Mushroom Classification

January 7, 2025

1 Introduction

1.1 Let's explore Mushrooms and try to classify them as poisonous or edible.

First we will perform EDA and try to get to grips with the data. Then we will train multiple classification models and compare their performance. Here are the models we will utilize: - Random Forest - Naive Bayes - Logistic Regression

[58]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report, confusion_matrix, f1_score
from sklearn.metrics import accuracy_score, precision_score, recall_score
from sklearn.metrics import roc_curve, roc_auc_score
```

[59]:
```python
# Loading the Mushroom dataset from the UCI ML repository
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/mushroom/
 ↪agaricus-lepiota.data"
column_names = ['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']

data = pd.read_csv(url, names=column_names)
data.head()
```

[59]:
```
  class cap-shape cap-surface cap-color bruises odor gill-attachment  \
0     p         x           s         n       t    p               f
1     e         x           s         y       t    a               f
```

```
2     e         b           s         w       t    l               f
3     p         x           y         w       t    p               f
4     e         x           s         g       f    n               f

  gill-spacing gill-size gill-color  … stalk-surface-below-ring  \
0            c         n          k  …                        s
1            c         b          k  …                        s
2            c         b          n  …                        s
3            c         n          n  …                        s
4            w         b          k  …                        s

  stalk-color-above-ring stalk-color-below-ring veil-type veil-color  \
0                      w                      w         p          w
1                      w                      w         p          w
2                      w                      w         p          w
3                      w                      w         p          w
4                      w                      w         p          w

  ring-number ring-type spore-print-color population habitat
0           o         p                 k          s       u
1           o         p                 n          n       g
2           o         p                 n          n       m
3           o         p                 k          s       u
4           o         e                 n          a       g

[5 rows x 23 columns]
```

2 Lets do some EDA!

Here we will check for missing values, create a count plot to reveal if we have a balanced dataset, explore the feature distribution specifically: oder, cap-shape and gill-size.

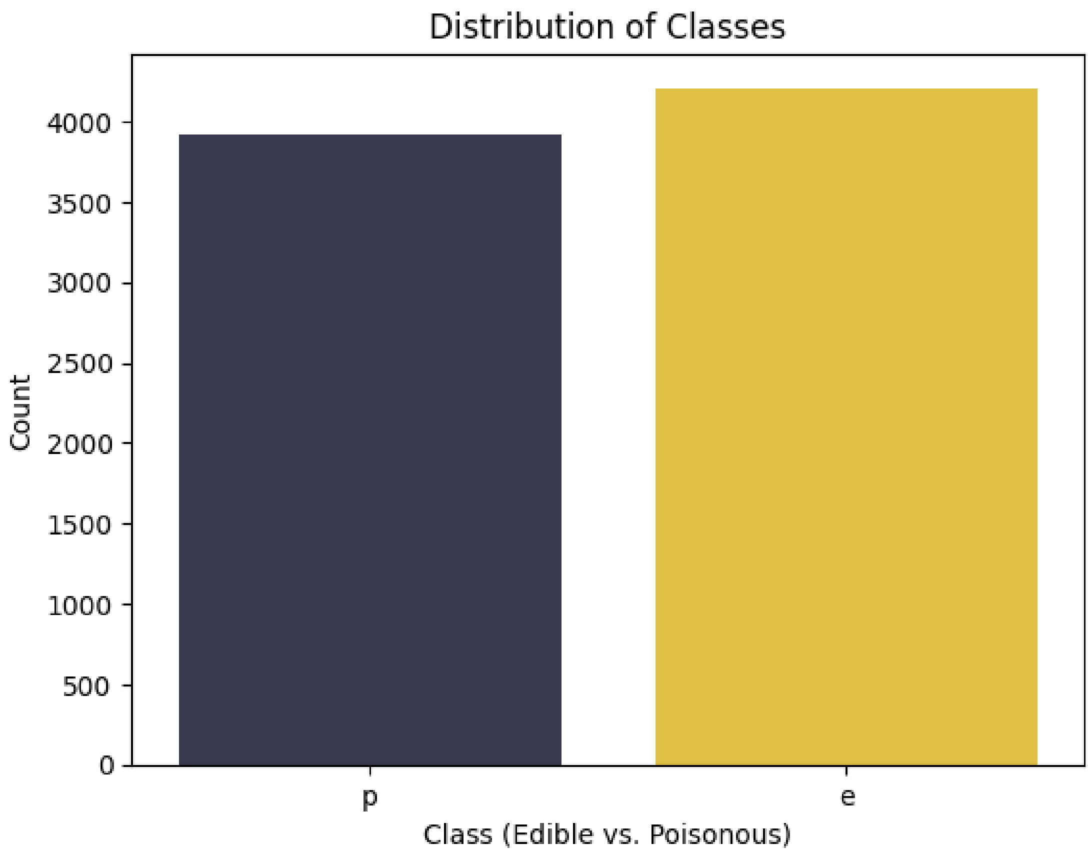
The dataset is entirely categorical, therefore we need to preprocess the data and convert categorical data into numerical values that machine learning models can process.

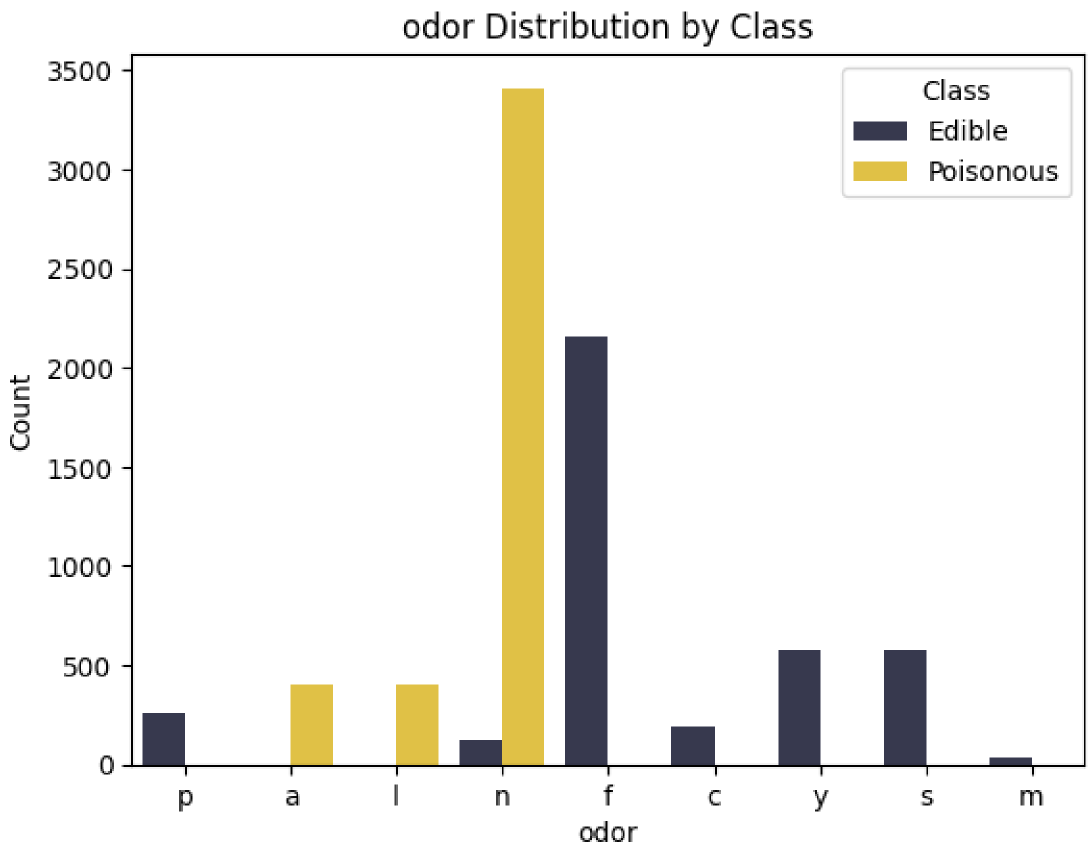
The visualizations (count plots) for features like cap-shape, odor, and gill-size reveal how these features vary across the target classes. These patterns provide a better understanding of the dataset's structure and help confirm which features are most influential.

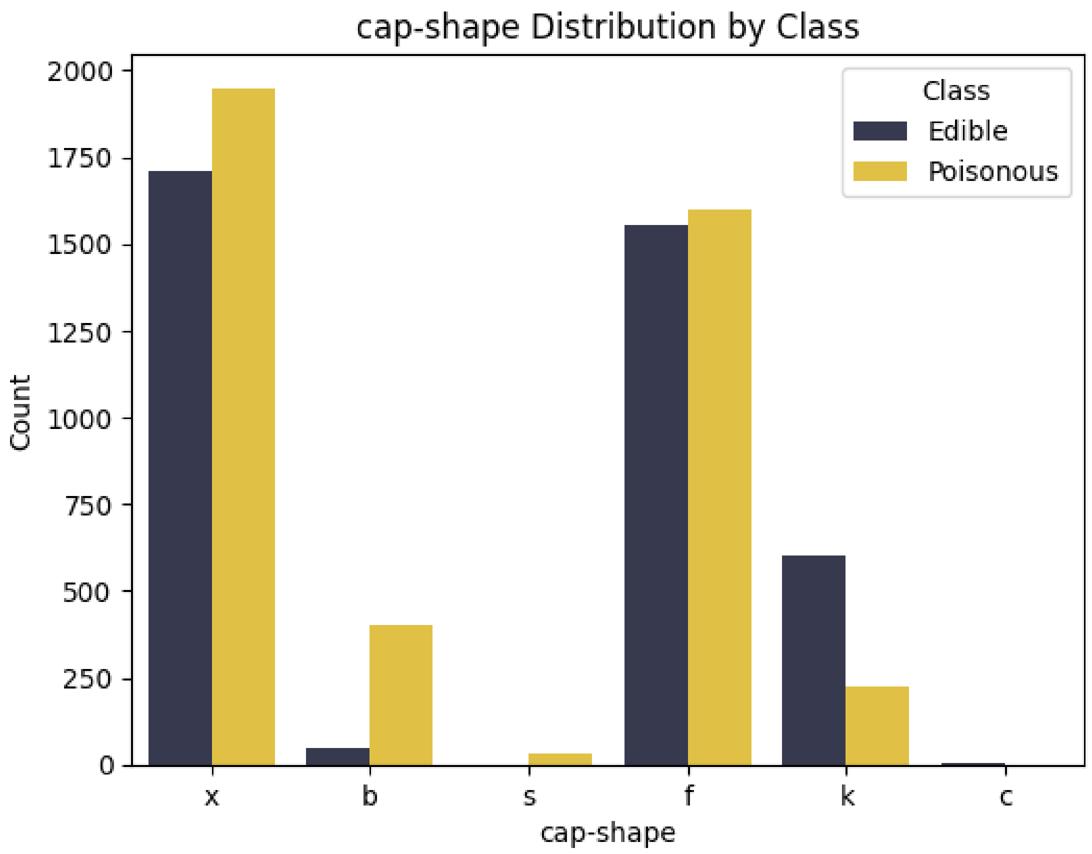
[60]:
```
# Basic information about the dataset
data.info()

# Check for missing values
print("Missing values per column:")
print(data.isnull().sum())

# Countplot for the class distribution
custom_palette = ['#333652', '#FAD02C']
```

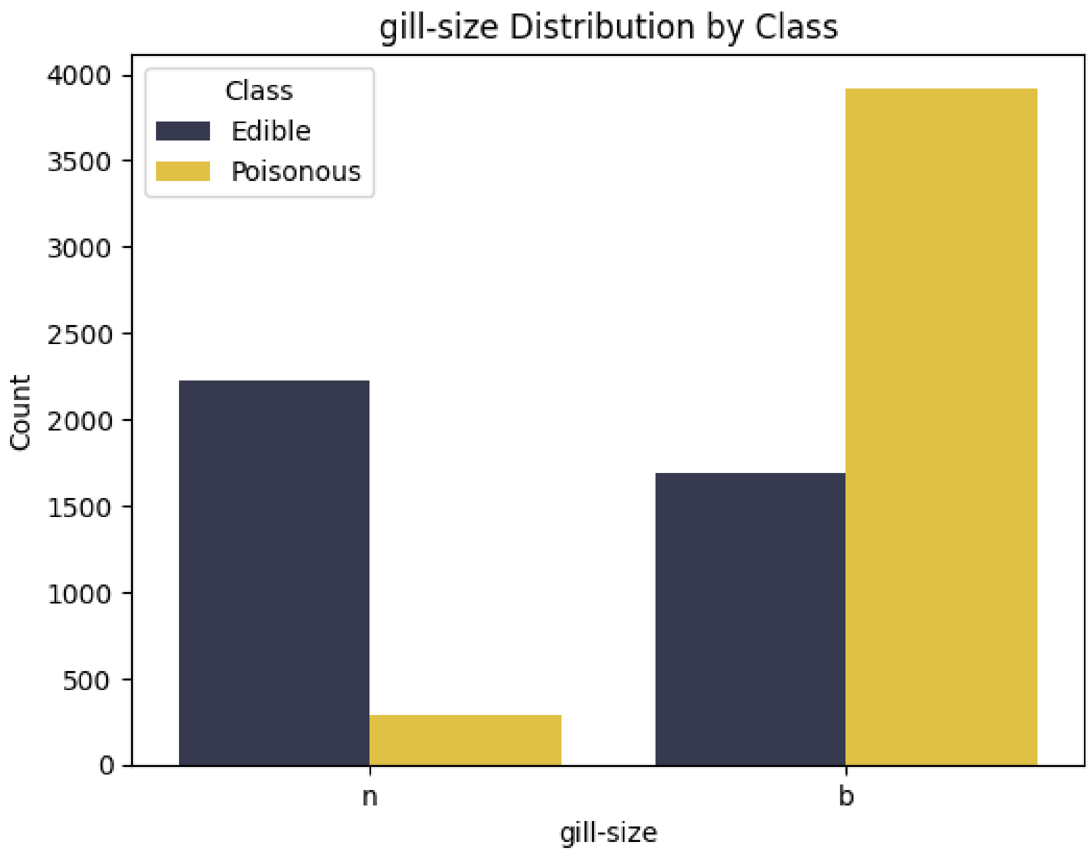
```python
sns.countplot(x='class', data=data, hue='class', palette=custom_palette,
              legend=False)
plt.title('Distribution of Classes')
plt.xlabel('Class (Edible vs. Poisonous)')
plt.ylabel('Count')
plt.show()

# Explore the feature distributions
data.describe(include='all')

features_to_plot = ['odor', 'cap-shape', 'gill-size']
for feature in features_to_plot:
    sns.countplot(x=feature, data=data, hue='class', palette=custom_palette,
                  legend=False)
    plt.title(f'{feature} Distribution by Class')
    plt.xlabel(feature)
    plt.ylabel('Count')
    plt.legend(title='Class', labels=['Edible', 'Poisonous'])
    plt.show()
```

```
<class 'pandas.core.frame.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
dtypes: object(23)
memory usage: 1.4+ MB
Missing values per column:
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: int64
```


Distribution of Classes
Count
0
500
1000
1500
2000
2500
3000
3500
4000
p
e
Class (Edible vs. Poisonous)


odor Distribution by Class
Count
odor
Class
Edible
Poisonous
p
a
l
n
f
c
y
s
m
0
500
1000
1500
2000
2500
3000
3500


cap-shape Distribution by Class
Count
2000
1750
1500
1250
1000
750
500
250
0
x
b
s
f
k
c
cap-shape
Class
Edible
Poisonous



[61]:
```python
data.replace('?', np.nan, inplace=True)
data.dropna(inplace=True)
encoder = LabelEncoder()
for column in data.columns:
    data[column] = encoder.fit_transform(data[column])

print(data.info())
print(data.head())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 5644 entries, 0 to 8114
Data columns (total 23 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   class                     5644 non-null   int64
 1   cap-shape                 5644 non-null   int64
 2   cap-surface               5644 non-null   int64
 3   cap-color                 5644 non-null   int64
 4   bruises                   5644 non-null   int64
 5   odor                      5644 non-null   int64
 6   gill-attachment           5644 non-null   int64
 7   gill-spacing              5644 non-null   int64
 8   gill-size                 5644 non-null   int64
 9   gill-color                5644 non-null   int64
 10  stalk-shape               5644 non-null   int64
 11  stalk-root                5644 non-null   int64
 12  stalk-surface-above-ring  5644 non-null   int64
 13  stalk-surface-below-ring  5644 non-null   int64
 14  stalk-color-above-ring    5644 non-null   int64
 15  stalk-color-below-ring    5644 non-null   int64
 16  veil-type                 5644 non-null   int64
 17  veil-color                5644 non-null   int64
 18  ring-number               5644 non-null   int64
 19  ring-type                 5644 non-null   int64
 20  spore-print-color         5644 non-null   int64
 21  population                5644 non-null   int64
 22  habitat                   5644 non-null   int64
dtypes: int64(23)
memory usage: 1.0 MB
None
   class  cap-shape  cap-surface  cap-color  bruises  odor  gill-attachment  \
0      1          5            2          4        1     6                1
1      0          5            2          7        1     0                1
2      0          0            2          6        1     3                1
3      1          5            3          6        1     6                1
4      0          5            2          3        0     5                1

   gill-spacing  gill-size  gill-color  …  stalk-surface-below-ring  \
0             0          1           2  …                         2
1             0          0           2  …                         2
2             0          0           3  …                         2
3             0          1           3  …                         2
4             1          0           2  …                         2

   stalk-color-above-ring  stalk-color-below-ring  veil-type  veil-color  \
0                       5                       5          0           0
1                       5                       5          0           0
2                       5                       5          0           0
3                       5                       5          0           0
4                       5                       5          0           0

   ring-number  ring-type  spore-print-color  population  habitat
0            1          3                  1           3        5
1            1          3                  2           2        1
2            1          3                  2           2        3
3            1          3                  1           3        5
4            1          0                  2           0        1
```

```
[5 rows x 23 columns]
```

3 Split and Train

Here we will split and train the data using 3 different ML models: - Random Forest - Naive Bayes - Logisitic Regression

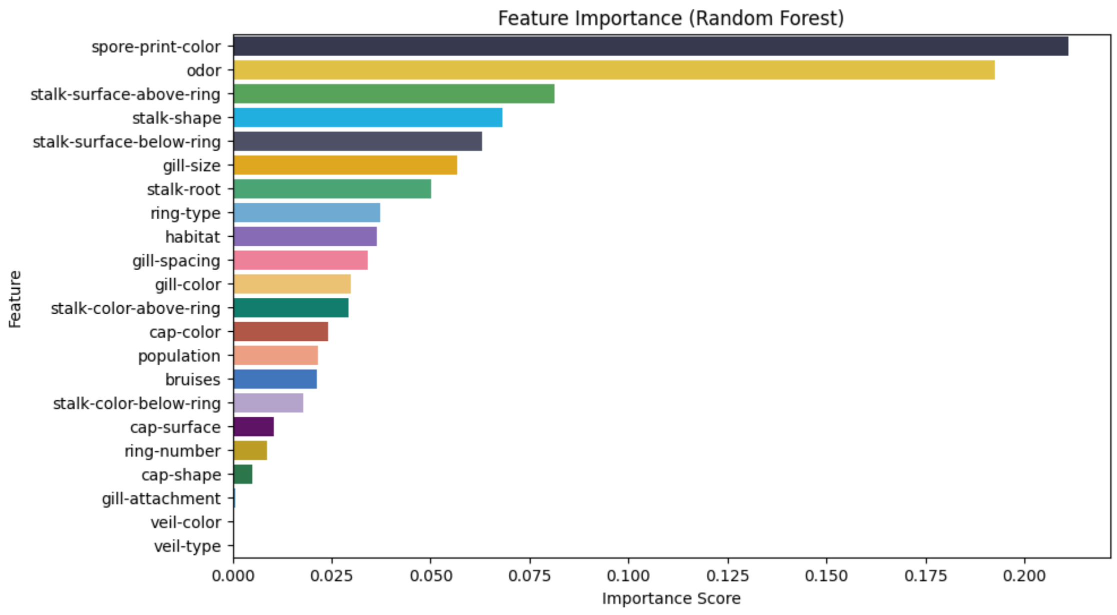
[62]:
```
X = data.drop('class', axis=1)
y = data['class']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
                                                    random_state=42)
```

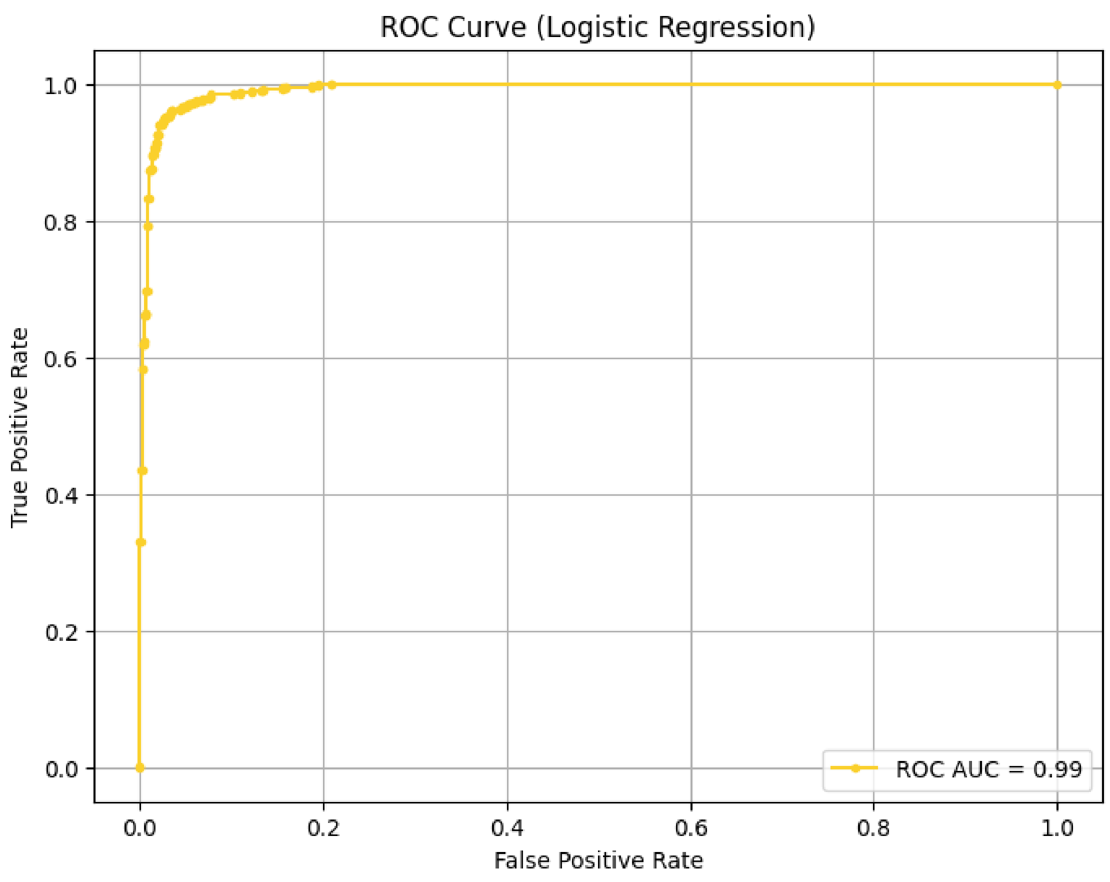
[63]:
```
models = {
    "Random Forest": RandomForestClassifier(),
    "Naive Bayes": GaussianNB(),
    "Logistic Regression": LogisticRegression(max_iter=1000)
}

for name, model in models.items():
    print(f"Training {name}...")
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    print(f"{name} Classification Report:")
    print(classification_report(y_test, y_pred))

    print(f"Confusion Matrix for {name}:")
    print(confusion_matrix(y_test, y_pred))
    print("-" * 50)

# Feature Importance for Random Forest
rf_model = models["Random Forest"]
feature_importances = pd.DataFrame({
    'Feature': X.columns,
    'Importance': rf_model.feature_importances_
}).sort_values(by='Importance', ascending=False)

custom_palette2 = ["#333652", "#FAD02C", "#4CAF50", "#00BFFF", "#4A4E69",
                   "#FFB400", "#3CB371", "#5DADE2", "#845EC2", "#FF6F91",
                   "#FFC75F", "#008F7A", "#C34A36", "#FF9671", "#2C73D2",
                   "#B39CD0", "#6A0572", "#D4AC0D", "#1E8449", "#3498DB",
                   "#E74C3C", "#F5B041"]
plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=feature_importances,
            hue = 'Feature', palette=custom_palette2, legend = False)
plt.title('Feature Importance (Random Forest)')
```

```python
plt.xlabel('Importance Score')
plt.ylabel('Feature')
plt.show()

# Predict probabilities for Logistic Regression
lr_model = models["Logistic Regression"]
y_pred_proba = lr_model.predict_proba(X_test)[:, 1]

fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
roc_auc = roc_auc_score(y_test, y_pred_proba)

plt.figure(figsize=(8, 6))
plt.plot(fpr,tpr,color='#FAD02C', marker='.', label=f'ROC AUC = {roc_auc:.2f}')
plt.title('ROC Curve (Logistic Regression)')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.grid()
plt.show()
```

```
Training Random Forest…
Random Forest Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1058
           1       1.00      1.00      1.00       636

    accuracy                           1.00      1694
   macro avg       1.00      1.00      1.00      1694
weighted avg       1.00      1.00      1.00      1694

Confusion Matrix for Random Forest:
[[1058    0]
 [   0  636]]
--------------------------------------------------
Training Naive Bayes…
Naive Bayes Classification Report:
              precision    recall  f1-score   support

           0       0.68      0.99      0.81      1058
           1       0.94      0.23      0.37       636

    accuracy                           0.70      1694
   macro avg       0.81      0.61      0.59      1694
weighted avg       0.78      0.70      0.64      1694

Confusion Matrix for Naive Bayes:
```

```
[[1049    9]
 [ 491  145]]
--------------------------------------------------
Training Logistic Regression…
Logistic Regression Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.98      0.97      1058
           1       0.97      0.93      0.95       636

    accuracy                           0.96      1694
   macro avg       0.96      0.96      0.96      1694
weighted avg       0.96      0.96      0.96      1694

Confusion Matrix for Logistic Regression:
[[1037   21]
 [  42  594]]
--------------------------------------------------
```

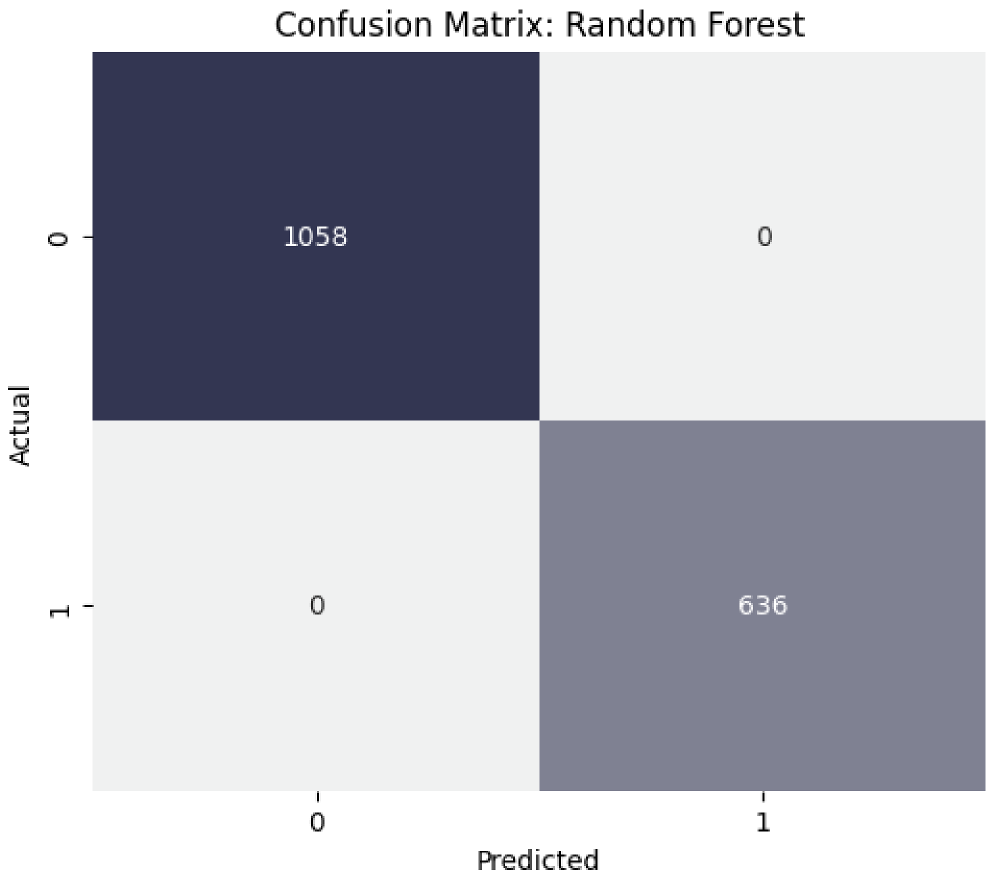


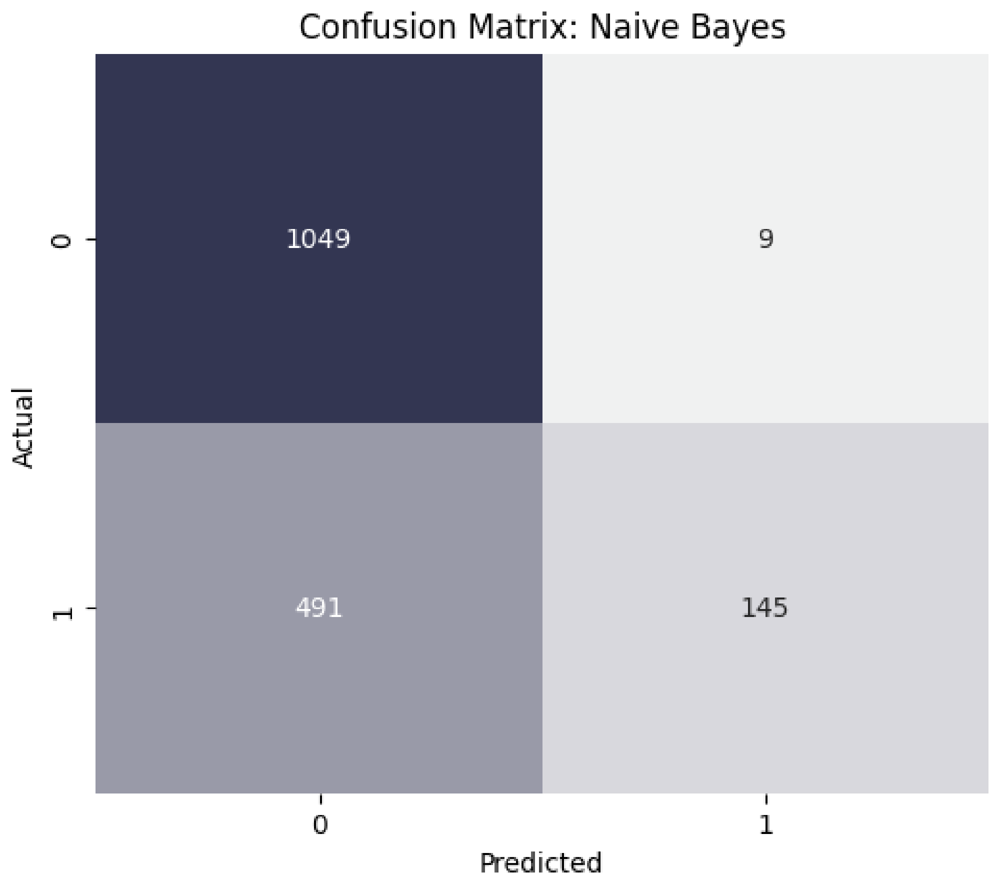


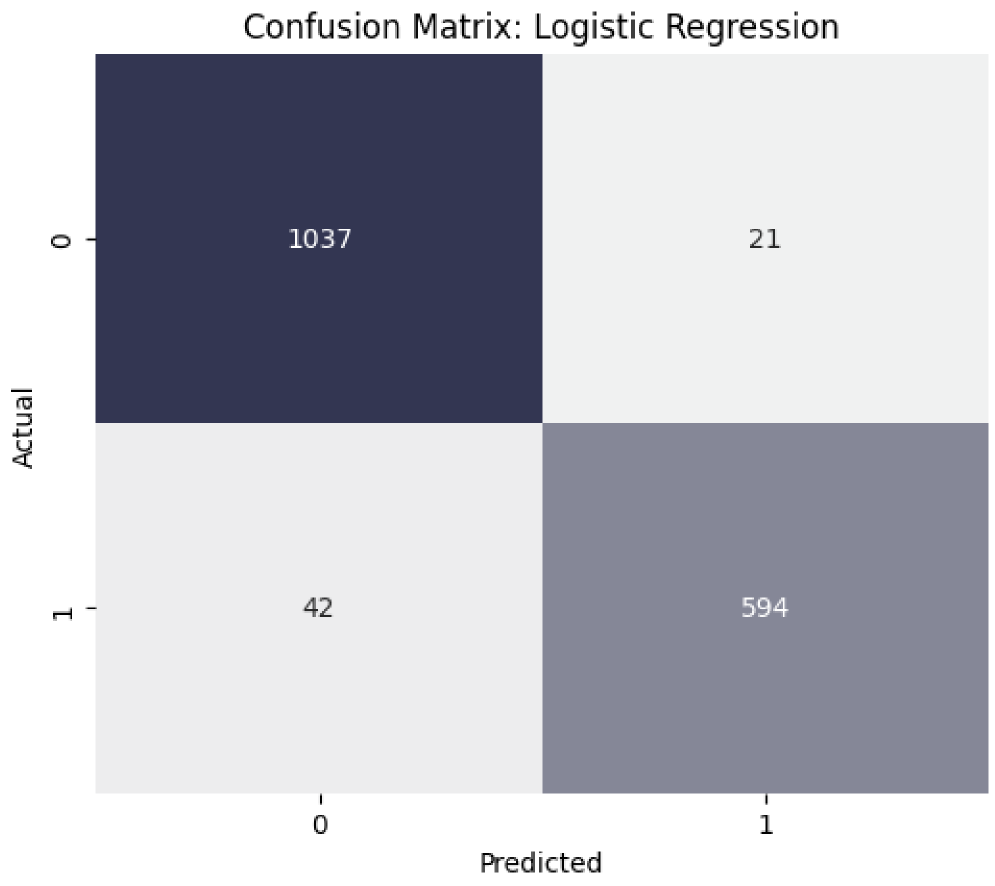
4 Time to Visualize!

In this section, we visualized the confusion matrices for each machine learning model to understand their performance in classifying mushrooms as edible or poisonous. Additionally, we compared the models using bar plots to evaluate their accuracy, precision, recall, and F1-score, highlighting the strengths and weaknesses of each algorithm.

[64]:
```
def plot_confusion_matrix(cm, model_name):
    plt.figure(figsize=(6, 5))
    sns.heatmap(cm, annot=True, fmt='d',
                cmap=sns.light_palette("#333652", as_cmap=True), cbar=False)
    plt.title(f"Confusion Matrix: {model_name}")
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
    plt.show()

for name, model in models.items():
    y_pred = model.predict(X_test)
    cm = confusion_matrix(y_test, y_pred)
    plot_confusion_matrix(cm, name)
```

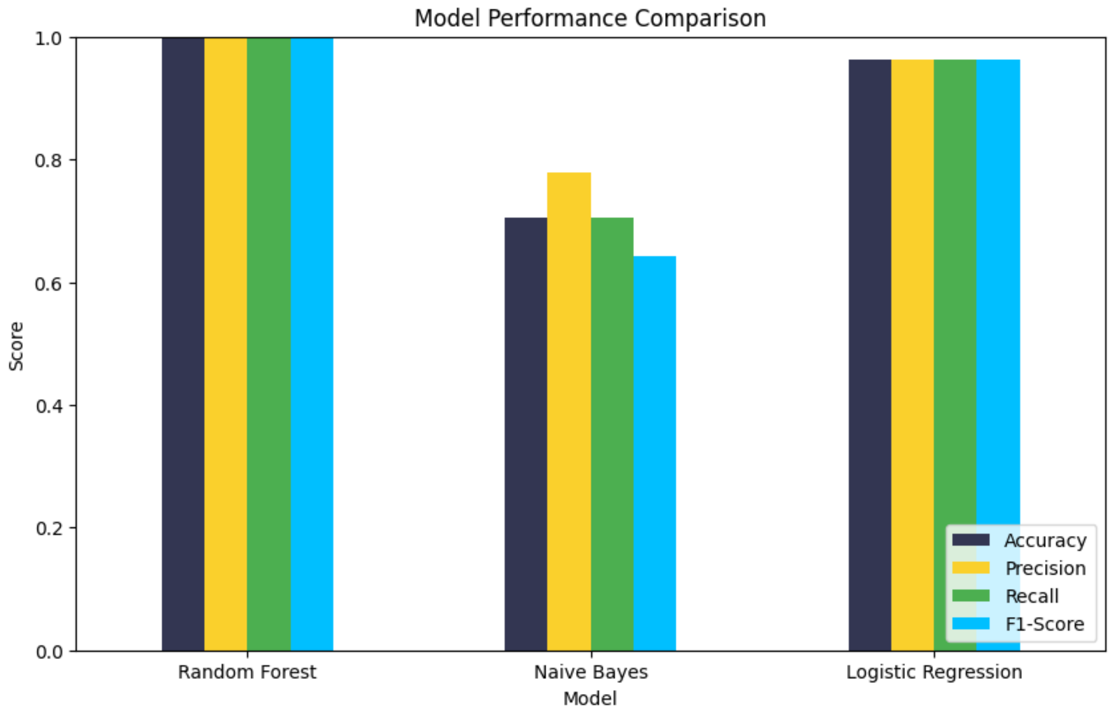



Confusion Matrix: Naive Bayes
1049
9
491
145
Actual
0
1
Predicted
0
1



[65]:
```
metrics = {
    "Model": [],
    "Accuracy": [],
    "Precision": [],
    "Recall": [],
    "F1-Score": []
}

for name, model in models.items():
    y_pred = model.predict(X_test)
    report = classification_report(y_test, y_pred, output_dict=True)

    metrics["Model"].append(name)
    metrics["Accuracy"].append(report["accuracy"])
    metrics["Precision"].append(report["weighted avg"]["precision"])
    metrics["Recall"].append(report["weighted avg"]["recall"])
    metrics["F1-Score"].append(report["weighted avg"]["f1-score"])

metrics_df = pd.DataFrame(metrics)
```

```
custom_palette3 = ['#333652', '#FAD02C', '#4CAF50', '#00BFFF']
metrics_df.set_index("Model").plot(kind="bar", figsize=(10, 6),
                                   color=custom_palette3)
plt.title("Model Performance Comparison")
plt.ylabel("Score")
plt.xticks(rotation=0)
plt.ylim(0, 1)
plt.legend(loc="lower right")
plt.show()
```



5 Conclusion

The Random Forest Classifier outperformed the other models with a perfect score across all evaluation metrics: accuracy, precision, recall, and F1-score, each achieving 100%. This demonstrates its exceptional ability to handle categorical data effectively and capitalize on its ensemble approach, which reduces variance and prevents overfitting. The results confirm that Random Forest is well-suited for this dataset, likely due to its capacity to manage complex interactions between features and its robustness in classification tasks. In comparison, while Logistic Regression and Naive Bayes also performed well, they did not achieve the same level of precision and recall, highlighting Random Forest's advantage in leveraging feature importance to enhance classification accuracy.