Pre-requisitos:

- Python 3.5 o superior, numpy, matplotlib, y scikit.

1. Crear un archivo nuevo de Python llamado “practice5.py”.
2. Importar las librerías comunes:

```
import numpy as np
import os
```

3. Configurar figuras estéticamente vistosas y un directorio para guardarlas.

```
import matplotlib as mpl
import matplotlib.pyplot as plt
mpl.rc('axes', labelsize=14)
mpl.rc('xtick', labelsize=12)
mpl.rc('ytick', labelsize=12)


PROJECT_ROOT_DIR = "."
CHAPTER_ID = "decision_trees"
IMAGES_PATH = os.path.join(PROJECT_ROOT_DIR, "images", CHAPTER_ID)
os.makedirs(IMAGES_PATH, exist_ok=True)

def save_fig(fig_id, tight_layout=True, fig_extension="png", resolution=300):
    path = os.path.join(IMAGES_PATH, fig_id + "." + fig_extension)
    print("Saving figure", fig_id)
    if tight_layout:
        plt.tight_layout()
    plt.savefig(path, format=fig_extension, dpi=resolution)
```

4. Inducir un árbol de decisión usando el dataset clásico del Iris: https://scikit-learn.org/stable/auto_examples/datasets/plot_iris_dataset.html
5. Dibujar el modelo del árbol de decisión usando el siguiente código:

```
from graphviz import Source
from sklearn.tree import export_graphviz

export_graphviz(
        tree_clf,
        out_file=os.path.join(IMAGES_PATH, "iris_tree.dot"),
        feature_names=iris.feature_names[2:],
        class_names=iris.target_names,
        rounded=True,
        filled=True
    )

Source.from_file(os.path.join(IMAGES_PATH, "iris_tree.dot"))
```

6. Visualizar el modelo del árbol de decisión al instalar el paquete `graphviz`. A continuación, entrar al directorio de trabajo usando una terminal (`workingfolder\images\decision_trees\`). Convertir el archive DOT en una imagen PNG usando el commando: `dot -Tpng iris_tree.dot -o iris_tree.png`

7. Probar el árbol de decisión final con los conjuntos de entrenamiento y de prueba.
8. Calcular la precisión del modelo.
9. Visualizar las particiones generadas por cada split, usando el siguiente código:

```
from matplotlib.colors import ListedColormap

def plot_decision_boundary(clf, X, y, axes=[0, 7.5, 0, 3], iris=True, legend=False,
plot_training=True):
    x1s = np.linspace(axes[0], axes[1], 100)
    x2s = np.linspace(axes[2], axes[3], 100)
    x1, x2 = np.meshgrid(x1s, x2s)
    X_new = np.c_[x1.ravel(), x2.ravel()]
    y_pred = clf.predict(X_new).reshape(x1.shape)
    custom_cmap = ListedColormap(['#fafab0','#9898ff','#a0faa0'])
    plt.contourf(x1, x2, y_pred, alpha=0.3, cmap=custom_cmap)
    if not iris:
        custom_cmap2 = ListedColormap(['#7d7d58','#4c4c7f','#507d50'])
        plt.contour(x1, x2, y_pred, cmap=custom_cmap2, alpha=0.8)
    if plot_training:
        plt.plot(X[:, 0][y==0], X[:, 1][y==0], "yo", label="Iris setosa")
        plt.plot(X[:, 0][y==1], X[:, 1][y==1], "bs", label="Iris versicolor")
        plt.plot(X[:, 0][y==2], X[:, 1][y==2], "g^", label="Iris virginica")
        plt.axis(axes)
    if iris:
        plt.xlabel("Petal length", fontsize=14)
        plt.ylabel("Petal width", fontsize=14)
    else:
        plt.xlabel(r"$x_1$", fontsize=18)
        plt.ylabel(r"$x_2$", fontsize=18, rotation=0)
    if legend:
        plt.legend(loc="lower right", fontsize=14)

plt.figure(figsize=(8, 4))
plot_decision_boundary(tree_clf, X, y)
plt.plot([2.45, 2.45], [0, 3], "k-", linewidth=2)
plt.plot([2.45, 7.5], [1.75, 1.75], "k--", linewidth=2)
plt.text(1.40, 1.0, "Depth=0", fontsize=15)
plt.text(3.2, 1.80, "Depth=1", fontsize=13)
save_fig("decision_tree_decision_boundaries_plot")
plt.show()
```

10. Aplicar el mismo proceso en los datasets wine y breast_cancer adjuntos a esta práctica, omitiendo el paso 10. A continuación se muestran los enlaces con información de los datasets:

    https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_wine.html

    https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html

Reporte de la práctica

Emplear el formato de práctica dado por el profesor y seguir las instrucciones mostradas. El archivo que se subirá a *Canvas* deberá estar estrictamente en formato PDF y deberá ser nombrado como `report.pdf`.

Usar el lenguaje `Python` para desarrollar la práctica. **Únicamente será aceptado este lenguaje para la generación de los programas**. Además, es forzoso el uso de `Scikit-learn`. Entregar el programa con extensión `.py` debidamente comentado. Entregar un archivo `README.txt` donde se exponga cómo ejecutar el programa (indicar los parámetros en caso de necesitarlos) y un ejemplo para cómo ejecutar el programa y producir así los resultados reportados.

Entrega global

Tanto el reporte y el programa deberán ser empaquetados en un archivo `.ZIP` y nombrarlo: `practice5.zip`. **Cualquier falta a las instrucciones pedidas implicará la anulación de la práctica para todos los integrantes del equipo.**