University for the Creative Arts

BERLIN SCHOOL OF BUSINESS & INNOVATION

Essay / Assignment Title: Unveiling Mushroom Mysteries: A Python-Powered Machine Learning Journey

Programme title: Predictive Analytics and Machine Learning using Python

Name: GUDIMETLLA BHARAT NAGENDRA REDDY (Q1076626)

Year: 2024-2025

CONTENTS

Statement of compliance with academic ethics and the avoidance of plagiarism

I honestly declare that this dissertation is entirely my own work and none of its part has been copied from printed or electronic sources, translated from foreign sources and reproduced from essays of other researchers or students. Wherever I have been based on ideas or other people texts I clearly declare it through the good use of references following academic ethics.

(In the case that is proved that part of the essay does not constitute an original work, but a copy of an already published essay or from another source, the student will be expelled permanently from the postgraduate program).

Name and Surname (Capital letters):

GUDIMETLLA BHARAT NAGENDRA REDDY

Date: 2025/01/25

INTRODUCTION

Mushrooms are a significant type of fungus, offering a rich source of vitamin B and a higher protein content compared to most vegetables. They are known for their health benefits, including cancer prevention, aiding in weight loss, and boosting the immune system. However, some mushrooms are toxic and can be harmful if consumed. Thus, it is crucial to distinguish between edible and poisonous varieties to ensure safety (Maurya and Singh, 2020). The field of Predictive Analytics and Machine Learning offers powerful tools to derive insights and solve complex problems across various domains. Using a dataset containing characteristics of mushrooms classified as "edible" or "poisonous," this set-exercise integrates theoretical knowledge with practical implementation, enhancing both analytical and technical skills.

This project is structured into two sections. The first focuses on fundamental concepts such as entropy, information gain, and decision tree construction, enabling a deeper understanding of data processing and classification principles. The second section involves hands-on Python programming on Mushrooms-dataset for exploratory data analysis (EDA) and the application of multiple machine learning algorithms. By evaluating their performance metrics, we identify the most effective model for predicting Mushroom edibility.

SECTION ONE: Decision Tree and Information Gain Calculations (Hypothetical Mushroom Database)

Question 1

- **Calculate the probability/proportions:**
  - Edible mushrooms: 5 out of 10 samples (5/10 = 0.5)
  - Poisonous mushrooms: 5 out of 10 samples (5/10 = 0.5)
- **Calculate the entropy:**
  - Entropy (S) = - [(P(Edible) * log2(P(Edible)) + (P(Poisonous) * log2(P(Poisonous))]
  - Entropy (S) = - [0.5 * log2(0.5) + 0.5 * log2(0.5)]
  - Entropy (S) = - [0.5 * (-1) + 0.5 * (-1)]
  - Entropy (S) = 1

Therefore, the entropy of the data with respect to the classification is 1.

Question 2

Information gain is a measure that is used to quantify the effectiveness of features in splitting datasets in classes for Decision Trees. Information gain calculates reduction in uncertainty of target variables when features are known. Information gain helps in understanding how features contribute to making the right predictions in decision trees (Azhagusundari and Thanamani, 2013).

1. Calculate Entropy for each Cap-Color Subset:

- **Formula for Entropy:** H(S) = - Σ P(x) * log2(P(x))

  where: * H(S) is the entropy of the subset S * P(x) is the probability of class x in the subset. Entropy is 0 for pure subsets where the features are only found in one class. All Log2 in this case are ($\log_2$).

• **w:** 3 Edible, 0 Poisonous H(w) = - [(3/3) * log2(3/3) + 0/3 * log2(0/3)] = **0**
• **y:** 2 Edible, 3 Poisonous H(y) = - [(2/5) * log2(2/5) + (3/5) * log2(3/5)] H(y) = - [0.4 * (-1.32) + 0.6 * (-0.737)] = **0.971**
• **g:** 0 Edible, 1 Poisonous H(g) = - [(0/1) * log2(0/1) + 1/1 * log2(1/1)] = **0**
• **r:** 0 Edible, 1 Poisonous H(r) = - [(0/1) * log2(0/1) + 1/1 * log2(1/1)] = **0**

2. Calculate Weighted Average Entropy of Cap-Color:

- **Formula:** H(Cap-Color) = Σ (|S_i| / |S|) * H(S_i) where: * |S_i| is the number of samples in subset i * |S| is the total number of samples * H(S_i) is the entropy of subset i
- H(Cap-Color) = [(3/10) * H(w) + (5/10) * H(y) + (1/10) * H(g) + (1/10) * H(r)]
- H(Cap-Color) = [(3/10) * 0 + (5/10) * 0.971 + (1/10) * 0 + (1/10) *0]
- H(Cap-Color) = 0.4855

3. Calculate Information Gain:

- **Formula:** Information Gain = H(S) - H(Cap-Color) where: * H(S) is the overall entropy of the dataset (1 from Q1)
- Information Gain = 1.0 - 0.4855 = 0.5145

Therefore, the information gain for Cap-Color is approximately 0.5145.

Question 3

After splitting by Cap-Color, the remaining attributes are Cap-Shape, Odor and Habitat. Each subset formed by Cap-Color has a different entropy and requires calculating the information gain for each remaining attribute. The attribute with the highest information gain among the remaining ones will be selected. The next attribute selection may not be unique, as multiple

attributes could have similar information gains depending on the distribution of values in the subsets.

If Cap-Color is the first attribute selected, we would need to determine the next attribute for the branches resulting from the Cap-Color splits.

- **Cap-Color = w, g, r:** These 3 branches are pure, no further splitting is needed.
  - "w" with only Edible. "g" and "r" with only Poisonous
- **Cap-Color = y:**
  - We would calculate the Information Gain of all remaining attributes (Cap-Shape, Odor, Habitat) on this subset of data.
  - The attribute with the highest Information Gain would be selected as the next node.

1. **Entropy of the Subset (S_y)**

- Number of Edible samples: 2
- Number of Poisonous samples: 3
- p(Edible) = 2/5 = 0.4
- p(Poisonous) = 3/5 = 0.6
- Entropy(S_y) = -p(Edible) * log2(p(Edible)) - p(Poisonous) * log2(p(Poisonous))
- Entropy(S_y) = -(0.4 * log2(0.4)) - (0.6 * log2(0.6))
- Entropy(S_y) = 0.971

2. **Calculating Entropy and Information for each value of Cap-Shape in respect to Parent Entropy Cap-Color-category 'y':**

Entropy for all Cap-Shape values:

• **c:** 0 Edible, 2 Poisonous | E(S_c) = - [(0/2) * log2(0/2) + 2/2 * log2(2/2)] = **0**
• **f:** 2 Edible, 0 Poisonous | E(S_f) = - [(2/2) * log2(2/2) + (0/2) * log2(2/2)] = **0**
• **b:** 0 Edible, 1 Poisonous | E(S_b) = - [(0/1) * log2(0/1) + 1/1 * log2(1/1)] = **0**

Gain(S_y, Cap-Shape) = Entropy(S_y) - ∑ [(|Sv|/|S_y|) * Entropy(Sv)]

- Gain(S_y, Cap-Shape) = 0.971 - [(2/5) * E(S_c) + (2/5) * E(S_f) + (1/5) * E(S_b)]
- Gain(S_y, Cap-Shape) = 0.971 - [(2/5) * 0 + (2/5) * 0 + (1/5) * 0]
- Gain(S_y, Cap-Shape) = 0.971

3. **Calculating Entropy and Information for each value of Odor in respect to Parent Entropy Cap-Color-category 'y':**

   Entropy for all Odor values;

• **p:** 1 Edible, 1 Poisonous | E(S_p) = - [(1/2) * log2(1/2) + 1/2 * log2(1/2)] = **1**
• **n: 1** Edible, 1 Poisonous | E(S_n) = - [(1/2) * log2(1/2) + (1/2) * log2(1/2)] = **1**
• **s:** 0 Edible, 1 Poisonous | E(S_s) = - [(0/1) * log2(0/1) + 1/1 * log2(1/1)] = **0**
• **a:** 0 Edible, 0 Poisonous | E(S_a) = - [(0/0) * log2(0/0) + 0/0 * log2(0/0)] = **0**

   Gain(S_y, Odor) = Entropy(S_y) - ∑ [(|Sv|/|S_y|) * Entropy(Sv)]

   - Gain(S_y, Odor) = 0.971 - [(2/5) * E(S_p) + (2/5) * E(S_n) + (0/5) * E(S_a) + (1/5) * E(S_s)]
   - Gain(S_y, Odor) = 0.971 - [(2/5) * 1 + (2/5) * 1 + (0/5) * 0 + (1/5) * 0)]
   - Gain(S_y, Odor) = 0.171

4. **Calculating Entropy and Information for each value of Habitat in respect to Parent Entropy Cap-Color-category 'y':**

   Entropy for all Habitat values;

• **m:** 0 Edible, 2 Poisonous | E(S_m) = - [(0/2) * log2(0/2) + 2/2 * log2(2/2)] = **0**
• **u:** 2 Edible, 0 Poisonous | E(S_u) = - [(2/2) * log2(2/2) + (0/2) * log2(0/2)] = **0**
• **l:** 0 Edible, 1 Poisonous | E(S_l) = - [(0/1) * log2(0/1) + 1/1 * log2(1/1)] = **0**

   Gain(S_y, Habitat) = Entropy(S_y) - ∑ [(|Sv|/|S_y|) * Entropy(Sv)]

   - Gain(S_y, Habitat) = 0.971 - [(2/5) * E(S_m) + (2/5) * E(S_u) + (0/5) * E(S_l)]
   - Gain(S_y, Habitat) = 0.971 - [(2/5) * 0 + (2/5) * 0 + (0/5) * 0)]
   - Gain(S_y, Habitat) = 0.971 – 0 = 0.971

5. **Comparison of Information Gains:**

   - Gain(S_y, Cap-Shape) = 0.971
   - Gain(S_y, Odor) = 0.171
   - Gain(S_y, Habitat) = 0.971

Since both Cap-Shape and Habitat have the same maximum Information Gain, either one could be selected as the next attribute for this branch of the decision tree. This answer is not unique because both attributes have the same information gain.

Question 4

1. Calculate the proportion of each class:

- **p1 (Proportion of Edible):** 5 Edible / 10 total samples = 0.5
- **p2 (Proportion of Poisonous):** 5 Poisonous / 10 total samples = 0.5

2. Calculate the probability of misclassification for each class:

- **Misclassifying Edible:** 1 - p1 = 1 - 0.5 = 0.5
- **Misclassifying Poisonous:** 1 - p2 = 1 - 0.5 = 0.5

3. Calculate the expected misclassification rate:

- Expected Misclassification Rate = (p1 * Misclassifying Edible) + (p2 * Misclassifying Poisonous)
- Expected Misclassification Rate = (0.5 * 0.5) + (0.5 * 0.5) = 0.25 + 0.25 = 0.5

Therefore, the expected misclassification rate is 0.5 or 50%.

This means that if you randomly predict "edible" with a probability of 0.5 and "poisonous" with a probability of 0.5, you can expect to misclassify 50% of the mushrooms in the dataset.

SECTION TWO: Python Code: Exploratory Data Analysis and Classification Algorithms on Mushroom Dataset

Exploratory Data Analysis (EDA)

The analysis of the Mushrooms dataset involved an exploratory data analysis (EDA) to understand the distribution and relationships of the variables. The dataset contains 8,124 samples and 23 categorical features, representing attributes like cap shape, cap color, habitat, odor, and more. The target variable, **class**, identifies whether a mushroom is edible (e) or poisonous (p).

Summary Statistics

The dataset is complete, with no missing values in any columns. Summary statistics reveal that the dataset is balanced, with 4,208 edible and 3,916 poisonous mushrooms. Key categorical attributes include cap-shape, cap-surface, odor, and habitat, each with multiple unique values.

Distribution Analysis

1. **Class Distribution**: Visualization of the target variable shows a slightly higher count of edible mushrooms compared to poisonous.

2. **Cap Shape by Class**: Edible mushrooms are more prevalent in specific cap shapes, like convex (x) and flat (f), while other shapes are more evenly distributed across the classes.

3. **Cap Color**: Brown (n) and Grey (g) are the most common cap colors across all samples, while Purple (u) and Green (r) are less frequent.

4. **Habitat**: The majority of mushrooms are found in woods (d) and grassy areas (g), while urban (u), meadows (m) and waste (w) habitats are less common.

Correlation Analysis

Using one-hot encoding for categorical variables, a correlation heatmap was generated to identify relationships between features. Most attributes showed weak correlations, as expected in a dataset with categorical values. Strong patterns, however, were observed for odor, which is strongly linked to the class label, making it a significant predictor of edibility.

Insights

The visualizations and summary statistics highlight that attribute like odor, cap-shape, and habitat play significant roles in distinguishing between edible and poisonous mushrooms. Future work could involve building classification models leveraging these insights.

In [22]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
# Loading the Mushroom dataset

df = pd.read_csv("mushrooms.csv")

# Displaying the first five rows of the data
df.head()
```

Out[2]:

	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	population	habitat
0	p	x	s	n	t	p	f	c	n	k	...	s	w	w	p	w	o	p	k	s	u
1	e	x	s	y	t	a	f	c	b	k	...	s	w	w	p	w	o	p	n	n	g
2	e	b	s	w	t	l	f	c	b	n	...	s	w	w	p	w	o	p	n	n	m
3	p	x	y	w	t	p	f	c	n	n	...	s	w	w	p	w	o	p	k	s	u
4	e	x	s	g	f	n	f	w	b	k	...	s	w	w	p	w	o	e	n	a	g

5 rows × 23 columns

In [3]:

```
# Summary Statistics

df.describe()
```

Out[3]:

	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	p
count	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	...	4	9	9	1	4	3	5	9	
top	e	x	y	n	f	n	f	c	b	b	...	s	w	w	p	w	o	p	w	
freq	4208	3656	3244	2284	4748	3528	7914	6812	5612	1728	...	4936	4464	4384	8124	7924	7488	3968	2388	

4 rows × 23 columns

In [5]:

```
# checking based on our target variale 'class', whether the given dataset is 'balanced' or 'imbalanced'

print(df['class'].value_counts())

print("\n")

print(f"Total values: {df.shape[0]}")
```

```
class
e    4208
p    3916
Name: count, dtype: int64


Total values: 8124
```

Visualisation

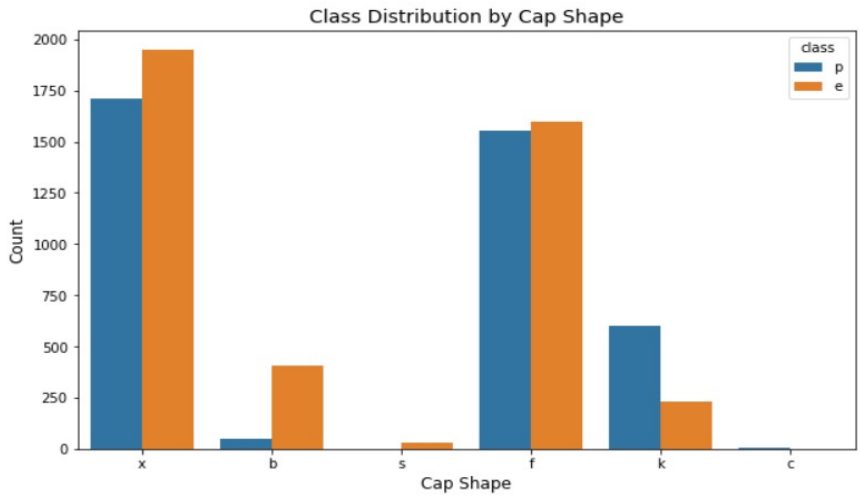
In [6]:

```
sns.countplot(x='class', data=df)
plt.title('Distribution of Edible vs Poisonous Mushrooms')
plt.show()
```

In [7]:

```
plt.figure(figsize=(10, 6))
sns.countplot(x='cap-shape', hue='class', data=df)
plt.title('Class Distribution by Cap Shape', fontsize=14)
plt.xlabel('Cap Shape', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.show()

print("Cap-shape: convex=x, bell=b, sunken=s, flat=f, knobbed=k, conical=c")
```

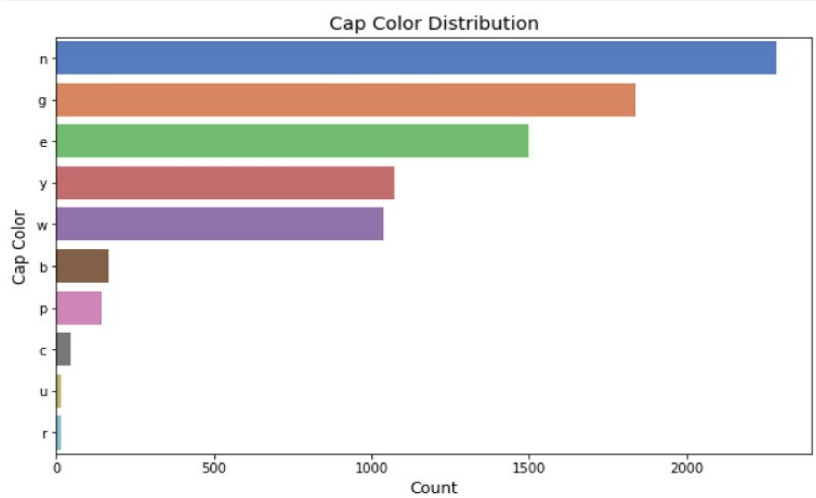



```
Cap-shape: convex=x, bell=b, sunken=s, flat=f, knobbed=k, conical=c
```

In [9]:

```
plt.figure(figsize=(10, 6))
sns.countplot(y='cap-color', data=df, order=df['cap-color'].value_counts().index, palette='muted')
plt.title('Cap Color Distribution', fontsize=14)
plt.xlabel('Count', fontsize=12)
plt.ylabel('Cap Color', fontsize=12)
plt.show()

print("Cap-color: brown=n, grey=g, red=e, yellow=y, white=w, buff=b, pink=p, cinnamon=c, purple=u, green=r")
```

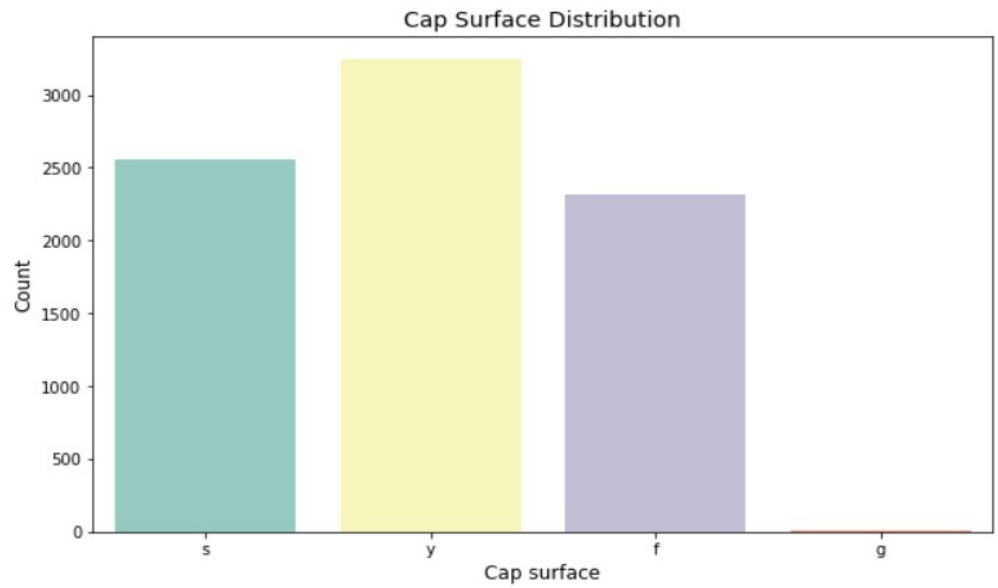



```
Cap-color: brown=n, grey=g, red=e, yellow=y, white=w, buff=b, pink=p, cinnamon=c, purple=u, green=r
```

In [10]:

```
plt.figure(figsize=(10, 6))
sns.countplot(x='cap-surface', data=df, palette='Set3')
plt.title('Cap Surface Distribution', fontsize=14)
plt.xlabel('Cap surface', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.show()

print("Cap-surface: smooth=s, scaly=y, fibrous=f, grooves=g")
```

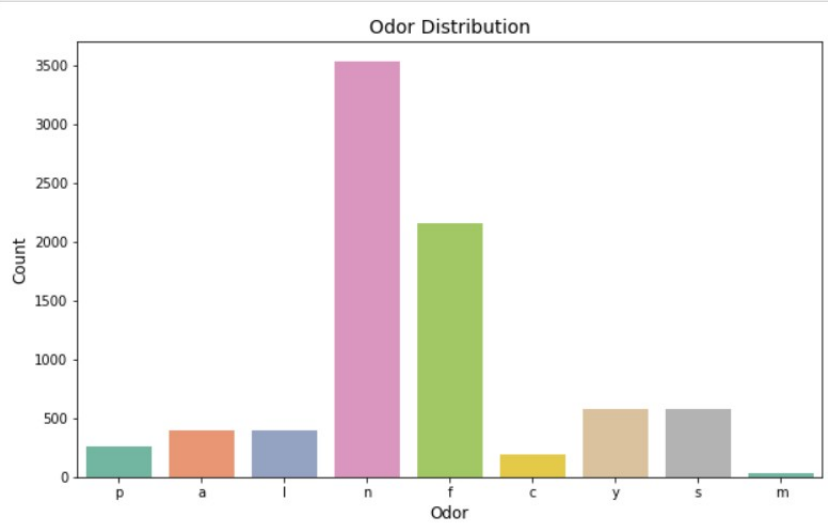



```
Cap-surface: smooth=s, scaly=y, fibrous=f, grooves=g
```

In [11]:

```
plt.figure(figsize=(10, 6))
sns.countplot(x='odor', data=df, palette='Set2')
plt.title('Odor Distribution', fontsize=14)
plt.xlabel('Odor', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.show()

print("Odor: pungent=p, almond=a, anise=l, none=n, foul=f, creosote=c, fishy=y, spicy=s, musty=m")
```

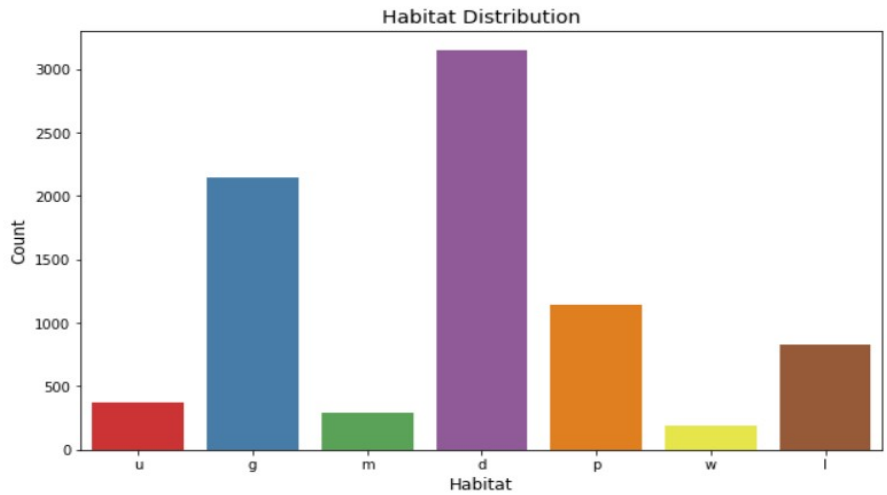



```
Odor: pungent=p, almond=a, anise=l, none=n, foul=f, creosote=c, fishy=y, spicy=s, musty=m
```

In [12]:

```
plt.figure(figsize=(10, 6))
sns.countplot(x='habitat', data=df, palette='Set1')
plt.title('Habitat Distribution', fontsize=14)
plt.xlabel('Habitat', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.show()

print("Habitat: urban=u, grasses=g, meadows=m, woods=d, paths=p, waste=w, leaves=l")
```




```
Habitat: urban=u, grasses=g, meadows=m, woods=d, paths=p, waste=w, leaves=l
```

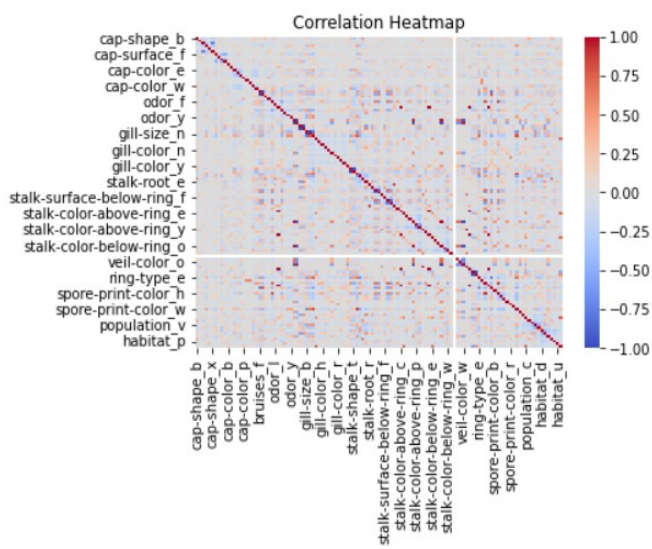
In [15]:

```
# Correlation heatmap

# in the 'encoded_df' we are only considering all the 'independent categorial variables'
# and droping our target variable:'class'.
# The 'independent categorical variables' are Nominal, hence, encoding them with pd.get_dummies

encoded_df = pd.get_dummies(df.drop('class', axis=1))
corr = encoded_df.corr()

sns.heatmap(corr, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



In [17]:

```
# Converting all the Categorical values into Numerical


from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
for column in df.columns:
    df[column] = le.fit_transform(df[column])
```

In [18]:

```
df.head()
```

Out[18]:

	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	populati
0	1	5	2	4	1	6	1	0	1	4	...	2	7	7	0	2	1	4	2	
1	0	5	2	9	1	0	1	0	0	4	...	2	7	7	0	2	1	4	3	
2	0	0	2	8	1	3	1	0	0	5	...	2	7	7	0	2	1	4	3	
3	1	5	3	8	1	6	1	0	1	5	...	2	7	7	0	2	1	4	2	
4	0	5	2	3	0	5	1	1	0	4	...	2	7	7	0	2	1	0	3	

5 rows × 23 columns

Machine Learning Algorithms

The task involved applying machine learning algorithms to predict whether a mushroom is edible or poisonous using the mushrooms dataset. Seven models were evaluated: Gaussian Naive Bayes, Logistic Regression, Decision Tree, Random Forest, Support Vector Classification (SVC), K-Nearest Neighbors (KNN), and XGBoost. Each model was assessed based on accuracy, precision, recall, and F1-score. Decision Tree, Random Forest, and XGBoost achieved perfect classification metrics (100% accuracy, precision, recall, and F1-score), demonstrating their ability to capture the dataset's feature relationships. XGBoost was particularly notable for its regularization and robustness. Other models Logistic Regression, Gaussian Naive Bayes, SVC and KNN performed well but slightly lagged behind.

Machine Learning Algorithms to predict whether a Mushroom is 'Edible' or 'Poisonous'

In [19]:

```
# Spliting the dataset into 'independent - X' and 'dependent - y'
# Then splitting these into 'training' and 'testing sets'


from sklearn.model_selection import train_test_split

# Split dataset
X = df.drop('class', axis=1)
y = df['class']
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

In [ ]:

```

```

Implementing the Machine Learning Algorithms

In [20]:

```
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from xgboost import XGBClassifier


from sklearn.metrics import classification_report, accuracy_score
```

Performance Evaluation

Initialising all the above models

Then, training, testing, evaluating with the Classification metrics

In [21]:

```
# Models to evaluate
models = {
    "Gaussian Naive Bayes": GaussianNB(),
    "Logistic Regression": LogisticRegression(),
    "Decision Tree": DecisionTreeClassifier(),
    "Random Forest": RandomForestClassifier(),
    "SVC": SVC(),
    "KNN": KNeighborsClassifier(),
    "XGBoost": XGBClassifier()
}

results = {}

# Evaluate each model
for name, model in models.items():
    model.fit(x_train, y_train)
    y_pred = model.predict(x_test)
    results[name] = classification_report(y_test, y_pred, output_dict=True)
    print(f"{name}:")
    print(classification_report(y_test, y_pred))
    print(f"Accuracy: {accuracy_score(y_test, y_pred):.2f}")


    print('\n')
```

```
Gaussian Naive Bayes:
              precision    recall  f1-score   support

           0       0.93      0.93      0.93      1257
           1       0.93      0.93      0.93      1181

    accuracy                           0.93      2438
   macro avg       0.93      0.93      0.93      2438
weighted avg       0.93      0.93      0.93      2438

Accuracy: 0.93


Logistic Regression:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95      1257
           1       0.95      0.94      0.95      1181

    accuracy                           0.95      2438
   macro avg       0.95      0.95      0.95      2438
weighted avg       0.95      0.95      0.95      2438

Accuracy: 0.95


Decision Tree:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1257
           1       1.00      1.00      1.00      1181

    accuracy                           1.00      2438
   macro avg       1.00      1.00      1.00      2438
weighted avg       1.00      1.00      1.00      2438

Accuracy: 1.00


Random Forest:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1257
           1       1.00      1.00      1.00      1181

    accuracy                           1.00      2438
   macro avg       1.00      1.00      1.00      2438
weighted avg       1.00      1.00      1.00      2438

Accuracy: 1.00
```

```
SVC:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      1257
           1       1.00      0.98      0.99      1181

    accuracy                           0.99      2438
   macro avg       0.99      0.99      0.99      2438
weighted avg       0.99      0.99      0.99      2438

Accuracy: 0.99
```

```
KNN:
              precision    recall  f1-score   support

           0       1.00      0.99      1.00      1257
           1       0.99      1.00      1.00      1181

    accuracy                           1.00      2438
   macro avg       1.00      1.00      1.00      2438
weighted avg       1.00      1.00      1.00      2438

Accuracy: 1.00
```

```
XGBoost:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1257
           1       1.00      1.00      1.00      1181

    accuracy                           1.00      2438
   macro avg       1.00      1.00      1.00      2438
weighted avg       1.00      1.00      1.00      2438

Accuracy: 1.00
```

Results and Model Comparison

The dataset was analyzed using seven machine learning algorithms to classify mushrooms as 'edible' or 'poisonous.' The models were evaluated based on accuracy, precision, recall, and F1-score. Below is a summary of the results for each algorithm:

1. **Gaussian Naive Bayes**

   - **Accuracy:** 93%
   - **Precision:** 93%
   - **Recall:** 93%
   - **F1-Score:** 93%
   - The model follows Bayes theorem to conduct probabilistic analysis (Ortega et al., 2020). Gaussian Naive Bayes performed well with consistent precision and recall across both classes. However, it underperformed compared to other algorithms due to its assumption of feature independence, which may not fully hold in this dataset.

2. **Logistic Regression**

   - **Accuracy:** 95%
   - **Precision:** 95%
   - **Recall:** 95%
   - **F1-Score:** 95%
   - Logistic Regression is a probabilistic model used for binary classification. It estimates the probability of a data point belonging to a particular class using a sigmoid function. It is a widely used and efficient algorithm for many classification tasks (Shah et al., 2020). Logistic Regression performed well, but slightly lagged compared to other models. Its linear nature limits its ability to capture complex, non-linear relationships in the dataset.

3. **Decision Tree**

- **Accuracy:** 100%
- **Precision:** 100%
- **Recall:** 100%
- **F1-Score:** 100%
- The model uses a tree-based classification and it uses information gain and entropy for training and learning (Ortega et al., 2020). The Decision Tree also achieved perfect results. However, as a single-tree model, it might be prone to overfitting on unseen data compared to Random Forest.

4. **Random Forest**

- **Accuracy:** 100%
- **Precision:** 100%
- **Recall:** 100%
- **F1-Score:** 100%
- The ensemble model uses the decision tree as the base learning algorithm (Pinky, Islam and Alice, 2019). Random Forest achieved perfect classification metrics, demonstrating its effectiveness in handling categorical features and capturing relationships within the dataset. It excelled due to its ensemble nature and ability to mitigate overfitting while still generalizing well.

5. **Support Vector Classification (SVC)**

- **Accuracy:** 98%
- **Precision:** 99%
- **Recall:** 99%
- **F1-Score:** 99%

- SVC aims to find the optimal hyperplane that best separates data points of different classes. It focuses on maximizing the margin between the hyperplane and the closest data points (support vectors). SVCs are known for their good generalization performance. SVC provided excellent results with near-perfect classification metrics. Its ability to create non-linear decision boundaries using the kernel trick makes it a strong performer.

6. **K-Nearest Neighbors (KNN)**

- **Accuracy:** 100%
- **Precision:** 100%
- **Recall:** 100%
- **F1-Score:** 100%
- KNN is a classification algorithm that predicts the class of a new data point based on the majority class of its k-nearest neighbors in the training data. It is simple to implement but can be computationally expensive for large datasets (Itoo, Meenakshi and Singh, 2021). KNN achieved perfect results by leveraging similarity-based classification. However, its performance is highly dependent on the choice of 'k' and the dataset size, which might lead to inefficiency for larger datasets.

7. **XGBoost**

- **Accuracy:** 100%
- **Precision:** 100%
- **Recall:** 100%
- **F1-Score:** 100%
- The XGBOOST is an improvement of the Gradient boosting algorithm which was created to have high predictive capabilities. XGBoost achieves high training efficiency using multi-core processing for tree construction and data structures designed to minimize data access time. This translates to faster model training and ultimately, better performance (Ramraj et al., 2016). XGBoost delivered flawless metrics. Its gradient-boosting framework effectively handles feature interactions,

leading to exceptional performance. Its regularization also helps prevent overfitting, making it robust.

Comparison of Models

The best-performing models were Decision Tree, Random Forest and XGBoost, all achieving perfect scores across all metrics. However, the Random Forest and XGBoost are preferred over Decision Tree due to their ensemble methods, which mitigate overfitting and enhance generalization. XGBoost stands out as the best model due to its ability to handle categorical data effectively, incorporate regularization, and optimize computational efficiency. It is highly robust and suitable for diverse datasets, making it an ideal choice for predicting mushroom edibility.

CONCLUDING REMARKS

In conclusion, the exploration and analysis of the mushroom's dataset provided valuable insights into the classification of mushrooms as edible or poisonous. Through Exploratory Data Analysis (EDA), we identified the distribution of various features, such as cap shape, cap color, odor, and habitat, and their relationships with the target variable. Visualizations, including count plots and correlation heatmaps, highlighted patterns within the dataset and reinforced the importance of feature encoding for machine learning tasks.

The manual entropy and information gain calculations demonstrated the importance of attributes like odor and cap color in distinguishing between edible and poisonous mushrooms. These results guided the decision tree construction, showcasing the utility of entropy-based measures in selecting optimal splits for classification.

Applying machine learning algorithms further validated these findings. Models like Decision Tree, Random Forest and XGBoost excelled with perfect accuracy, precision, recall, and F1-scores, indicating their effectiveness in handling the dataset's structure. Logistic Regression, SVC, and KNN also performed admirably but fell slightly short of the top performers. Gaussian Naive Bayes, while efficient, struggled due to its simplifying assumptions about feature independence. Overall, these tasks underscore the importance of rigorous data preprocessing, exploratory analysis, and model evaluation in achieving high classification performance for real-world datasets. These findings highlight the power of ensemble and tree-based models for predictive tasks.

BIBLIOGRAPHY

Azhagusundari, B. and Thanamani, A.S., 2013. Feature selection based on information gain. *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, *2*(2), pp.18-21.

Itoo, F., Meenakshi and Singh, S., 2021. Comparison and analysis of logistic regression, Naïve Bayes and KNN machine learning algorithms for credit card fraud detection. *International Journal of Information Technology*, *13*(4), pp.1503-1511.

Maurya, P. and Singh, N.P., 2020. Mushroom classification using feature-based machine learning approach. In *Proceedings of 3rd International Conference on Computer Vision and Image Processing: CVIP 2018, Volume 1* (pp. 197-206). Springer Singapore.

Ortega, J.H.J.C., Lagman, A.C., Natividad, L.R.Q., Bantug, E.T., Resureccion, M.R. and Manalo, L.O., 2020. Analysis of performance of classification algorithms in mushroom poisonous detection using confusion matrix analysis. *International Journal*, *9*(1.3).

Pinky, N.J., Islam, S.M. and Alice, R.S., 2019. Edibility detection of mushroom using ensemble methods. *International Journal of Image, Graphics and Signal Processing*, *Vol.11, No.4*, pp. 55-62.

Ramraj, S., Uzir, N., Sunil, R. and Banerjee, S., 2016. Experimenting XGBoost algorithm for prediction and classification of different datasets. *International Journal of Control Theory and Applications*, *9*(40), pp.651-662.

Shah, K., Patel, H., Sanghvi, D. and Shah, M., 2020. A comparative analysis of logistic regression, random forest and KNN models for the text classification. *Augmented Human Research*, *5*(1), p.12.

APPENDIX (if necessary)

Entropy Calculation, Information Gain & Decision Tree Learning | by Badiuzzaman Pranto | Analytics Vidhya | Medium

Understanding Information Gain in Decision Trees: A Complete Guide | by Amit Yadav | Biased-Algorithms | Medium