Source code:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import os
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.tree import export_graphviz
import graphviz
print(os.listdir("C:/Users/Kanchi/PycharmProjects/Mushroom-Classification"))
df = pd.read_csv("mushrooms.csv")
df.head()
df.info()
df.describe()
print("Dataset shape:", df.shape)
df['class'].value_counts()
df["class"].unique()
count = df['class'].value_counts()
plt.figure(figsize=(8,7))
sns.barplot(count.index, count.values, alpha=0.8, palette="prism")
plt.ylabel('Count', fontsize=12)
plt.xlabel('Class', fontsize=12)
plt.title('Number of poisonous/edible mushrooms')
#plt.savefig("mushrooms1.png", format='png', dpi=900)
plt.show()
```

```
df = df.astype('category')

df.dtypes

labelencoder=LabelEncoder()

for column in df.columns:

    df[column] = labelencoder.fit_transform(df[column])

df.head()

df['veil-type']

df=df.drop(["veil-type"],axis=1)

df_div = pd.melt(df, "class", var_name="Characteristics")

fig, ax = plt.subplots(figsize=(16,6))

p = sns.violinplot(ax = ax, x="Characteristics", y="value", hue="class", split = True, data=df_div, inner = 'quartile', palette = 'Set1')

df_no_class = df.drop(["class"],axis = 1)

p.set_xticklabels(rotation = 90, labels = list(df_no_class.columns));

#plt.savefig("violinplot.png", format='png', dpi=900, bbox_inches='tight')

plt.figure(figsize=(14,12))

sns.heatmap(df.corr(),linewidths=.1,cmap="Purples", annot=True, annot_kws={"size": 7})

plt.yticks(rotation=0);

#plt.savefig("corr.png", format='png', dpi=900, bbox_inches='tight')

df[['class', 'gill-color']].groupby(['gill-color'], as_index=False).mean().sort_values

new_var = df[['class', 'gill-color']]

new_var = new_var[new_var['gill-color']<=3.5]

sns.factorplot('class', col='gill-color', data=new_var, kind='count', size=4.5, aspect=.8, col_wrap=4);

#plt.savefig("gillcolor1.png", format='png', dpi=900, bbox_inches='tight')

new_var=df[['class', 'gill-color']]

new_var=new_var[new_var['gill-color']>3.5]
```

```
sns.factorplot('class', col='gill-color', data=new_var, kind='count', size=4.5, aspect=.8, col_wrap=4);

#plt.savefig("gillcolor2.png", format='png', dpi=900, bbox_inches='tight')

X = df.drop(['class'], axis=1)

y = df["class"]

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42, test_size=0.1)

from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier()

dt.fit(X_train, y_train)

os.environ["PATH"] += os.pathsep + 'C:/Program Files (x86)/Graphviz2.38/bin/'

dot_data = export_graphviz(dt, out_file=None,

                           feature_names=X.columns,

                           filled=True, rounded=True,

                           special_characters=True)

graph = graphviz.Source(dot_data)

#graph.render(filename='DecisionTree')

graph

features_list = X.columns.values

feature_importance = dt.feature_importances_

sorted_idx = np.argsort(feature_importance)

plt.figure(figsize=(8,7))

plt.barh(range(len(sorted_idx)), feature_importance[sorted_idx], align='center', color ="red")

plt.yticks(range(len(sorted_idx)), features_list[sorted_idx])

plt.xlabel('Importance')

plt.title('Feature importance')

plt.draw()
```

```
#plt.savefig("featureimp.png", format='png', dpi=900, bbox_inches='tight')
plt.show()
y_pred_dt = dt.predict(X_test)
print("Decision Tree Classifier report: \n\n", classification_report(y_test, y_pred_dt))
print("Test Accuracy: {}%".format(round(dt.score(X_test, y_test)*100, 2)))
y_pred_svm = svm.predict(X_test)
print("SVM Classifier report: \n\n", classification_report(y_test, y_pred_svm))
cm = confusion_matrix(y_test, y_pred_svm)

x_axis_labels = ["Edible", "Poisonous"]
y_axis_labels = ["Edible", "Poisonous"]

f, ax = plt.subplots(figsize =(7,7))
sns.heatmap(cm, annot = True, linewidths=0.2, linecolor="black", fmt = ".0f", ax=ax, cmap="Purples", xticklabels=x_axis_labels, yticklabels=y_axis_labels)
plt.xlabel("PREDICTED LABEL")
plt.ylabel("TRUE LABEL")
plt.title('Confusion Matrix for SVM Classifier')
#plt.savefig("svmcm.png", format='png', dpi=900, bbox_inches='tight')
plt.show()
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb.fit(X_train, y_train)
print("Test Accuracy: {}%".format(round(nb.score(X_test, y_test)*100, 2)))

y_pred_nb = nb.predict(X_test)
```

```
print("Naive Bayes Classifier report: \n\n", classification_report(y_test, y_pred_nb))

cm = confusion_matrix(y_test, y_pred_nb)


x_axis_labels = ["Edible", "Poisonous"]

y_axis_labels = ["Edible", "Poisonous"]


f, ax = plt.subplots(figsize =(7,7))

sns.heatmap(cm, annot = True, linewidths=0.2, linecolor="black", fmt = ".0f", ax=ax, cmap="Purples", xticklabels=x_axis_labels, yticklabels=y_axis_labels)

plt.xlabel("PREDICTED LABEL")

plt.ylabel("TRUE LABEL")

plt.title('Confusion Matrix for Naive Bayes Classifier')

#plt.savefig("nbcm.png", format='png', dpi=900, bbox_inches='tight')

plt.show()


from sklearn.ensemble import RandomForestClassifier


rf = RandomForestClassifier(n_estimators=100, random_state=42)

rf.fit(X_train, y_train)


print("Test Accuracy: {}%".format(round(rf.score(X_test, y_test)*100, 2)))


y_pred_rf = rf.predict(X_test)

print("Random Forest Classifier report: \n\n", classification_report(y_test, y_pred_rf))

cm = confusion_matrix(y_test, y_pred_rf)
```

```
x_axis_labels = ["Edible", "Poisonous"]
y_axis_labels = ["Edible", "Poisonous"]

f, ax = plt.subplots(figsize =(7,7))
sns.heatmap(cm, annot = True, linewidths=0.2, linecolor="black", fmt = ".0f", ax=ax, cmap="Purples", xticklabels=x_axis_labels, yticklabels=y_axis_labels)
plt.xlabel("PREDICTED LABEL")
plt.ylabel("TRUE LABEL")
plt.title('Confusion Matrix for Random Forest Classifier');
#plt.savefig("rfcm.png", format='png', dpi=900, bbox_inches='tight')
plt.show()
preds = dt.predict(X_test)

print(preds[:36])
print(y_test[:36].values)

# 0 - Edible
# 1 - Poisonous
```