MINORS Lab 3: KNN and DT

Varun Kamath

2019110023

6/12/2022

1. Upload csv files and import libraries

```
[4] from google.colab import drive
    drive.mount('/content/drive')
    df = pd.read_csv('/content/drive/MyDrive/data/breast-cancer.csv')
```

```
Mounted at /content/drive
```

```
[6] df.head()
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...	radius_worst	texture_worst	per
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	...	25.38	17.33	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	...	24.99	23.41	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	...	23.57	25.53	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	...	14.91	26.50	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	...	22.54	16.67	

5 rows × 32 columns

2. Check for null values and proceed to replace the categorical data to some integer value

```
[13] df['diagnosis'].replace(['B', 'M'], [0, 1], inplace=True)
```

3. Import decision tree classifier and split the dataset into train and test data. Predict outcome of test input and compare with the actual outcomes for obtaining accuracy

```
[16] from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier
     from sklearn.model_selection import train_test_split # Import train_test_split function
     from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation
```

```
[18] datavar = pd.DataFrame(df, columns=['id','radius_mean', 'texture_mean', 'perimeter_mean',
            'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
            'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
            'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
            'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
            'fractal_dimension_se', 'radius_worst', 'texture_worst',
            'perimeter_worst', 'area_worst', 'smoothness_worst',
            'compactness_worst', 'concavity_worst', 'concave points_worst',
            'symmetry_worst', 'fractal_dimension_worst'])

     target=df['diagnosis']
```

```
X = datavar
y = target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
clf = DecisionTreeClassifier()
clf = clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9385964912280702
```

4. Repeat the same procedure with decision tree regressor

```
from sklearn.tree import DecisionTreeRegressor


regressor = DecisionTreeRegressor(random_state = 0)
regressor.fit(X, y)
```

```
DecisionTreeRegressor(random_state=0)
```

```
[21] y_pred = regressor.predict(X_test)
```

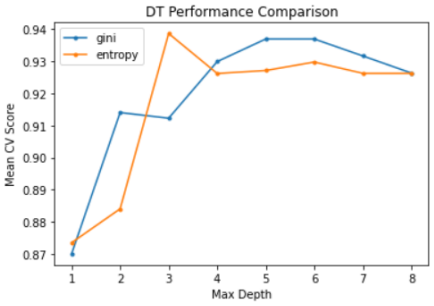
```
[22] print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

     Accuracy: 1.0
```

5. Plot gini impurity and the entropy info gain

```
for i in ['gini', 'entropy']:
    temp = results_DT[results_DT['criterion'] == i]
    temp_average = temp.groupby('max_depth').agg({'test_score': 'mean'})
    plt.plot(temp_average, marker = '.', label = i)


plt.legend()
plt.xlabel('Max Depth')
plt.ylabel("Mean CV Score")
plt.title("DT Performance Comparison")
plt.show()
```



- Successfully trained a model and predicted outcome of the test input data using decision tree
- Plotted gini impurity and entropy information gain to compare using which parameter a higher accuracy can be achieved.