Tarea Árboles de Decisión

Joaquín Sandoval Miramontes

Noviembre 2022

1. Introducción

Teniendo los conjuntos de datos `glass.csv` y `diabetes.csv` vamos a estudiar cómo funcionan los **árboles de decisión** como **clasificadores**. En específico, revisaremos la implementación hecha por `scikit-learn`, llamada `DecissionTreeClassifier` [2]. Así mismo, utilizaremos el paquete `pandas` para el manejo de los datos.

2. Desarrollo

2.1. Medición de la Precisión de Árboles Multinivel y De un solo Nivel

El primer ejercicio consiste en entrenar árboles de decisión de un nivel, y multinivel en ambos conjuntos de datos, y comparar su precisión.

Para trabajar con los clasificadores, lo primero es leer los datos. La biblioteca `pandas` cuenta con la función `read_csv` la cual nos facilita este paso y nos regresa los datos en forma de `dataframe`. Una vez obtenidos los datos, separamos la clase y los valores con la flexibilidad de *pandas*. Después, dividimos los datos en datos de entrenamiento y datos de prueba con la función `train_test_split()` de *scikit-learn*.

Lo que sigue es definir el árbol de decisión, eligiendo *entropy* como criterio de separación. Para construir el árbol de un nivel, se establece el parámetro `max_depth` igual a 1. Para árboles multi-nivel, se asigna el valor `None`. [1]

Medimos la precisión de clasificación del árbol con la función `accuracy_score()`, la cual recibe el valor real y el valor predicho, y con eso calcula qué tan precisa fue la clasificación. Para hacer más vistoso el resultado, se graficó la precisión de cada árbol con una gráfica de barras.

El código es el siguiente:

```
def test_models(dataset_path: str):

    data = pd.read_csv(dataset_path)
    Y = data['class']
    X = data.drop(['class'],axis=1)
    X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.34, random_state=10)


    tree_one_level   = DecisionTreeClassifier(criterion='entropy',max_depth=1)
    tree_multi_level = DecisionTreeClassifier(criterion='entropy',max_depth=None)

    tree_one_level   = tree_one_level.fit(X_train,Y_train)
    tree_multi_level = tree_multi_level.fit(X_train,Y_train)


    Y_prediction_one_level   = tree_one_level.predict(X_test)
    Y_prediction_multi_level = tree_multi_level.predict(X_test)

    accuracy = {
                'One-level Decission Tree':accuracy_score(Y_test,Y_prediction_one_level),
```

```
                'Multi-level Decission Tree':accuracy_score(Y_test,Y_prediction_multi_level)
                }


    labels = list(accuracy.keys())
    vals   = list(accuracy.values())

    fig = plt.figure(figsize=(4,3))
    plt.bar(labels,vals,color='green',width=0.2)
    plt.ylabel('Precision')
    nombre = dataset_path.split('.')[0]
    plt.title('Precision de arboles de decision '+nombre)
    plt.savefig(nombre+'_accuracy.png')


    tree_mlt = export_text(tree_multi_level,feature_names=list(X.columns))
    f = open('multilevel_tree_structure_'+nombre+'.txt','w')
    f.write(tree_mlt)
    f.close()

    tree_one = export_text( tree_one_level,feature_names=list(X.columns))
    f = open('onelevel_tree_structure_'+nombre+'.txt','w')
    f.write(tree_one)
    f.close()
```

Los resultados fueron los siguientes:

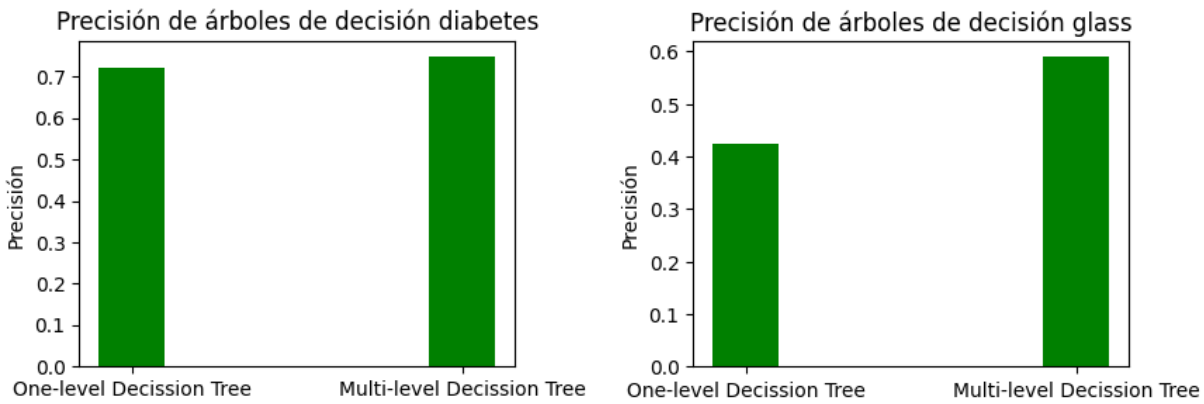

Figura 1: Precisión medida de cada Árbol Clasificador con cada conjunto de datos

Como se puede ver, en todos los casos la precisión no es muy buena, pues sólo se cambió el parámetro que afecta al número de niveles, mas no ajustamos qué tan específico debía ser el árbol. Curiosamente, con el conjunto de datos de diabetes, la precisión de ambos árboles es muy similar, lo que indica que se pudo encontrar un parámetro lo suficientemente bueno para clasificar las entradas a su respectiva clase.

En cambio, en el conjunto de datos del vidrio, la precisión varía cerca de un 20 % entre el árbol de un solo nivel, y el árbol multinivel, lo que demuestra que el evaluar varios atributos ayuda a mejorar la clasificación con este conjunto de datos.

2.2. Medición de la Precisión de Árboles Multinivel de Acuerdo al Número de Muestras para Dividir

Lo siguiente fuer revisar cómo si aumentamos el número mínimo de hojas que debe tener cada rama de cada nodo al ser dividido. Generalmente, un número muy pequeño genera un árbol demasiado específico, mientras que un número muy grande evitará que el árbol aprenda correctamente los datos, siendo demasiado general. [1]

Para observar este efecto, repetimos los pasos anteriores de lectura de datos, separación en datos de entrenamiento y datos de prueba, y usamos un ciclo donde variamos el parámetro `min_samples_leaf()`, entrenamos el árbol, medimos su precisión y lo almacenamos en una lista para finalmente graficarla.

El código para realizar esto es el siguiente:

```
    def measure_plot_accuracy(dataset_path:str):
        data = pd.read_csv(dataset_path)
        Y = data['class']
        X = data.drop(['class'],axis=1)
        X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.34, random_state
    =10)

        accuracy = []
        num_samples = []


        for n in range(1,len(Y_train),5):
            multi_level_tree = DecisionTreeClassifier(criterion='entropy',max_depth=None,
    min_samples_leaf=n)
            multi_level_tree = multi_level_tree.fit(X_train, Y_train)

            prediction = multi_level_tree.predict(X_test)

            accuracy.append(accuracy_score(Y_test,prediction))
            num_samples.append(n)


        nombre = dataset_path.split('.')[0]
        fig = plt.figure(figsize=(4,3))
        plt.plot(num_samples,accuracy)
        plt.xlabel('Min Samples Leaf')
        plt.ylabel('Accuracy')
        plt.title('Variacion de precision segun numero de muestras '+nombre)
        plt.savefig('min_sample_acc_'+nombre+'.png')
```

Los resultados obtenidos fueron los siguientes:

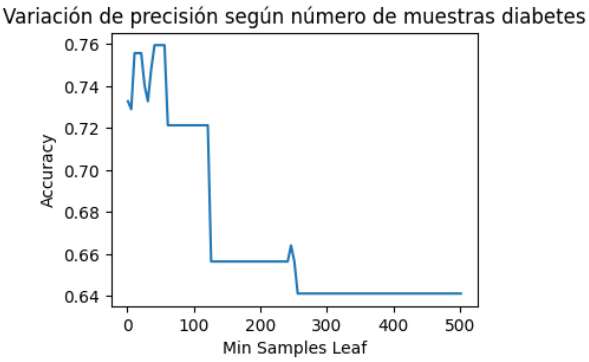

Figura 2: Precisión contra Número Mínimo de Hojas en cada Rama en datos de Diabetes

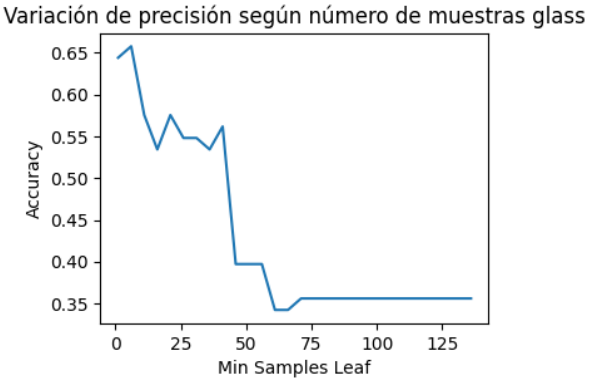

Figura 3: Precisión contra Número Mínimo de Hojas en cada Rama en datos de Vidrio

Como se puede ver en las figuras 2 y 3, conforme aumenta el parámetro `min_sample_leaf`, disminuye considerablemente la precisión. Curiosamente en ambos casos, la precisión con un número pequeño de hojas es alta,y comienza a bajar cuando el parámetro es aproximadamente 1/5 del tamaño del conjunto de datos.

También cabe destacar que cuando el parámetro es muy pequeño, la precisión no es tan alta, aunque este resultado es visible en una zona muy pequeña. Debido a que la variación del parámetro la hicimos de 5 en 5, si hubiéramos tomado una resolución más baja quizás habríamos observado mejor los efectos de la variación del parámetro.

Podemos detectar que la zona de **sobreajuste** es desde 1 a 5 en el conjunto de datos de vidrio, y de 1 a 5 aproximadamente en el conjunto de datos de diabetes. Luego, se tiene una zona con un rendimiento bueno, para llegar a la zona de **subajuste** de 75 en adelante en los datos de diabetes, y de 15 en adelante para los datos del vidrio.

3. Conclusiones

La tarea fue bastante ilustrativa para ver cómo funcionan los árboles de decisión, qué tan efectivos son, y cómo tenemos que ajustarlos para que funcionen de la mejor manera. De la misma manera, pudimos experimentar con los distintos parámetros y ver su efecto en la precisión de la clasificación. Creo que el trabajo tuvo resultados positivos, pues el comportamiento de los modelos y su precisión fue el esperado.

Referencias

[1] scikit-learn. *Decision Trees*. 2022. URL: `https://scikit-learn.org/stable/modules/tree.html`.

[2] scikit-learn. *sklearn.tree.DecisionTreeClassifier*. 2022. URL: `https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html`.