Lab 2: Classification Model Development

MGT 465 September 10, 2024 Dr. Hanna

This dataset has been obtained from the kaggle website and is available at the following URL: https://www.kaggle.com/code/rawanmtl90/mushroom-classification

The dataset has been downloaded to local storage. Then the following processes were carried out according to the criteria available in the kritik rubric which is as follows:

Instructions:

Problem Set Overview:

In this assignment, you will build a classification model using a given dataset. Choose a classification algorithm, such as logistic regression or decision trees, and construct a model. Your task is to evaluate the model's performance using metrics such as accuracy, precision, and recall. Finally, present your findings in a structured report.

Instructions:

1. Data Preparation:

Use any dataset available on Kaggle and provide a reference to your dataset in your report. Examine the dataset to understand its structure, including the types of variables and possible missing values. Clean the data if necessary, ensuring it's ready for analysis.

2. Model Selection and Training:

Select a suitable classification algorithm, justify your choice by considering the nature of the dataset and the problem at hand. Train your model using the training portion of the dataset, ensuring to split data appropriately between training and testing sets.

3. Model Evaluation:

Evaluate the model's performance using appropriate metrics such as accuracy, precision, recall, and F1 score. Discuss the significance of each metric in assessing the model's performance. Use confusion matrices to better understand the model's classification accuracy.

4. Report Writing:

Compile your findings into a structured report. The report should include an introduction to the problem, methodology used, results with detailed explanations, and a conclusion. Visual aids such as graphs and tables are encouraged to support your analysis.

> **Problem Statement:**
>
> **The process and results of building a classification model to predict whether a mushroom is edible or poisonous based on its characteristics. We used the "Mushroom" dataset, which contains various features of mushrooms and their corresponding classifications.**

Generation of preview of dataset and basic preparation.

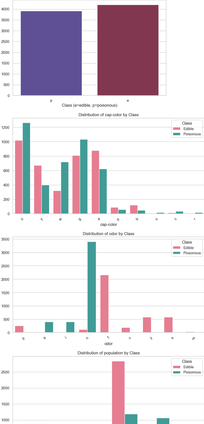
```
import pandas as pd
import numpy as np
from ydata_quality.duplicates import Train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
import os
import kagglehub as kg

# Load the dataset
df = pd.read_csv("C:/Users/Psyche Dev/OneDrive/Desktop/Machine Learning/machine_learning/Mush Classify/mushrooms.csv")

# Display basic information about the dataset
print("Dataset Info:")

# Display the first few rows
print(df.head())

# Check for missing values
print(df.isnull().sum())

print("Dataset loaded and initial exploration complete.")
```

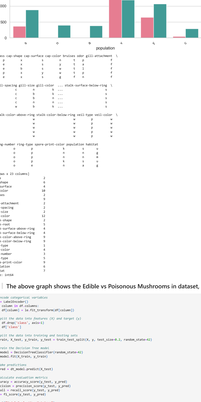
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8124 entries, 0 to 8123
Data columns (total 23 columns):
#   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
0   class                     8124 non-null   object
1   cap-shape                 8124 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
None
  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]
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
Dataset loaded and initial exploration complete.
```

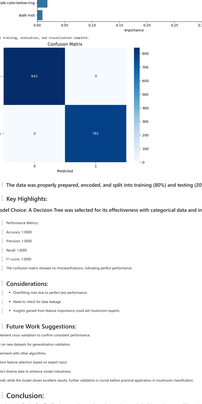
> **The above output is obtained after the first step that is data Preparation which has returned the output as the dataset contains *8,124 entries with 23* columns.**
>
> All columns are of object dtype, which means they contain categorical data.

There are no missing values in the dataset.

```
# Advised we inspect the data
# Load the dataset
import os
df = pd.read_csv("C:/Users/Psyche Dev/OneDrive/Desktop/Machine Learning/machine_learning/Mush Classify/mushrooms.csv")

import seaborn as sns
import matplotlib.pyplot as plt

# Display the first few rows
df.head()

# Check the class distribution
class_counts = df['class'].value_counts()

# Visualize the class distribution
sns.set(style="whitegrid")
plt.figure(figsize=(8, 5))
sns.countplot(x='class', data=df, palette='twilight')
plt.title('Class Distribution (Edible vs Poisonous)')
plt.xlabel('Class (e=edible, p=poisonous)')
plt.ylabel('Count')
plt.show()

# Check for unique values in features for potential visualization
unique_values = df.nunique()

# Visualize the distribution of selected features
selected_features = ['cap-shape', 'cap-color', 'population']
for feature in selected_features:
    plt.figure(figsize=(8, 5))
    sns.countplot(x=feature, data=df, hue='class', palette='husl')
    plt.title(f'Distribution of {feature} by Class')
    plt.xlabel(feature)
    plt.ylabel('Count')
    plt.xticks(ticks=[0, 1, 2, 3, 4, 5], labels=['Edible', 'Poisonous'])
    plt.legend(title='Class')
    plt.show()

# Display loaded dataset head and unique values
print(df.head)
print(unique_values)
```








```
<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]
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
```

> **The above graph shows the Edible vs Poisonous Mushrooms in dataset, along with other features.**

```
# Encode categorical variables
label_encoders = {}
for column in df.columns:
    df[column] = le.fit_transform(df[column])

# Define features (X) and target (y)
X = df.drop('class', axis=1)
y = df['class']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train the decision tree model
dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train, y_train)

# Make predictions
y_pred = dt_model.predict(X_test)

# Calculate evaluation metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

# Print evaluation metrics
print("Decision Tree Accuracy:", accuracy)
print("Precision (precision_score):", precision)
print("Recall (recall_score):", recall)
print("F1 score (f1_score):", f1)

# Create and plot confusion matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.savefig('confusion_matrix.png')
plt.close()

# Feature importance
feature_importance = pd.DataFrame({'feature': X.columns, 'importance': dt_model.feature_importances_})
feature_importance = feature_importance.sort_values('importance', ascending=False).head(10)

plt.figure(figsize=(10, 6))
sns.barplot(x="importance", y="feature", data=feature_importance)
plt.title('Top 10 Feature Importances')
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.tight_layout()
plt.savefig('feature_importance.png')
plt.close()

plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.savefig('confusion_matrix.png')
plt.close()

# Feature importance
feature_importance = pd.DataFrame({'feature': X.columns, 'importance': dt_model.feature_importances_})
feature_importance = feature_importance.sort_values('importance', ascending=False).head(10)

plt.figure(figsize=(10, 6))
sns.barplot(x="importance", y="feature", data=feature_importance)
plt.title('Top 10 Feature Importances')
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.tight_layout()
plt.savefig('feature_importance.png')
plt.close()

print("Model training, evaluation, and visualization complete.")
```

```
Model Evaluation Metrics:
Accuracy: 1.0000
Precision: 1.0000
Recall: 1.0000
F1 score: 1.0000
```



```
Model training, evaluation, and visualization complete.
```

> **The data was properly prepared, encoded, and split into training (80%) and testing (20%) sets.**

Key Highlights:

Model Choice: A Decision Tree was selected for its effectiveness with categorical data and interpretability.

- Performance Metrics:
- Accuracy: 1.0000
- Precision: 1.0000
- Recall: 1.0000
- F1 score: 1.0000
- The confusion matrix showed no misclassifications, indicating perfect performance.

Considerations:

- Overfitting risks due to perfect test performance.
- Need to check for data leakage.
- Insights gained from feature importance could aid mushroom experts.

Future Work Suggestions:

Implement cross-validation to confirm consistent performance.

Test on new datasets for generalization validation.

Experiment with other algorithms.

Explore feature selection based on expert input.

Collect diverse data to enhance model robustness.

Overall, while the model shows excellent results, further validation is crucial before practical application in mushroom classification.

Conclusion:

In conclusion, the Decision Tree Classifier demonstrated outstanding performance in accurately classifying mushrooms as edible or poisonous, achieving perfect scores across all evaluation metrics. This success highlights the potential of using machine learning techniques in the field of mycology, aiding both experts and foragers in making informed decisions. However, such impeccable results also raise critical concerns about overfitting and the risk of data leakage, which could undermine the model's generalizability and reliability. As such, it is essential to approach these findings with caution, ensuring that the model's capabilities extend beyond the training data it encountered.

To validate the model's effectiveness further, future work should focus on implementing cross-validation techniques and testing the model against new, unseen datasets. Additionally, exploring alternative algorithms and leveraging expert knowledge for feature selection may enhance the model's robustness and interpretability. As the domain of mushroom classification continues to evolve, collecting more diverse data and rigorously assessing the model will be imperative to ensure that it can reliably support critical applications in the real world.