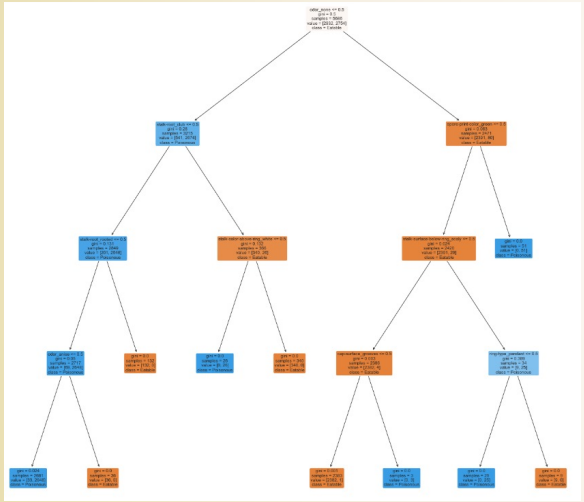


Depth 3

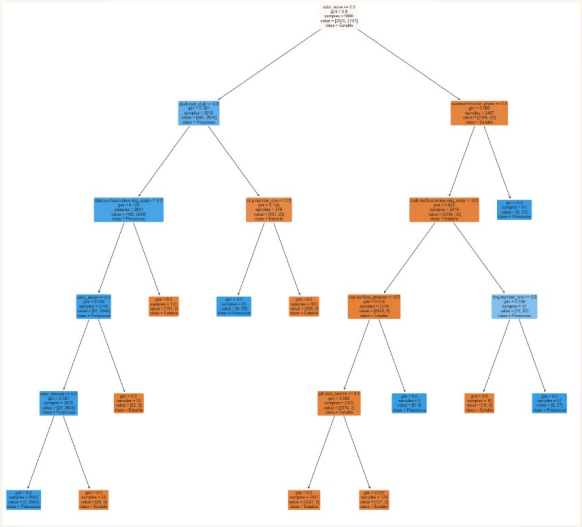


Decision Tree

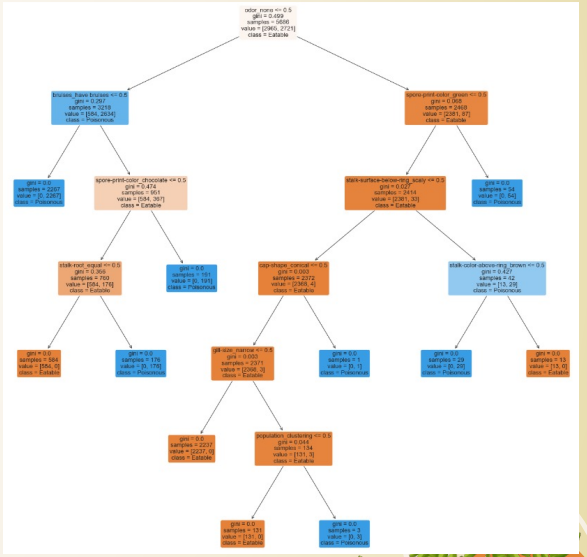
Depth 4



Depth 5

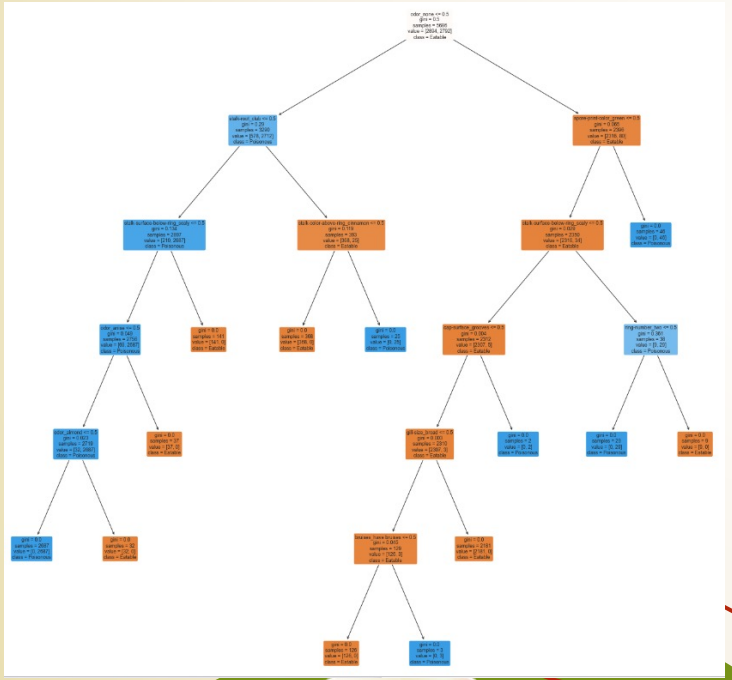


Depth 6

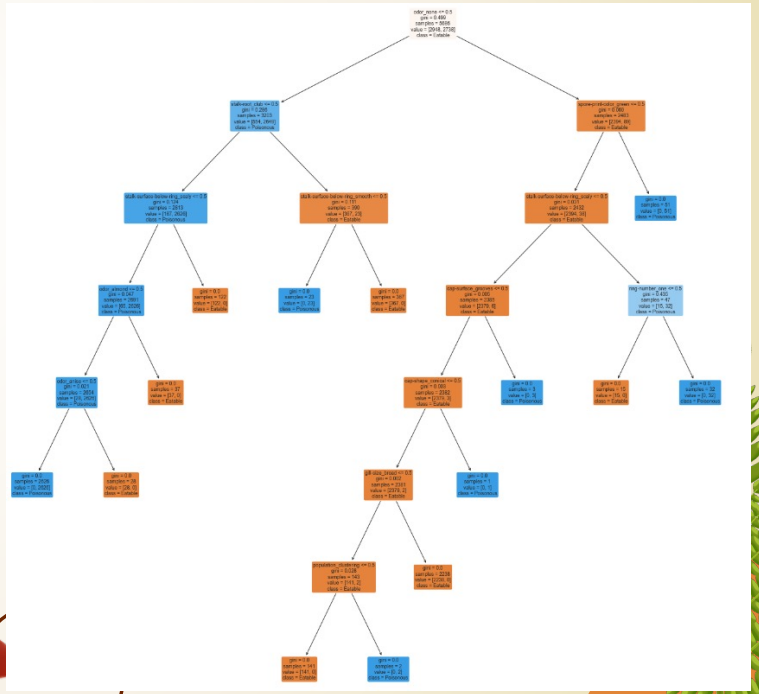


Decision Tree

Depth 7



Depth 8

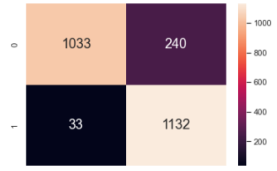


Decision Tree – Confusion Matrix (For test set)

Depth 1

```
Test Data
Accuracy  :     0.8880229696472518

TPR Test :     0.9716738197424892
TNR Test :     0.8114689709347996

FPR Test :     0.1885310290652003
FNR Test :     0.0283261802575107 3
```

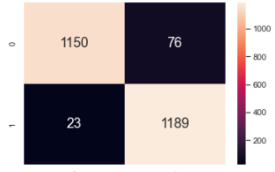
Out[227]: <AxesSubplot:>



Depth 2

```
Test Data
Accuracy  :     0.9593929450369155

TPR Test :     0.981023102310231
TNR Test :     0.938009787928221 9

FPR Test :     0.0619902120717781 4
FNR Test :     0.0189768976897689 78
```

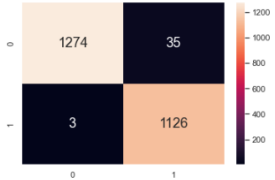
Out[231]: <AxesSubplot:>



Depth 3

```
Test Data
Accuracy  :     0.9844134536505332

TPR Test :     0.9973427812223207
TNR Test :     0.973262032085561 5

FPR Test :     0.0267379679144385 02
FNR Test :     0.0026572187776793 62
```

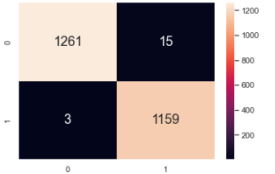
Out[234]: <AxesSubplot:>



Depth 4

```
Test Data
Accuracy  :     0.992616899097621

TPR Test :     0.9974182444061962
TNR Test :     0.988244514106583 1

FPR Test :     0.0117554858934169 29
FNR Test :     0.0025817555938037 868
```

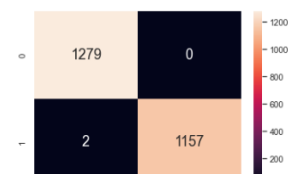
Out[237]: <AxesSubplot:>



Depth 5

```
Test Data
Accuracy  :     0.9991796554552912

TPR Test :     0.998274374460742
TNR Test :     1.0

FPR Test :     0.0
FNR Test :     0.0017256255392579 81
```

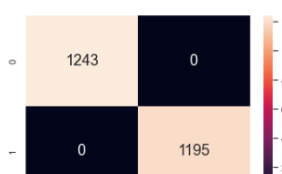
Out[240]: <AxesSubplot:>



Depth 6

```
Test Data
Accuracy  :     1.0

TPR Test :     1.0
TNR Test :     1.0

FPR Test :     0.0
FNR Test :     0.0
```

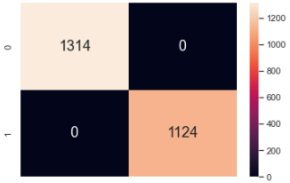
Out[243]: <AxesSubplot:>



Depth 7

```
Test Data
Accuracy  :     1.0

TPR Test :     1.0
TNR Test :     1.0

FPR Test :     0.0
FNR Test :     0.0
```

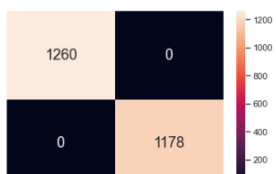
Out[246]: <AxesSubplot:>



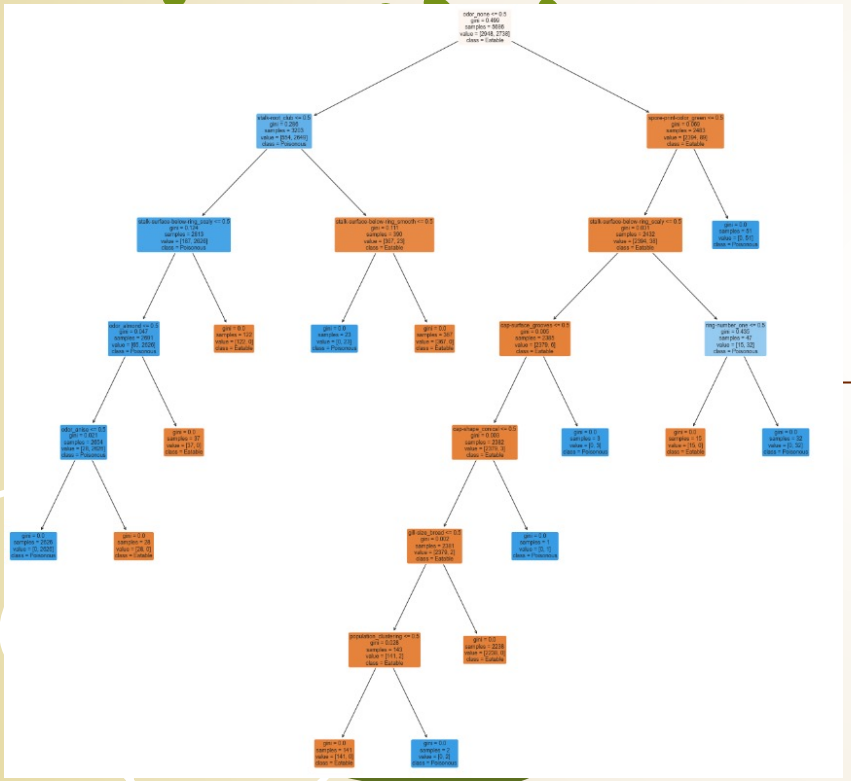
Depth 8

```
Test Data
Accuracy  :     1.0

TPR Test :     1.0
TNR Test :     1.0

FPR Test :     0.0
FNR Test :     0.0
```

Out[249]: <AxesSubplot:>



Depth 8



TPR rate for test

01

1.0

Important Feature we got

02

- odor_none
- stalk-root_club
- stalk-surface-below-ring_scaly
- odor_anise
- odor_almond
- ring-number_one
- spore-print-color_green
- stalk-surface-below-ring_scaly
- cap-surface_grooves
- cap-shape_conical
- gill-size_broad
- population_clustering
- ring-type_evanescent

Random Forest

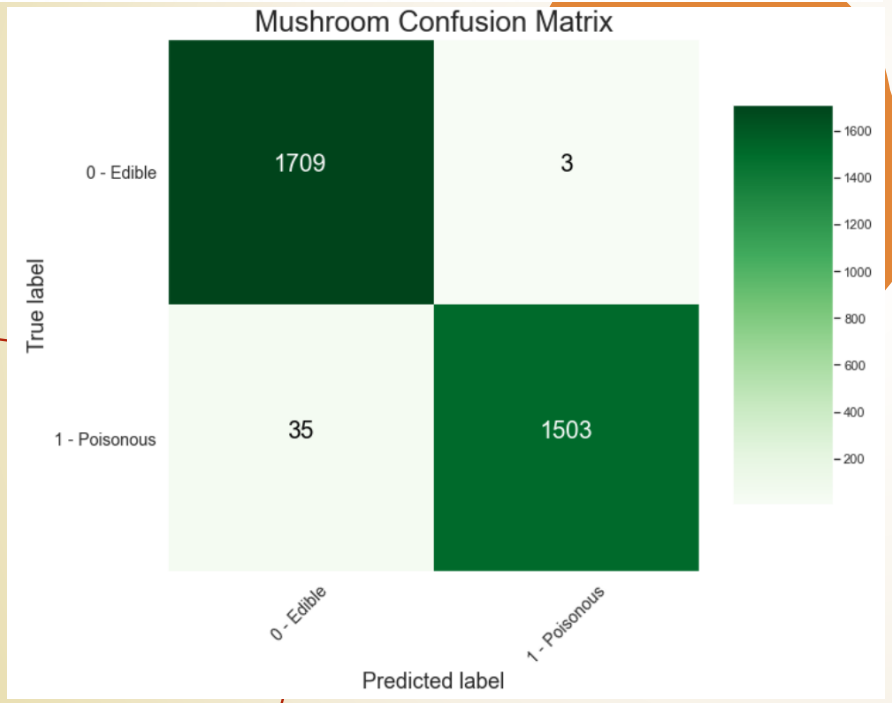
First Model

```
rf_classifier = RandomForestClassifier(
                        min_samples_leaf=50,
                        n_estimators=150,
                        bootstrap=True,
                        oob_score=True,
                        n_jobs=-1,
                        random_state=seed,
                        max_features='auto')
```

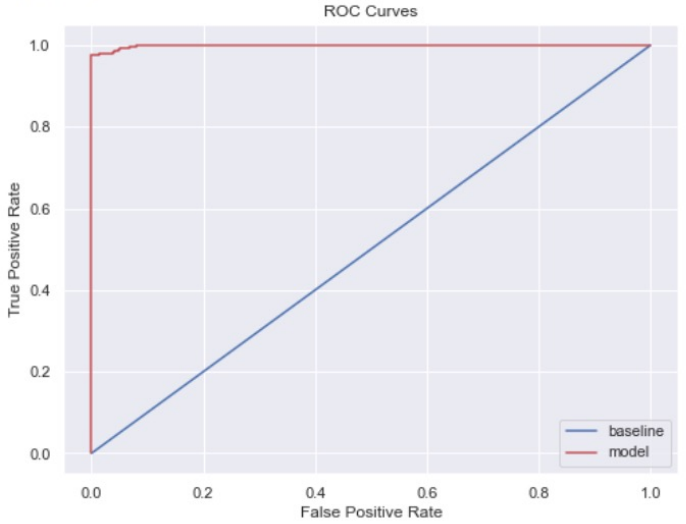
We construct the random forest with **150 decision trees**.

ROC Curve & Confusion Matrix

```
Train ROC AUC Score: 0.9989009846308069
Test ROC AUC  Score: 0.999158774565623
```

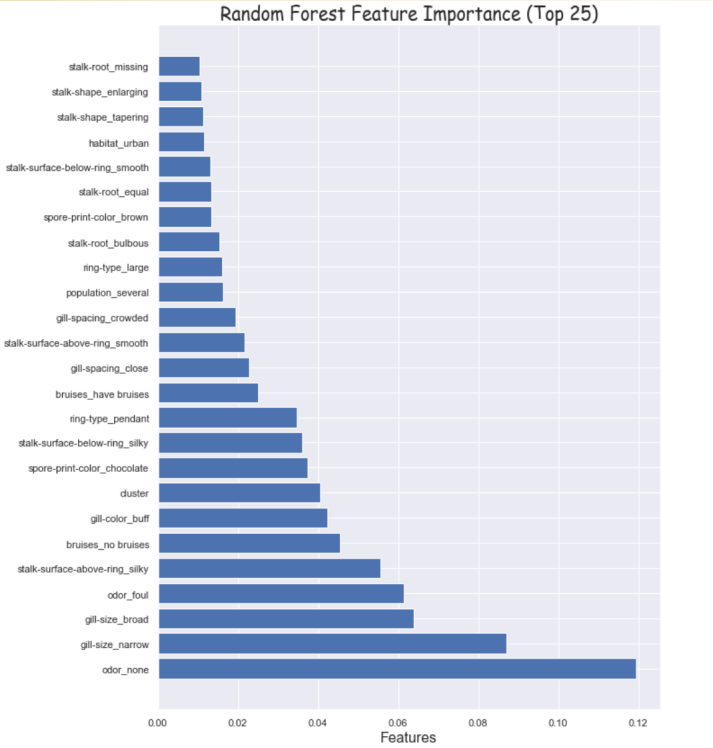



```
Baseline: 1.0
 Test: 0.98
 Train: 0.98
Baseline: 0.47
 Test: 1.0
 Train: 1.0
Baseline: 0.5
 Test: 1.0
 Train: 1.0
```



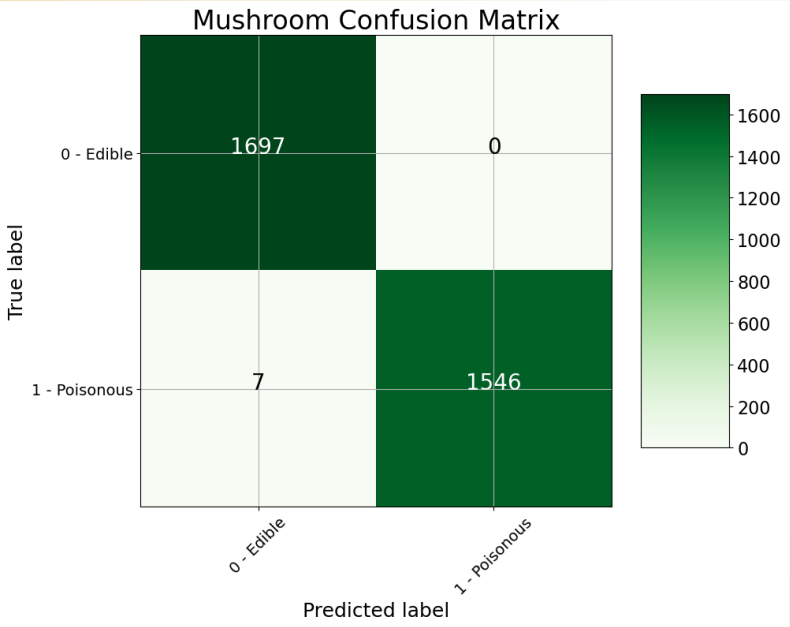

```
The accuracy of the model is 99.0 %
```

Important characterisics From Random Forest



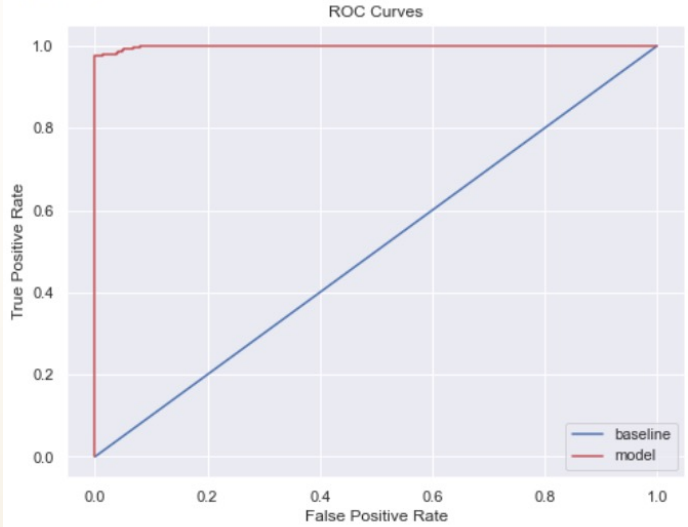
- Odor_none
- Gill-size_narrow
- Gill-size_broad
- Odor_foul
- Stalk-surface-above-ring_silky
- Bruises_no bruises
- Gill-color_buff
- Cluster
- Spore-print-color_chocolate
- Stalk-surface-below-ring_silky
- Ring-type_pendant
- Bruises_have bruises
- Gill-spacing_close
- Stalk-surface-above-ring_smooth
- Gill-spacing_crowede
- Population_several
- Ring-type_large
- Stalk-root_bulbous
- Spore-print-color_brown
- Stalk-root_equal
- Stalk-surface-below-ring_smooth
- Habitat_urban
- Stalk-shape_tapering
- Stalk-shape_enlarging
- Stalk-root_missing

To be BETTER ...



Adjusting the parameters to improve our model.

```
Baseline: 1.0
 Test: 0.98
 Train: 0.98
Baseline: 0.47
 Test: 1.0
 Train: 1.0
Baseline: 0.5
 Test: 1.0
 Train: 1.0
```



04

AI model

Image Manipulation



Increase the number of unique training examples by introducing small **distortions** to the images

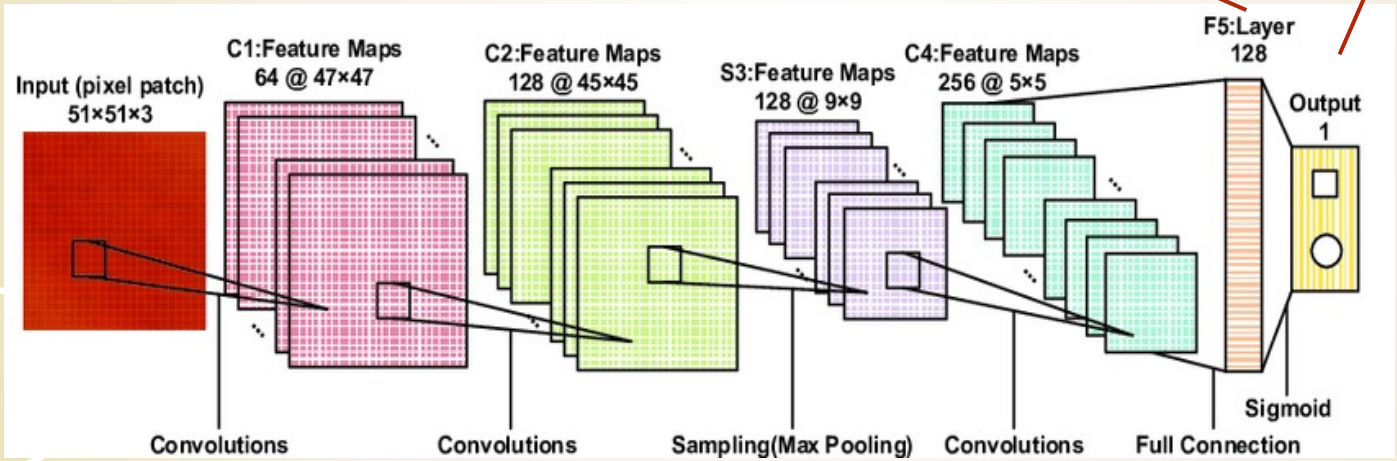


CNN Model Architecture

3 convolutional layers each followed by a pooling layer

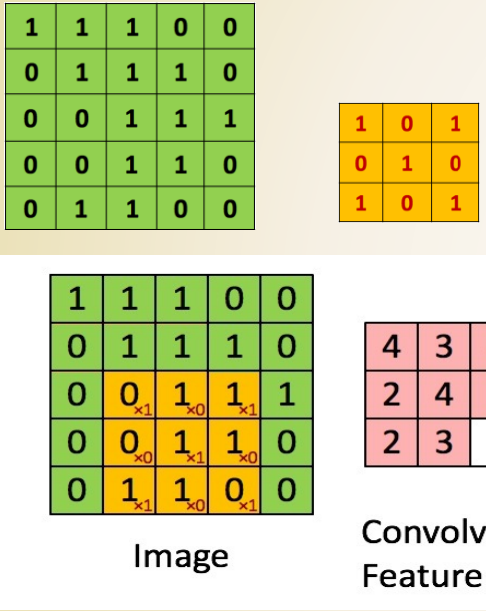
A fully connected layer

An output layer with a **single node**, output a probability value



CNN Model Architecture

Model Explanation -- Convolutional layers



Matrix Convolution

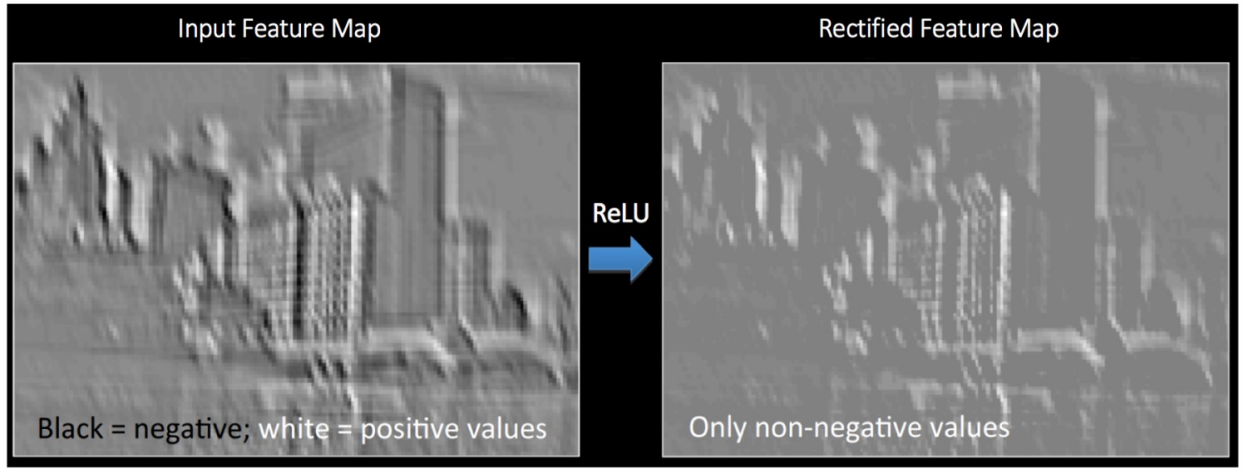
- Every image can be considered as a matrix of pixel values.
- We use the 3*3 matrix (**filter**) to slide over the original image by 1 pixel (**stride**) and for every position we will compute an integer which forms a single element of the output matrix (**Feature Map**)

```
model.add(Conv2D(32, (3, 3),
model.add(Conv2D(32, (3, 3)))
model.add(Conv2D(64, (3, 3)))
```

Our Model

- The depths of our three convolutional layers are 32,32 and 64.

ReLU activation & Max Pooling



ReLU: Introduce non-linearity in ConvNet.

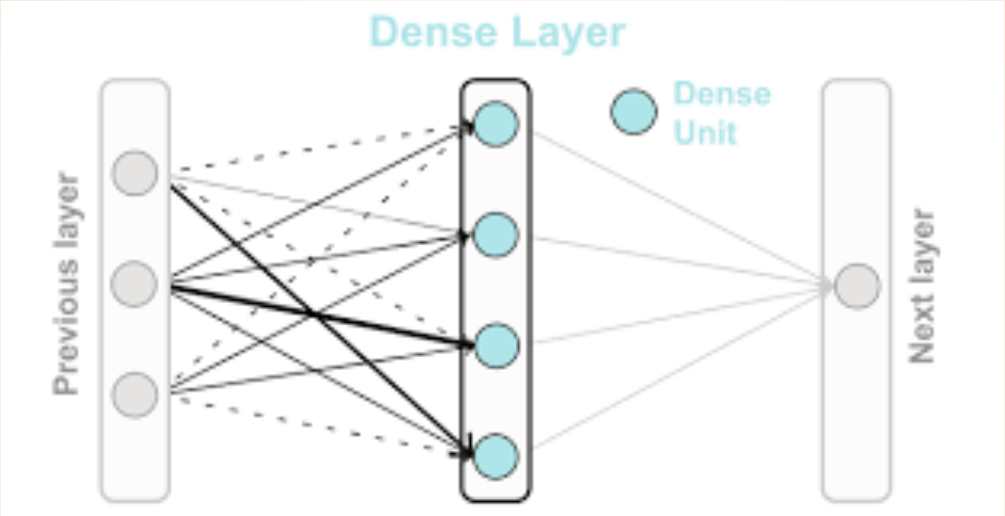
12	20	30	0
8	12	2	0
34	70	37	4
112	100	25	12

2×2 Max-Pool

20	30
112	37

Max Pooling: Reduces the dimensionality of each feature map but remains the most important information

Dense Layers & Output



Dense Layer: Receive input from all the previous layer and classify image.

Output: Probability between 0 and 1

Final Model

```
Epoch 44/50
280/280 [==============================] - 313s 1s/step - loss: 0.4223 - accuracy: 0.8287
- val_loss: 0.4899 - val_accuracy: 0.8062
Epoch 45/50
280/280 [==============================] - 314s 1s/step - loss: 0.3881 - accuracy: 0.8412
- val_loss: 0.3850 - val_accuracy: 0.8634
Epoch 46/50
280/280 [==============================] - 318s 1s/step - loss: 0.3934 - accuracy: 0.8448
- val_loss: 0.5451 - val_accuracy: 0.8375
Epoch 47/50
280/280 [==============================] - 314s 1s/step - loss: 0.4152 - accuracy: 0.8410
- val_loss: 1.5599 - val_accuracy: 0.7384
Epoch 48/50
280/280 [==============================] - 313s 1s/step - loss: 0.3761 - accuracy: 0.8428
- val_loss: 0.4646 - val_accuracy: 0.8045
Epoch 49/50
280/280 [==============================] - 307s 1s/step - loss: 0.4103 - accuracy: 0.8442
- val_loss: 0.3176 - val_accuracy: 0.8527
Epoch 50/50
280/280 [==============================] - 315s 1s/step - loss: 0.3984 - accuracy: 0.8381
- val_loss: 0.8348 - val_accuracy: 0.7286
```

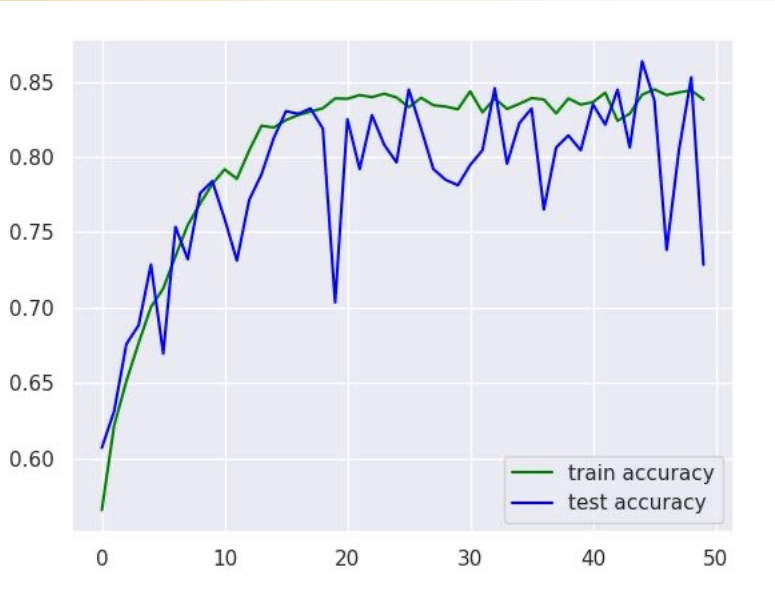
Final Model

```
Epoch 50/50
280/280 [==============================] - 315s 1s/step - loss: 0.3984 - accuracy: 0.8381
- val_loss: 0.8348 - val_accuracy: 0.7286
```

- Val_: test set
- **Loss**: The difference between the predicted output and the true output.
- **Accuracy**: It is calculated by dividing the number of correct predictions by the total number of predictions.

Training and Test

```
epochs = 50 # the
batch_size = 16 #
```

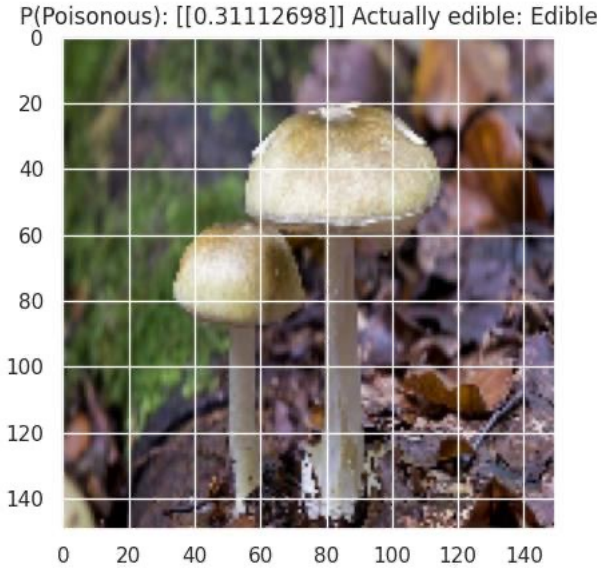


- We split the image into train and test sets with 0.8:0.2 portion.
- Then the model will be trained for a total of 50 passes over the entire training dataset. (i.e., the network was trained on 50 epochs)
- During each iteration of training, the model will process and update its weights based on 16 samples from the training dataset.

- As the train accuracy reaches a stage at **85%** , the testing accuracy waves **around 80%.**

05 Applications

AI Prediction-Application



- User can upload an image, and our AI model will output a single value describing the probability that a mushroom is poisonous
- The left image shows the prediction of one test case (outside the original image set).

User input

```
new_mushroom['ring-number'] = input('Enter the ring number of the mushroom (none,one,two): ')
new_mushroom['ring-type'] = input('Enter the ring type of the mushroom (cobwebby,evanescent,flaring,large,none,pendant,sheathing,
new_mushroom['spore-print-color'] = input('Enter the spore print color of the mushroom (black,brown,buff,chocolate,green,orange,
new_mushroom['population'] = input('Enter the population of the mushroom (abundant,clustered,numerous,scattered,several,solitary)
new_mushroom['habitat'] = input('Enter the habitat of the mushroom (grasses,leaves,meadows,paths,urban,waste,woods): ')
```

```
Enter the cap shape of the mushroom (bell,conical,convex,flat, knobbed,sunken):
```

```
Enter the cap shape of the mushroom (bell, conical, convex, flat, knobbed, sunken): convex
Enter the cap surface of the mushroom (fibrous, grooves, scaly, smooth): scaly
Enter the cap color of the mushroom (brown, buff, cinnamon, gray, green, pink, purple, red, white, yellow): white
Does the mushroom have bruises? (have bruises/no bruises): have bruises
Enter the odor of the mushroom (almond, anise, creosote, fishy, foul, musty, none, pungent, spicy): pungent
Enter the gill attachment of the mushroom (attached, descending, free, notched): free
Enter the gill spacing of the mushroom (close, crowded, distant): close
Enter the gill size of the mushroom (broad, narrow): narrow
Enter the gill color of the mushroom (black, brown, buff, chocolate, gray, green, orange, pink, purple, red, white, yellow): pink
Enter the stalk shape of the mushroom (enlarging, tapering): enlarging
Enter the stalk root of the mushroom (bulbous, club, cup, equal, rhizomorphs, rooted, missing): equal
Enter the stalk surface above the ring of the mushroom (fibrous, scaly, silky, smooth): smooth
Enter the stalk surface below the ring of the mushroom (fibrous, scaly, silky, smooth): smooth
Enter the stalk color above the ring of the mushroom (brown, buff, cinnamon, gray, orange, pink, red, white, yellow): white
Enter the stalk color below the ring of the mushroom (brown, buff, cinnamon, gray, orange, pink, red, white, yellow): white
Enter the veil type of the mushroom (partial, universal): partial
Enter the veil color of the mushroom (brown, orange, white, yellow): white
Enter the ring number of the mushroom (none, one, two): one
Enter the ring type of the mushroom (cobwebby, evanescent, flaring, large, none, pendant, sheathing, zone): pendant
Enter the spore print color of the mushroom (black, brown, buff, chocolate, green, orange, purple, white, yellow): black
Enter the population of the mushroom (abundant, clustered, numerous, scattered, several, solitary): several
Enter the habitat of the mushroom (grasses, leaves, meadows, paths, urban, waste, woods): grasses
```

```
1  new_mushroom_df = pd.DataFrame(new_mushroom, index=[0])
2
3  new_mushroom_df = new_mushroom_df.reindex(columns=X.columns, fill_value=0)  # align with training data columns
4
```

Result

- **Prediction '0'**: the mushroom is classified as **edible**
- **Otherwise**, classified as **poisonous**.

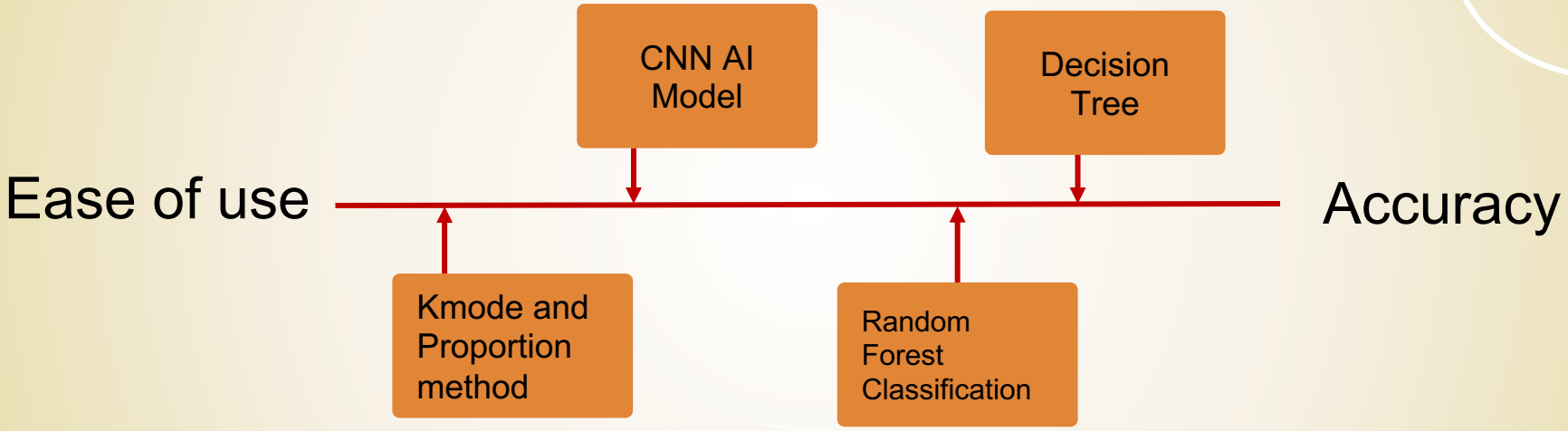
```
prediction = tree_clf.predict(encoded_new_mushroom_df)
if prediction == 0:
    print('The mushroom is edible.')
else:
    print('The mushroom is poisonous.')

```

```
The mushroom is poisonous.
```

06

Conclusion and Evaluation

Findings


CNN AI Model
Decision Tree
Ease of use
Accuracy
Kmode and Proportion method
Random Forest Classification

Takeaways

Utility Methods

1. **OneHotEncoder** (Transformation technique to convert categorical data into numerical format)
2. **PipeLine** (Chaining together multiple machine learning steps into a single object)
3. **KMode** (Method to analyse categorical data)
4. **Fit-Transform** (Standardise data)

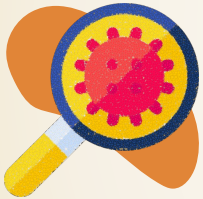
Models

1. **Random forest classification** (Machine Learning Algorithm)
2. **CNN** (artificial neural network specializes in visual deta)

Future Improvements

Dataset's flaw

- Only 1 set of data on 2 families of mushrooms



AI model's flaw

- Since the accuracy of the AI model is **80%** which can still be improve (compare with the decision tree and random forest)

Dataset's improvement

- Use a larger and more balanced dataset

AI improvement

- adjust some parameters (e.g.: epoch num)
- More complete image set

Thanks!

Risk Warning: Although our model's accuracy is high, we still strongly advise against eating any wild mushrooms.

Reference

- Bonthu, H. (2023, April 18). KModes clustering algorithm for categorical data. *Analytics Vidhya.* https://www.analyticsvidhya.com/blog/2021/06/kmodes-clustering-algorithm-for-categorical-data/
- Chollet, B. F. (n.d.). Building powerful image classification models using very little data. *The Keras Blog ATOM.* https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html
- Koivisto, T., Nieminen, T., & Harjunpää, J. (2018). *Deep Shrooms: classifying mushroom images.* https://tuomonieminen.github.io/deep-shrooms/
- Li, H., Zhang, Y., Zhang, H., Zhou, J., Liang, J., Yin, Y., He, Q., Jiang, S., Zhang, Y., Yuan, Y., Lang, N., Cheng, B., Wang, M., & Sun, C. (2023). Mushroom poisoning outbreaks — China, 2022. *China CDC Weekly*, 5(3), 45–50. https://doi.org/10.46234/ccdcw2023.009
- Li, H., Zhang, H., Zhang, Y., Zhang, K., Zhou, J., Yin, Y., Jiang, S., Ma, P., He, Q., Zhang, Y., Wen, K., Yuan, Y., Lang, N., Lu, J., & Sun, C. (2020, January 1). Mushroom poisoning outbreaks - China, 2019. *China CDC Weekly.* https://weekly.chinacdc.cn/en/article/doi/10.46234/ccdcw2020.005#:~:text=At%20least%20100%20estimated%20people,China%20(2%2D5).

Reference

- Nuse, I. P., & Christensen, A. (2016, September 13). Finding mushrooms with your mobile phone. *Sciencenorway.* https://sciencenorway.no/forskningno-norway-smartphone/finding-mushrooms-with-your-mobile-phone/1437329
- Sandhyakrishnan02. (2022, March 1). Mushroom Classification - Decision Tree Classifier. *Kaggle.* https://www.kaggle.com/code/sandhyakrishnan02/mushroom-classification-decision-tree-classifier/notebook#11.-Decision-Tree-Creation
- scikit-learn developers. (2023). *sklearn.preprocessing.LabelEncoder.* Retrieved April 22, 2023, from https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html
- Some disturbing facets of the super mario universe. (n.d.). *VGJUNK.* Retrieved April 23, 2023, from http://retrovania-vgjunk.blogspot.com/2012/08/some-disturbing-facets-of-super-mario.html

Reference

- Tran, J. (2021, December 14). Random Forest Classifier in Python - Towards Data Science. *Medium.* https://towardsdatascience.com/my-random-forest-classifier-cheat-sheet-in-python-fedb84f8cf4f
- Ujjwalkarn. (2017, May 29). An Intuitive Explanation of Convolutional Neural Networks. *Ujjwal Karn.* https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/
- Wikipedia contributors. (2023). Convolutional neural network. *Wikipedia.* https://en.wikipedia.org/wiki/Convolutional_neural_network
- Wild Food UK. (2023, February 13). *Wild UK mushrooms (fungi): Guide to identification & picking.* Retrieved April 23, 2023, from https://www.wildfooduk.com/mushroom-guide/