Mushrooms EDA

Exploratory Data Analysis

Author:

- Yasser Abduallah – 118940309
- Jason Miller – 118940309
- Ali Lotfi – 118940309

This dataset describes mushroom species above around thousand types of mushrooms, both edible and poisonous. The data contains attributes like the cap shape, surface, color, gill attachment, spacing, size, colors, bruises, odor, shape, root, surface below ground, color below ring, ring number, ring type, spore print color, population, habitat.

Presented

Topic Sent: 21-10940309

Revue Stage: 21-10940309

HW Done: 21-10940309

Data Exploration

Importing Important Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
rcParams['figure.figure.max_open_warning'] = 0
```

Loading The data

```
df = pd.read_csv('/content/mushrooms.csv')
```

```
df.shape
```

```
(8124, 23)
```

```
df.head(5)
```

class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	stalk-shape	stalk-root	stalk-surface-above-ring	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	ring-number	ring-type	spore-print-color	population	habitat
p	x	s	n	t	p	f	c	n	k	e	e	s	s	w	w	p	w	o	p	k	s	u
e	x	s	y	t	a	f	c	b	k	e	c	s	s	w	w	p	w	o	p	n	n	g
e	b	s	w	t	l	f	c	b	n	e	c	s	s	w	w	p	w	o	p	n	n	m
p	x	y	w	t	p	f	c	n	n	e	e	s	s	w	w	p	w	o	p	k	s	u
e	x	s	g	f	n	f	w	b	k	t	e	s	s	w	w	p	w	o	e	n	a	g

5 rows × 23 columns

```
df.info()
```

```
<class 'DataFrame'>
RangeIndex: 8124 entries, 0 to 8123
Data columns (total 23 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   class                     8124 non-null   object
 1   cap-shape                 8124 non-null   object
 2   cap-surface               8124 non-null   object
 3   cap-color                 8124 non-null   object
 4   bruises                   8124 non-null   object
 5   odor                      8124 non-null   object
 6   gill-attachment           8124 non-null   object
 7   gill-spacing              8124 non-null   object
 8   gill-size                 8124 non-null   object
 9   gill-color                8124 non-null   object
 10  stalk-shape               8124 non-null   object
 11  stalk-root                8124 non-null   object
 12  stalk-surface-above-ring  8124 non-null   object
 13  stalk-surface-below-ring  8124 non-null   object
 14  stalk-color-above-ring    8124 non-null   object
 15  stalk-color-below-ring    8124 non-null   object
 16  veil-type                 8124 non-null   object
 17  veil-color                8124 non-null   object
 18  ring-number               8124 non-null   object
 19  ring-type                 8124 non-null   object
 20  spore-print-color         8124 non-null   object
 21  population                8124 non-null   object
 22  habitat                   8124 non-null   object
dtype: object
```

```
df.describe(include='all')
```

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	stalk-shape	stalk-root	stalk-surface-above-ring	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	ring-number	ring-type	spore-print-color	population	habitat
count	8124	8124	8124	8124	8124	8124	8124	8124	8124	8124	8124	8124	8124	8124	8124	8124	8124	8124	8124	8124	8124	8124	8124
unique	2	6	4	10	2	9	2	2	2	12	2	5	4	4	9	9	1	4	3	5	9	6	7
top	e	x	y	n	f	n	f	c	n	b	t	b	s	s	w	w	p	w	o	p	w	v	d
freq	4208	3656	3244	2284	4748	3528	7914	6812	5612	1728	4608	3776	5176	4936	4464	4384	8124	7924	7488	3968	2388	4040	3148

```
df.isnull().sum()
```

```
class                       0
cap-shape                   0
cap-surface                 0
cap-color                   0
bruises                     0
odor                        0
gill-attachment             0
gill-spacing                0
gill-size                   0
gill-color                  0
stalk-shape                 0
stalk-root                  0
stalk-surface-above-ring    0
stalk-surface-below-ring    0
stalk-color-above-ring      0
stalk-color-below-ring      0
veil-type                   0
veil-color                  0
ring-number                 0
ring-type                   0
spore-print-color           0
population                  0
habitat                     0
dtype: object
```

There are no empty cells, but there are missing values that are displayed like this '?'

```
df.isin(['?']).sum(axis=0)
```

```
class                       0
cap-shape                   0
cap-surface                 0
cap-color                   0
bruises                     0
odor                        0
gill-attachment             0
gill-spacing                0
gill-size                   0
gill-color                  0
stalk-shape                 0
stalk-root                  2480
stalk-surface-above-ring    0
stalk-surface-below-ring    0
stalk-color-above-ring      0
stalk-color-below-ring      0
veil-type                   0
veil-color                  0
ring-number                 0
ring-type                   0
spore-print-color           0
population                  0
habitat                     0
dtype: object
```

Before we continue to explore the data, it's important to identify what each column in the table represents

Attribute Information:

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

Let's examine the distribution of the data in all our columns using a visualization

```
%matplotlib inline
plt.rcParams['figure.max_open_warning'] = 0
# set the background for the graph
sns.set(rc={'axes.facecolor':'#ffffff',
            'figure.facecolor':'#eeeeee',
            'axes.labelcolor':'black',
            'xtick.color':'black',
            'ytick.color':'black',
            'text.color':'black'})
for col in df.columns:
    sns.catplot(x=col, kind="count",palette="ch:.25", data=df,
                edecolor="black",
                linewidth=2,
                height=5,
                aspect=1.4,
                legend=False
                ).fig.suptitle('All Columns Distribution',
                               fontweight='bold',fontsize=20).set(
                               ).set_ylabels("Count",fontsize=15).set_xlabels(col,fontsize=15).
set_xticklabels(rotation=90, ha='center', fontsize=12).set_yticklabels(fontsize=12).
set_titles("{col_name}").
plt.tight_layout(w_pad=1, h_pad=1).
plt.subplots_adjust(top=0.9, wspace=0.1, hspace=0.1).
plt.show()
```

We can see that the 'veil-type' column is not distributed and contains only one value, therefore this column doesn't contribute to our data analysis, so we will delete it from the dataset

```
df.drop(columns=['veil-type'],inplace=True)
```

```
df.columns.shape
```

```
(23,)
```

```
['class', 'cap-shape', 'cap-surface', 'cap-color',
       'bruises', 'odor', 'gill-attachment', 'gill-spacing',
       'gill-size', 'gill-color', 'stalk-shape', 'stalk-root',
       'stalk-surface-above-ring', 'stalk-surface-below-ring',
       'stalk-color-above-ring', 'stalk-color-below-ring',
       'veil-type', 'veil-color', 'ring-number', 'ring-type',
       'spore-print-color', 'population', 'habitat']
```

To make this analysis more understandable, we will substitute all values in full names

```
df = df.replace({
    'class': {'p': 'poisonous', 'e': 'edible'},
    'cap-shape': {'b': 'bell', 'c': 'conical', 'x': 'convex', 'f': 'flat', 'k': 'knobbed', 's': 'sunken'},
    'cap-surface': {'f': 'fibrous', 'g': 'grooves', 'y': 'scaly', 's': 'smooth'},
    'cap-color': {'n': 'brown', 'b': 'buff', 'c': 'cinnamon', 'g': 'gray', 'r': 'green', 'p': 'pink', 'u': 'purple', 'e': 'red', 'w': 'white', 'y': 'yellow'},
    'bruises': {'t': 'bruises', 'f': 'no'},
    'odor': {'a': 'almond', 'l': 'anise', 'c': 'creosote', 'y': 'fishy', 'f': 'foul', 'm': 'musty', 'n': 'none', 'p': 'pungent', 's': 'spicy'},
    'gill-attachment': {'a': 'attached', 'd': 'descending', 'f': 'free', 'n': 'notched'},
    'gill-spacing': {'c': 'close', 'w': 'crowded', 'd': 'distant'},
    'gill-size': {'b': 'broad', 'n': 'narrow'},
    'gill-color': {'k': 'black', 'n': 'brown', 'b': 'buff', 'h': 'chocolate', 'g': 'gray', 'r': 'green', 'o': 'orange', 'p': 'pink', 'u': 'purple', 'e': 'red', 'w': 'white', 'y': 'yellow'},
    'stalk-shape': {'e': 'enlarging', 't': 'tapering'},
    'stalk-root': {'b': 'bulbous', 'c': 'club', 'u': 'cup', 'e': 'equal', 'z': 'rhizomorphs', 'r': 'rooted', '?': 'missing'},
    'stalk-surface-above-ring': {'f': 'fibrous', 'y': 'scaly', 'k': 'silky', 's': 'smooth'},
    'stalk-surface-below-ring': {'f': 'fibrous', 'y': 'scaly', 'k': 'silky', 's': 'smooth'},
    'stalk-color-above-ring': {'n': 'brown', 'b': 'buff', 'c': 'cinnamon', 'g': 'gray', 'o': 'orange', 'p': 'pink', 'e': 'red', 'w': 'white', 'y': 'yellow'},
    'stalk-color-below-ring': {'n': 'brown', 'b': 'buff', 'c': 'cinnamon', 'g': 'gray', 'o': 'orange', 'p': 'pink', 'e': 'red', 'w': 'white', 'y': 'yellow'},
    'veil-type': {'p': 'partial', 'u': 'universal'},
    'veil-color': {'n': 'brown', 'o': 'orange', 'w': 'white', 'y': 'yellow'},
    'ring-number': {'n': 'none', 'o': 'one', 't': 'two'},
    'ring-type': {'c': 'cobweb', 'e': 'evanescent', 'f': 'flaring', 'l': 'large', 'n': 'none', 'p': 'pendant', 's': 'sheathing', 'z': 'zone'},
    'spore-print-color': {'k': 'black', 'n': 'brown', 'b': 'buff', 'h': 'chocolate', 'r': 'green', 'o': 'orange', 'u': 'purple', 'e': 'red', 'w': 'white', 'y': 'yellow'},
    'population': {'a': 'abundant', 'c': 'clustered', 'n': 'numerous', 's': 'scattered', 'v': 'several', 'y': 'solitary'},
    'habitat': {'g': 'grasses', 'l': 'leaves', 'm': 'meadows', 'p': 'paths', 'u': 'urban', 'w': 'waste', 'd': 'woods'}
})
```

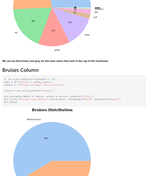
```
df.head(5)
```

class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	stalk-shape	stalk-root	stalk-surface-above-ring	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	ring-number	ring-type	spore-print-color	population	habitat
poisonous	convex	smooth	brown	bruises	pungent	free	close	narrow	black	enlarging	equal	smooth	smooth	white	white	partial	white	one	pendant	black	scattered	urban
edible	convex	smooth	yellow	bruises	almond	free	close	broad	black	enlarging	club	smooth	smooth	white	white	partial	white	one	pendant	brown	numerous	grasses
edible	bell	smooth	white	bruises	anise	free	close	broad	brown	enlarging	club	smooth	smooth	white	white	partial	white	one	pendant	brown	numerous	meadows
poisonous	convex	smooth	yellow	bruises	pungent	free	close	narrow	black	enlarging	equal	smooth	smooth	white	white	partial	white	one	pendant	black	scattered	urban
edible	convex	smooth	green	no	none	free	crowded	broad	black	tapering	equal	smooth	smooth	white	white	partial	white	none	evanescent	brown	abundant	grasses

Now we will start to examine the information obtained from our columns using different visualizations

Classification column

```
# Pie chart
labels = df['class'].value_counts(sort = True).index
sizes = df['class'].value_counts(sort = True)
colors = ["SkyBlue","PeachPuff"]
plt.rcParams.update({'font.size': 15})
plt.pie(sizes,
        labels=labels,
        autopct="%1.1f%",
        startangle=90,
        pctdistance=0.6,
        shadow = False,
        colors=colors,
        frame=False,
        labeldistance=1.1,
        wedgeprops={"edgecolor":"0.7", 'linewidth': 0.0, 'linestyle': 'solid', 'antialiased': True},
        explode=[0,0.1]
        )
plt.title('Class',{'fontweight':'bold','fontsize':20,'fontfamily':'sans-serif'})
plt.axis('equal')
plt.legend(frame=False,bbox_to_anchor=(1.2,0.8))
plt.show()
```



Our data is split almost 50% 50% poisonous and edible

cap-shape Column

```
# Count the occurrences of each value in the 'cap-shape' column
counts = df['cap-shape'].value_counts()
labels = counts.index
sizes = counts.values

plt.rcParams.update({'font.size': 15})
plt.pie(sizes, labels=labels, autopct='%1.1f%%', shadow=False, startangle=90, frame=False, labeldistance=1.1, wedgeprops={'linewidth': 0.0, 'linestyle': 'solid', 'antialiased': True})
plt.title('Cap-Shape Distribution', {'fontweight':'bold','fontsize':20,'fontfamily':'sans-serif'})
plt.axis('equal')
plt.legend(frame=False,bbox_to_anchor=(1.2,0.8))
plt.show()
```

It can be concluded from the graph that convex is the popular cap shape in mushrooms, and there is almost no conical shape in mushrooms.

cap-surface column

```
# Count the occurrences of each value in the 'cap-surface' column
counts = df['cap-surface'].value_counts()
labels = counts.index
sizes = counts.values

plt.rcParams.update({'font.size': 15})
plt.pie(sizes, labels=labels, autopct='%1.1f%%', shadow=False, startangle=90, frame=False, labeldistance=1.1, wedgeprops={'linewidth': 0.0, 'linestyle': 'solid', 'antialiased': True})
plt.title('Cap Surface Distribution', {'fontweight':'bold','fontsize':20,'fontfamily':'sans-serif'})
plt.axis('equal')
plt.legend(frame=False,bbox_to_anchor=(1.2,0.8))
plt.show
```

We can see that there are more mushrooms with a scaly surface on their cap

cap-color Column

```
# Count the occurrences of each value in the 'cap-color' column
counts = df['cap-color'].value_counts()
labels = counts.index
sizes = counts.values

plt.rcParams.update({'font.size': 15})
plt.pie(sizes, labels=labels, autopct='%1.1f%%', shadow=False, startangle=90, frame=False, labeldistance=1.1, wedgeprops={'linewidth': 0.0, 'linestyle': 'solid', 'antialiased': True})
plt.title('Cap Color Distribution', {'fontweight':'bold','fontsize':20,'fontfamily':'sans-serif'})
plt.axis('equal')
plt.legend(frame=False,bbox_to_anchor=(1.2,0.8))
plt.show
```

We can see that brown and gray are the main colors that exist in the cap of the mushroom

Bruises Column

```
# Count the occurrences of each value in the 'bruises' column
counts = df['bruises'].value_counts
labels = counts.index
sizes = counts.values

plt.rcParams.update({'font.size': 15})
plt.pie(sizes, labels=labels, autopct='%1.1f%%', shadow=False, startangle=90, frame=False, labeldistance=1.1, wedgeprops={'linewidth': 0.0, 'linestyle': 'solid', 'antialiased': True})
plt.title('Bruises Distribution', {'fontweight':'bold','fontsize':20,'fontfamily':'sans-serif'})
plt.axis('equal')
plt.legend(frame=False,bbox_to_anchor=(1.2,0.8))
plt.show
```

Odor Column

Let's check what poisonous mushrooms smell like and what edible mushrooms smell like

```
# Filter the DataFrame to only include poisonous mushrooms
df_poisonous = df[df['class'] == 'poisonous']

# Count the occurrences of each value in the 'odor' column for poisonous mushrooms
counts_poisonous = df_poisonous['odor'].value_counts()

# Filter the DataFrame to only include edible mushrooms
df_edible = df[df['class'] == 'edible']

# Count the occurrences of each value in the 'odor' column for edible mushrooms
counts_edible = df_edible['odor'].value_counts()

# Create a figure with two subplots
fig, ax1, ax2 = plt.subplots(1, 2, figure=(10, 5))

# Plot the pie chart for edible mushrooms
ax1.pie(counts_edible, labels=counts_edible.index, autopct='%1.1f%%', shadow=False, startangle=90, frame=False, labeldistance=1.1, wedgeprops={'linewidth': 0.0, 'linestyle': 'solid', 'antialiased': True})
ax1.title('Edible', {'fontweight':'bold','fontsize':20,'fontfamily':'sans-serif'})
ax1.axis('equal')
ax1.legend(frame=False,bbox_to_anchor=(1.2,0.8))

# Plot the pie chart for poisonous mushrooms
ax2.pie(counts_poisonous, labels=counts_poisonous.index, autopct='%1.1f%%', shadow=False, startangle=90, frame=False, labeldistance=1.1, wedgeprops={'linewidth': 0.0, 'linestyle': 'solid', 'antialiased': True})
ax2.title('Poisonous', {'fontweight':'bold','fontsize':20,'fontfamily':'sans-serif'})
ax2.axis('equal')
ax2.legend(frame=False,bbox_to_anchor=(1.2,0.8))

plt.suptitle('Difference in mushrooms odor', {'fontweight':'bold','fontsize':20,'fontfamily':'sans-serif'})
plt.show
```

Never eat a mushroom smell fishy because it is probably poisonous. According to the dataset, edible mushrooms smell nothing or have almond or anise.

stalk-root column

```
df['stalk-root'].value_counts
```

```
2480
b    3776
e    1120
c     556
r     192
u     148
z     120
?     2480
Name: stalk-root, dtype: int64
```

out of 8124, there are 2480 missing values, now we will check how many of them are poisonous and how many are edible.

```
# Create a figure with two subplots
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 5))

# Plot the pie chart for edible mushrooms
ax1.pie(counts_edible, labels=counts_edible.index, autopct='%1.1f%%', shadow=False, startangle=90, frame=False, labeldistance=1.1, wedgeprops={'linewidth': 0.0, 'linestyle': 'solid', 'antialiased': True})
ax1.title('Edible', {'fontweight':'bold','fontsize':20,'fontfamily':'sans-serif'})
ax1.axis('equal')
ax1.legend(frame=False,bbox_to_anchor=(1.2,0.8))

# Plot the pie chart for poisonous mushrooms
ax2.pie(counts_poisonous, labels=counts_poisonous.index, autopct='%1.1f%%', shadow=False, startangle=90, frame=False, labeldistance=1.1, wedgeprops={'linewidth': 0.0, 'linestyle': 'solid', 'antialiased': True})
ax2.title('Poisonous', {'fontweight':'bold','fontsize':20,'fontfamily':'sans-serif'})
ax2.axis('equal')
ax2.legend(frame=False,bbox_to_anchor=(1.2,0.8))

plt.suptitle('Distribution of classification colours', {'fontweight':'bold','fontsize':20,'fontfamily':'sans-serif'})
plt.show
```

out of 2480 of the missing values, 1792 were belongs to poisonous class and 1120 belongs to edible class

Completing missing values

we will fill the missing values with the most frequent object

```
# Group by the "class" column and count the occurrences of each value in the "stalk-root" column
grouped = df.groupby('class')['stalk-root'].
# Find the most frequent value in each group
mode_values = grouped.apply(lambda x: x.mode()[0])
# Print the mode values
print(mode_values)
```

```
class
edible    b
poisonous    b
Name: stalk-root, dtype: object
```

```
# Replace missing values with the mode values
df['stalk-root'] = df['stalk-root'].fillna(df['class'].map(mode_values))
```

```
df['stalk-root'].value_counts
```

```
b    4056
e    1120
c     556
r     192
u     148
z     120
Name: stalk-root, dtype: int64
```

Population Column

```
In [ ]: # Population Column
        plt.figure(figure_size=(8,6))
        df['population'].value_counts(normalize=True).plot(kind='pie',
        plt.title('Population Distribution', fontsize=20)
        plt.legend(loc='upper right', bbox_to_anchor=(1.1, 1.1), ncolors
        plt.ylabel('population', fontsize=20)
        plt.xlabel('', fontsize=20)
        plt.tight_layout()
```

Habitat Column

Let's check the habitat of edible and poisonous mushrooms

```
In [ ]: # Habitat Column
        plt.figure(figure_size=(8,6))
        df['habitat'].value_counts(normalize=True).plot(kind='pie',
        plt.title('Habitat Distribution', fontsize=20)
        plt.legend(loc='upper right', bbox_to_anchor=(1.1, 1.1), ncolors
        plt.ylabel('habitat', fontsize=20)
        plt.xlabel('', fontsize=20)
        plt.tight_layout()
```

According to the given details, mushrooms from both the classes and genus are usually edible rather than poisonous.

Connections between columns

Correlation Matrix

Correlation matrix is a table which shows the correlation coefficients between variables. Each cell in the table shows the correlation between two variables.

```
In [ ]: # Correlation Matrix
        from sklearn.preprocessing import LabelEncoder

        # Create an instance of LabelEncoder
        le = LabelEncoder()

        # Apply the fit_transform method to the DataFrame
        df_encoded = df.apply(lambda x: le.fit_transform(x.astype(str)))

        # Calculate the correlation matrix
        corr_matrix = df_encoded.corr()

        # Print the correlation matrix
        print(corr_matrix)
```

```
                 class  cap-shape  cap-surface  cap-color  bruises  \
class         1.000000   0.001828    0.178011   0.098819  0.501625
cap-shape     0.001828   1.000000    0.011730   0.043123  0.004732
cap-surface   0.178011   0.011730    1.000000   0.046024  0.038772
cap-color     0.098819   0.043123    0.046024   1.000000  0.094559
bruises       0.501625   0.004732    0.038772   0.094559  1.000000
odor          0.712804   0.045772    0.182811   0.071466  0.302272
gill-attachment  0.129200   0.000477    0.037436   0.011337  0.010976
gill-spacing  0.348387   0.037553    0.065981   0.035943  0.150715
gill-size     0.540024   0.053828    0.148083   0.129666  0.419363
gill-color    0.530566   0.046592    0.133731   0.644811  0.181659
stalk-shape   0.102571   0.064755    0.054151   0.075195  0.075808
stalk-root    0.346201   0.038871    0.095772   0.039342  0.244756
stalk-surface-above-ring  0.334011   0.054677    0.293643   0.047226  0.332351
stalk-surface-below-ring  0.304780   0.054307    0.279755   0.046401  0.326966
stalk-color-above-ring  0.222799   0.046401    0.096730   0.645677  0.152836
stalk-color-below-ring  0.218987   0.046401    0.096730   0.645677  0.152836
veil-type     NaN        NaN         NaN        NaN       NaN
veil-color    0.145142   0.046401    0.096730   0.645677  0.152836
ring-number   0.219066   0.046401    0.096730   0.645677  0.152836
ring-type     0.411771   0.046401    0.096730   0.645677  0.152836
spore-print-color  0.346201   0.038871    0.095772   0.039342  0.244756
population    0.298686   0.046401    0.096730   0.645677  0.152836
habitat       0.107283   0.046401    0.096730   0.645677  0.152836

                 odor  gill-attachment  gill-spacing  gill-size  gill-color  \
class        0.712804         0.129200      0.348387   0.540024   0.530566
cap-shape    0.045772         0.000477      0.037553   0.053828   0.046592
cap-surface  0.182811         0.037436      0.065981   0.148083   0.133731
cap-color    0.071466         0.011337      0.035943   0.129666   0.644811
bruises      0.302272         0.010976      0.150715   0.419363   0.181659
odor         1.000000         0.070909      0.304351   0.398293   0.327324
gill-attachment  0.070909         1.000000      0.031046   0.038772   0.046592
gill-spacing 0.304351         0.031046      1.000000   0.148083   0.133731
gill-size    0.398293         0.038772      0.148083   1.000000   0.129666
gill-color   0.327324         0.046592      0.133731   0.129666   1.000000
stalk-shape  0.075808         0.064755      0.054151   0.075195   0.075808
stalk-root   0.244756         0.038871      0.095772   0.039342   0.244756
stalk-surface-above-ring  0.332351         0.054677      0.293643   0.047226   0.332351
stalk-surface-below-ring  0.326966         0.054307      0.279755   0.046401   0.326966
stalk-color-above-ring  0.152836         0.046401      0.096730   0.645677   0.152836
stalk-color-below-ring  0.152836         0.046401      0.096730   0.645677   0.152836
veil-type    NaN              NaN           NaN        NaN        NaN
veil-color   0.152836         0.046401      0.096730   0.645677   0.152836
ring-number  0.152836         0.046401      0.096730   0.645677   0.152836
ring-type    0.152836         0.046401      0.096730   0.645677   0.152836
spore-print-color  0.244756         0.038871      0.095772   0.039342   0.244756
population   0.152836         0.046401      0.096730   0.645677   0.152836
habitat      0.152836         0.046401      0.096730   0.645677   0.152836

                 stalk-shape  stalk-root  stalk-surface-above-ring  \
class        0.102571   0.346201   0.334011
cap-shape    0.064755   0.038871   0.054677
cap-surface  0.054151   0.095772   0.293643
cap-color    0.075195   0.039342   0.047226
bruises      0.075808   0.244756   0.332351
odor         0.075808   0.244756   0.332351
gill-attachment  0.064755   0.038871   0.054677
gill-spacing 0.054151   0.095772   0.293643
gill-size    0.075195   0.039342   0.047226
gill-color   0.075808   0.244756   0.332351
stalk-shape  1.000000   0.075808   0.054677
stalk-root   0.075808   1.000000   0.095772
stalk-surface-above-ring  0.054677   0.095772   1.000000
stalk-surface-below-ring  0.054307   0.095772   0.279755
stalk-color-above-ring  0.046401   0.095772   0.096730
stalk-color-below-ring  0.046401   0.095772   0.096730
veil-type    NaN        NaN        NaN
veil-color   0.046401   0.095772   0.096730
ring-number  0.046401   0.095772   0.096730
ring-type    0.046401   0.095772   0.096730
spore-print-color  0.038871   0.095772   0.095772
population   0.046401   0.095772   0.096730
habitat      0.046401   0.095772   0.096730

                 stalk-surface-below-ring  stalk-color-above-ring  \
class        0.304780   0.222799
cap-shape    0.054307   0.046401
cap-surface  0.279755   0.096730
cap-color    0.046401   0.645677
bruises      0.326966   0.152836
odor         0.326966   0.152836
gill-attachment  0.054307   0.046401
gill-spacing 0.279755   0.096730
gill-size    0.046401   0.645677
gill-color   0.326966   0.152836
stalk-shape  0.054307   0.046401
stalk-root   0.095772   0.095772
stalk-surface-above-ring  0.279755   0.096730
stalk-surface-below-ring  1.000000   0.096730
stalk-color-above-ring  0.096730   1.000000
stalk-color-below-ring  0.096730   1.000000
veil-type    NaN        NaN
veil-color   0.096730   0.645677
ring-number  0.096730   0.645677
ring-type    0.096730   0.645677
spore-print-color  0.095772   0.095772
population   0.096730   0.645677
habitat      0.096730   0.645677

                 stalk-color-below-ring  veil-type  veil-color  ring-number  \
class        0.218987        NaN   0.145142   0.219066
cap-shape    0.046401        NaN   0.046401   0.046401
cap-surface  0.096730        NaN   0.096730   0.096730
cap-color    0.645677        NaN   0.645677   0.645677
bruises      0.152836        NaN   0.152836   0.152836
odor         0.152836        NaN   0.152836   0.152836
gill-attachment  0.046401        NaN   0.046401   0.046401
gill-spacing 0.096730        NaN   0.096730   0.096730
gill-size    0.645677        NaN   0.645677   0.645677
gill-color   0.152836        NaN   0.152836   0.152836
stalk-shape  0.046401        NaN   0.046401   0.046401
stalk-root   0.095772        NaN   0.095772   0.095772
stalk-surface-above-ring  0.096730        NaN   0.096730   0.096730
stalk-surface-below-ring  0.096730        NaN   0.096730   0.096730
stalk-color-above-ring  1.000000        NaN   0.645677   0.645677
stalk-color-below-ring  1.000000        NaN   0.645677   0.645677
veil-type    NaN        NaN        NaN        NaN
veil-color   0.645677        NaN   1.000000   0.645677
ring-number  0.645677        NaN   0.645677   1.000000
ring-type    0.645677        NaN   0.645677   0.645677
spore-print-color  0.095772        NaN   0.096730   0.096730
population   0.645677        NaN   0.645677   0.645677
habitat      0.645677        NaN   0.645677   0.645677

                 ring-type  spore-print-color  population  habitat
class        0.411771   0.346201   0.298686   0.107283
cap-shape    0.046401   0.038871   0.046401   0.046401
cap-surface  0.096730   0.095772   0.096730   0.096730
cap-color    0.645677   0.039342   0.645677   0.645677
bruises      0.152836   0.244756   0.152836   0.152836
odor         0.152836   0.244756   0.152836   0.152836
gill-attachment  0.046401   0.038871   0.046401   0.046401
gill-spacing 0.096730   0.095772   0.096730   0.096730
gill-size    0.645677   0.039342   0.645677   0.645677
gill-color   0.152836   0.244756   0.152836   0.152836
stalk-shape  0.046401   0.038871   0.046401   0.046401
stalk-root   0.095772   0.095772   0.095772   0.095772
stalk-surface-above-ring  0.096730   0.095772   0.096730   0.096730
stalk-surface-below-ring  0.096730   0.095772   0.096730   0.096730
stalk-color-above-ring  0.645677   0.095772   0.645677   0.645677
stalk-color-below-ring  0.645677   0.095772   0.645677   0.645677
veil-type    NaN        NaN        NaN        NaN
veil-color   0.645677   0.096730   0.645677   0.645677
ring-number  0.645677   0.096730   0.645677   0.645677
ring-type    0.645677   0.095772   0.645677   0.645677
spore-print-color  0.095772   1.000000   0.095772   0.095772
population   0.645677   0.095772   1.000000   0.645677
habitat      0.645677   0.095772   0.645677   1.000000
```

Now we will examine correlations between columns using a heatmap.

```
In [ ]: # Correlation Matrix
        corr_matrix = df_encoded.corr()
        plt.figure(figure_size=(12,10))
        sns.heatmap(corr_matrix, annot=True, cmap='Pastel1', linewidths=0.5
        plt.title('Correlation Matrix', fontsize=20)
        plt.xlabel('', fontsize=20)
        plt.ylabel('', fontsize=20)
        plt.tight_layout()
```

It can be seen that there is a high correlation between the columns:

- veil-color & gill-attachment
- spore-print-color & stalk-root
- gill-size & stalk-root
- gill-size & class

We will look at the connections between the different column and the classification column

Cap Shape & Class

```
In [ ]: # Cap Shape & Class
        plt.figure(figure_size=(8,6))
        sns.countplot(x='cap-shape', hue='class', palette=['#1f77b4','#ff7f0e'],
                      edgecolor=
                      linewidth=
        plt.xlabel('Cap Shape', fontsize=20)
        plt.ylabel('Count of Mushrooms', fontsize=20)
        plt.title('Distribution of the Mushrooms by their Classes and Cap Shapes', fontsize=20)
        plt.legend(['Edible', 'Poisonous'], fontsize=20)
        plt.xticks([0, 1, 2, 3, 4, 5], ['Bell', 'Conical', 'Convex', 'Flat', 'Knobbed', 'Sunken'], fontsize=20)
        plt.tight_
```

According to the plot, the Convex Cap Shape and Flat Cap Shape have the most mushrooms and probably they are mostly edible. Bell Cap Shape and Sunken Cap Shape have the least mushrooms and probably they are mostly poisonous.

Cap Surface & Class

```
In [ ]: # Cap Surface & Class
        plt.figure(figure_size=(8,6))
        sns.countplot(x='cap-surface', hue='class', palette=['#1f77b4','#ff7f0e'],
                      edgecolor=
                      linewidth=
        plt.xlabel('Cap Surface', fontsize=20)
        plt.ylabel('Count of Mushrooms', fontsize=20)
        plt.title('Distribution of the Mushrooms by their Classes and Cap Surfaces', fontsize=20)
        plt.legend(['Edible', 'Poisonous'], fontsize=20)
        plt.xticks([0, 1, 2, 3], ['Fibrous', 'Grooved', 'Scaly', 'Smooth'], fontsize=20)
        plt.tight_
```

Cap Surface of the Mushrooms

As seen from the figure, the Scaly and Smooth types are high probability poisonous. In contrast, the Fibrous type is mostly edible.

Cap Color & Class

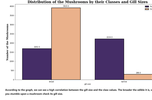
```
In [ ]: # Cap Color & Class
        plt.figure(figure_size=(8,6))
        sns.countplot(x='cap-color', hue='class', palette=['#1f77b4','#ff7f0e'],
                      edgecolor=
                      linewidth=
        plt.xlabel('Cap Color', fontsize=20)
        plt.ylabel('Count of Mushrooms', fontsize=20)
        plt.title('Distribution of the Mushrooms by their Classes and Cap Colors', fontsize=20)
        plt.legend(['Edible', 'Poisonous'], fontsize=20)
        plt.xticks([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], ['Brown', 'Buff', 'Cinnamon', 'Gray', 'Green', 'Pink', 'Purple', 'Red', 'White', 'Yellow'], fontsize=20)
        plt.tight_
```

Cap Color of the Mushrooms

According to the graph, the White and Poisonous class mushrooms are pretty close to each other. Yellow, Gray, and Red caps colors, The White cap color mushrooms are highly edible. Purple and Green caps colored mushrooms have no poisonous class.

Bruises & Class

```
In [ ]: # Bruises & Class
        plt.figure(figure_size=(8,6))
        sns.countplot(x='bruises', hue='class', palette=['#1f77b4','#ff7f0e'],
                      edgecolor=
                      linewidth=
        plt.xlabel('Mushrooms Without Bruises / Mushrooms With Bruises', fontsize=20)
        plt.ylabel('Count of Mushrooms', fontsize=20)
        plt.title('Distribution of the Mushrooms by Classes and Bruises', fontsize=20)
        plt.legend(['Edible', 'Poisonous'], fontsize=20)
        plt.xticks([0, 1], ['Mushrooms Without Bruises', 'Mushrooms With Bruises'], fontsize=20)
        plt.tight_
```



As we can see from the graph above, most of edible mushrooms have bruises, however, a lot of the poisonous mushrooms have no bruises.

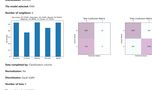
Odor & Class

```
In [ ]: # Odor & Class
        plt.figure(figure_size=(8,6))
        sns.countplot(x='odor', hue='class', palette=['#1f77b4','#ff7f0e'],
                      edgecolor=
                      linewidth=
        plt.xlabel('Odor', fontsize=20)
        plt.ylabel('Count of Mushrooms', fontsize=20)
        plt.title('Distribution of the Mushrooms by Classes and Odors', fontsize=20)
        plt.legend(['Edible', 'Poisonous'], fontsize=20)
        plt.xticks([0, 1, 2, 3, 4, 5, 6, 7, 8], ['Almond', 'Anise', 'Creosote', 'Fishy', 'Foul', 'Musty', 'Pungent', 'Spicy', 'None'], fontsize=20)
        plt.tight_
```

According to the graph above, mushrooms with no odor are highly safe for humans. But the mushrooms with the Foul, Fishy, and Spicy odor are highly Poisonous.

Gill Attachment & Class

```
In [ ]: # Gill Attachment & Class
        plt.figure(figure_size=(8,6))
        sns.countplot(x='gill-attachment', hue='class', palette=['#1f77b4','#ff7f0e'],
                      edgecolor=
                      linewidth=
        plt.xlabel('Gill Attachment', fontsize=20)
        plt.ylabel('Count of Mushrooms', fontsize=20)
        plt.title('Distribution of the Mushrooms by their Classes and Gill Attachments', fontsize=20)
        plt.legend(['Edible', 'Poisonous'], fontsize=20)
        plt.xticks([0, 1], ['Free', 'Attached'], fontsize=20)
        plt.tight_
```



According to the graph, the Free type gill attachments are approximately equally distributed. In contrast, the Attached type gill attachments are mostly poisonous.

Gill Size & Class

```
In [ ]: # Gill Size & Class
        plt.figure(figure_size=(8,6))
        sns.countplot(x='gill-size', hue='class', palette=['#1f77b4','#ff7f0e'],
                      edgecolor=
                      linewidth=
        plt.xlabel('Gill Size', fontsize=20)
        plt.ylabel('Count of Mushrooms', fontsize=20)
        plt.title('Distribution of the Mushrooms by their Classes and Gill Sizes', fontsize=20)
        plt.legend(['Edible', 'Poisonous'], fontsize=20)
        plt.xticks([0, 1], ['Broad', 'Narrow'], fontsize=20)
        plt.tight_
```

According to the graph, we can see a high correlation between the gill size and the class values. The broader the gill size is, so it you handle again a poisonous mushroom to get one.

Habitat & Class

```
In [ ]: # Habitat & Class
        plt.figure(figure_size=(8,6))
        sns.countplot(x='habitat', hue='class', palette=['#1f77b4','#ff7f0e'],
                      edgecolor=
                      linewidth=
        plt.xlabel('Habitat', fontsize=20)
        plt.ylabel('Count of Mushrooms', fontsize=20)
        plt.title('Distribution of the Mushrooms by their Classes and Habitat
        plt.legend(['Edible', 'Poisonous'], fontsize=20)
        plt.xticks([0, 1, 2, 3, 4, 5, 6], ['Grasses', 'Leaves', 'Paths', 'Urban', 'Wood', 'Waste'], fontsize=20)
        plt.tight_
```

Preparation of Test and Train data

The Final process is the separation of test and train data.

```
In [ ]: # Preparation of Test and Train data
        X = df_encoded.drop('class', axis=1)
        y = df_encoded['class']
        from sklearn.model_selection import train_test_split
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)
        print('X_train shape: {}'.format(X_train.shape))
        print('X_test shape: {}'.format(X_test.shape))
        print('y_train shape: {}'.format(y_train.shape))
        print('y_test shape: {}'.format(y_test.shape))
```

```
X_train shape: (6499, 22)
X_test shape: (1625, 22)
y_train shape: (6499,)
y_test shape: (1625,)
```

Training Experiments On The Models

In this section we will run various classification models on our data.

In each experiment we will create a different division of the training set and the test set and perform different operations on the data (discretization, normalization, binning) a different threshold for the decision tree, etc.; in addition we will run the classification algorithm.

The classification algorithms that we will use are:

- Naive Bayes Classification
- Decision Tree Classification
- KNN Classification
- Ensemble Classification

Evaluating The Models

In addition we will calculate measurements that the performance of the models. We will use the results in the notebook in order to make comparisons between them and to draw conclusions.

The different indices we will use:

Confusion Matrix

True Positive(TP): Describe that the data count belonging to the "Class 1" class is estimated to be "Class 1"

True Negative(TN): Describe that the data count belonging to the "Class 2" class is estimated to be "Class 2"

False Positive(FP): Describe that the data count belonging to the "Class 2" class is estimated to be "Class 1". (Type I Error)

Accuracy: Specifies the percentage of data correct estimates.

Precision: Specifies the proportion of correct estimates only in a case where the class "Class 1" is estimated.

F1 Score: Specifies the harmonic mean of Recall and Precision values.

Recall: The recall is obtained by the ability of the classifier to find all the positive samples.

Mapping: The percentage of the majority classification data out of all the values in the classification column.

First experiment:

Distribution of 80% train and 20% test

Data categorized by: Classification column

Normalization: Yes

Discretization: Without

The random selected: 919

Number of neighbors: 7

Data categorized by: Classification column

Normalization: Yes

Discretization: Without

The random selected: 2760

Number of neighbors: 7

Data categorized by: Classification column

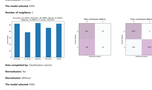
Normalization: Yes

Discretization: Equal width

Number of bins: 7

The random selected: 2760

Number of neighbors: 7



Data categorized by: Classification column

Normalization: Yes

Discretization: Equal depth

Number of bins: 7

The random selected: 2760

Number of neighbors: 7

Data scaling/norm: All data

Normalization: no

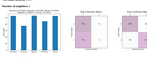
Discretization: Equal width

Number of Bins: 5

The model selected: Decision Tree

Max Depth(optimized): 3

Number of neighbors: 1



Data scaling/norm: All data

Normalization: no

Discretization: Without

The model selected: Decision Tree

Max Depth(optimized): 3

Data complexity: All data

Normalization: no

Discretization: Equal width

Number of Bins: 3

The model selected: KNN

Number of neighbors: 3

Data complexity: All data

Normalization: no

Discretization: Without

The model selected: Decision Tree

Max Depth(Hyperparameter): 3

Data complexity: All data

Normalization: no

Discretization: Without

The model selected: Decision Tree

Max Depth(Hyperparameter): 4

Data complexity: All data

Normalization: no

Discretization: Without

The model selected: Decision Tree

Max Depth(Hyperparameter): 5

Data complexity: All data

Normalization: no

Discretization: Without

The model selected: Decision Tree

Max Depth(Hyperparameter): 6

Data complexity: All data

Normalization: no

Discretization: Without

The model selected: Naive Bayes

Data complexity: All data

Normalization: no

Discretization: Equal frequency

The model selected: Naive Bayes

Data complexity: All data

Normalization: no

Discretization: Without

The model selected: Naive Bayes

Data complexity: All data

Normalization: no

Discretization: Without

The model selected: Naive Bayes

Data complexity: All data

Normalization: no

Discretization: Without

The model selected: Naive Bayes

Data complexity: All data

Normalization: no

Discretization: Without

The model selected: Naive Bayes

Data complexity: All data

Normalization: no

Discretization: Equal frequency

The model selected: Naive Bayes

Data complexity: All data

Normalization: no

Discretization: Equal frequency

Number of Bins: 3

The model selected: Decision Tree

Data complexity: All data

Normalization: no

Discretization: Without

Number of Bins: 3

The model selected: Naive Bayes

Conclusions:

1. Decision Bayes implementation gave better accuracy compared to the Sklearn Bayes implementation, we assume this is due to the fact that our implementation is not algorithmic. The best result was obtained by binning the data into KNN training and KNN testing.

2. We implemented all the Naive Bayes Bayes there were no significant differences between all data distributions.

3. These rate no change in any of the distance when we performed discretization, normalization and completed of missing values because all the columns are categorical and in addition, there were no missing values appearing in empty cells.

4. When running the various models, we often got better results by dividing the data by 50% training and 50% testing, this assume this is because the more information there is in data the more accurate the model will be.

5. In the implementation of the Decision Tree, as we increased the depth of the tree (threshold), the accuracy indicators gave a better result this is because we make the tree more specific and allow it to catch more information.

6. In KNN the best result is the 80% 20% distribution was in Sklearn, in the 80% 20% distribution there were 3 neighbors, in the 70% 30% distribution there were 5 neighbors.
as assume that this is because the data set contains noise and it is possible the algorithm found different nearest neighbors at each distribution

7. In addition the best result for 3 clusters was in 50% 50% split and as we increased the number of clusters the accuracy decreased. On the other hand, in the division of 80% 20% we increased the number of clusters the accuracy increased and in the division of 90% 10% the result remained the same.

Summary:

In this exercise, we used simple KNN methods to explore the multivariate features. While doing that, we learned pretty many things about implementing all new tools.

For the last step, we tested different machine learning models which are 2 Decision trees, 2 Naive Bayes, KNN, and Sklearn.

Due to the results of these machine learning models, we obtained different accuracy scores and more evaluation metrics scores for each implementation.

As part of a new exploration, we checked the size of the dataset we were working on. We is binning values, and performed various visualizations on most of the columns to deduce relevant information about the multivariate. In addition, we examined possible correlations between features and performed feature selection and our classification procedures. The conclusions we discovered along the way are detailed throughout the notebook.