Machine Learning Lab3

July 21, 2019

Aim: Implement Decsion Tree classifier

1. Objective:

- Implement Decision Tree classifier using scikit learn library
- Test the classifier in the following dataset,
    - Dataset1: Weather example discussed in the class
    - Dataset2: Zoo dataset

2. Weather Example

```
Step 1: Import necessary libraries.
```

```
In [1]: from sklearn import preprocessing
        from sklearn.tree import DecisionTreeClassifier
```

```
Step 2: Prepare dataset.
```

```
In [2]: #Predictor variables
        Outlook = ['Rainy', 'Rainy', 'Overcast', 'Sunny', 'Sunny', 'Sunny', 'Overcast',
                    'Rainy', 'Rainy', 'Sunny', 'Rainy','Overcast', 'Overcast', 'Sunny']
        Temperature = ['Hot', 'Hot', 'Hot', 'Mild', 'Cool', 'Cool', 'Cool',
                        'Mild', 'Cool', 'Mild', 'Mild', 'Mild', 'Hot', 'Mild']
        Humidity = ['High', 'High', 'High', 'High', 'Normal', 'Normal', 'Normal',
                    'High', 'Normal', 'Normal', 'Normal', 'High', 'Normal', 'High']
        Wind = ['False', 'True', 'False', 'False', 'False', 'True', 'True',
                    'False', 'False', 'False', 'True', 'True', 'False', 'True']

        #Class Label:
        Play = ['No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'No',
        'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'No']
```

```
Step 3: Digitize the data set using encoding
```

```
In [3]: #creating labelEncoder
        le = preprocessing.LabelEncoder()

        # Converting string labels into numbers.
        Outlook_encoded = le.fit_transform(Outlook)
        Outlook_name_mapping = dict(zip(le.classes_, le.transform(le.classes_)))
        print("Outllok mapping:",Outlook_name_mapping)

        Temperature_encoded = le.fit_transform(Temperature)
        Temperature_name_mapping = dict(zip(le.classes_, le.transform(le.classes_)))
        print("Temperature mapping:",Temperature_name_mapping)

        Humidity_encoded = le.fit_transform(Humidity)
        Humidity_name_mapping = dict(zip(le.classes_, le.transform(le.classes_)))
        print("Humidity mapping:",Humidity_name_mapping)

        Wind_encoded = le.fit_transform(Wind)
        Wind_name_mapping = dict(zip(le.classes_, le.transform(le.classes_)))
        print("Wind mapping:",Wind_name_mapping)

        Play_encoded = le.fit_transform(Play)
        Play_name_mapping = dict(zip(le.classes_, le.transform(le.classes_)))
        print("Play mapping:",Play_name_mapping)

        print("\n\n")
        print("Weather:" ,Outlook_encoded)
        print("Temerature:" ,Temperature_encoded)
        print("Humidity:" ,Humidity_encoded)
        print("Wind:" ,Wind_encoded)
        print("Play:" ,Play_encoded)
```

```
Outllok mapping: {'Overcast': 0, 'Rainy': 1, 'Sunny': 2}
Temperature mapping: {'Cool': 0, 'Hot': 1, 'Mild': 2}
Humidity mapping: {'High': 0, 'Normal': 1}
Wind mapping: {'False': 0, 'True': 1}
Play mapping: {'No': 0, 'Yes': 1}



Weather: [1 1 0 2 2 2 0 1 1 2 1 0 0 2]
Temerature: [1 1 1 2 0 0 0 2 0 2 2 2 1 2]
Humidity: [0 0 0 0 1 1 1 0 1 1 1 0 1 0]
Wind: [0 1 0 0 0 1 1 0 0 0 1 1 0 1]
Play: [0 0 1 1 1 0 1 0 1 1 1 1 1 0]
```

Step 4: Merge different features to prepare dataset

```
In [4]: #Combinig all features into single listof tuples
        features=tuple(zip(Outlook_encoded,Temperature_encoded,Humidity_encoded,Wind_encoded))
        print("Features:",features)
```

```
Features: ((1, 1, 0, 0), (1, 1, 0, 1), (0, 1, 0, 0), (2, 2, 0, 0), (2, 0, 1, 0), (2, 0, 1, 1),
```

Step 5: Train Create and Train DecisionTreeClassifier

```
In [5]: #Create a Decision Tree Classifier (using Entropy)
        clf_entropy = DecisionTreeClassifier(criterion = "entropy")

        # Train the model using the training sets
        clf_entropy.fit(features,Play_encoded)
```

```
Out[5]: DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
                    max_features=None, max_leaf_nodes=None,
                    min_impurity_decrease=0.0, min_impurity_split=None,
                    min_samples_leaf=1, min_samples_split=2,
                    min_weight_fraction_leaf=0.0, presort=False,
                    random_state=None, splitter='best')
```

Step 6: Predict Output for new data

```
In [6]: #Predict Output
        predicted= clf_entropy.predict([[1,2,1,0],[2,0,0,1]])
        print("Predicted Play Values:", predicted)
```

```
Predicted Play Values: [1 0]
```

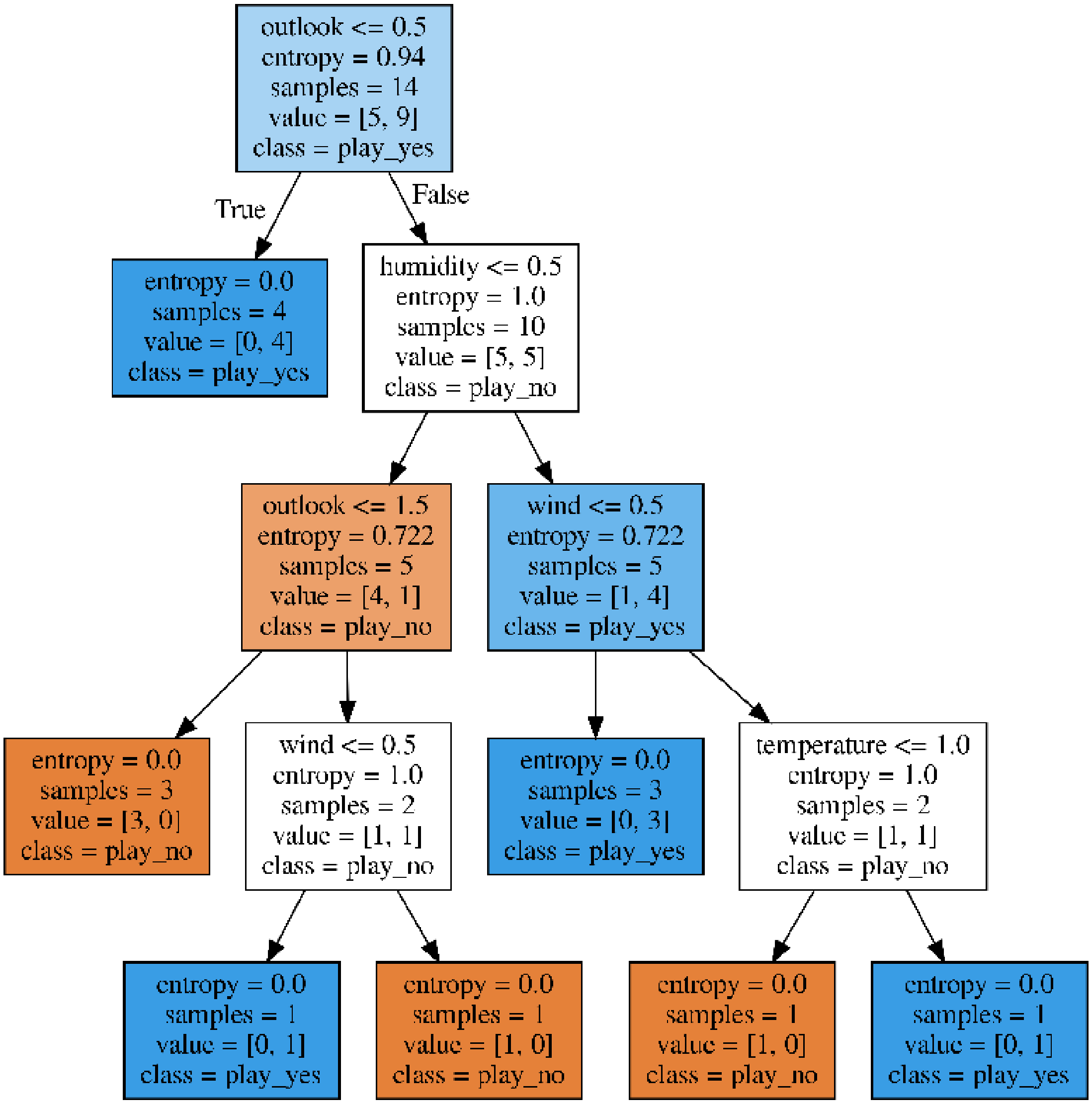
Step 7: Display Decsion Tree Created

- This step requires graphviz and tkinter packages installed on your system
- For CentOS Linux use
    - yum install graphviz
    - yum install python36u-tkinter

```
In [8]: from sklearn.tree import export_graphviz
        export_graphviz(clf_entropy,out_file='tree_entropy.dot',
                        feature_names=['outlook','temperature','humidity','wind'],
                        class_names=['play_no','play_yes'],
                        filled=True)

        # Convert to png
        from subprocess import call
        call(['dot', '-Tpng', 'tree_entropy.dot', '-o', 'tree_entropy.png', '-Gdpi=600'])

        # Display in python
        import matplotlib.pyplot as plt
        plt.figure(figsize = (14, 18))
        plt.imshow(plt.imread('tree_entropy.png'))
        plt.axis('off');
        plt.show();
```



3. Exercise:

   1. Create a Decision Tree for the same weather data as above with Gini as attribute selection measure and find the confusion matrix, accuracy, precision and recall
   2. Create a Decision Tree for zoo dataset using both entropy and gini as attribute selection measure and find the confusion matrix, accuracy, precision and recall