CO544 - Machine Learning and Data Mining
Lab 05

E/17/407
WIJESOORIYA H.D

Task 1: Build two decision tree classifiers with Gini index and entropy criteria for the given Wine.csv data set.

Classifier with the Gini Index criterion

```python
clf_gini = DecisionTreeClassifier(criterion='gini'
,max_depth=4,random_state=0) # Create decision treeclassifier object
clf_gini.fit(X_train, Y_train) # Train the classifier
```

```
DecisionTreeClassifier(max_depth=4, random_state=0)
```

Classifier with the Gini Index criterion

```python
[75]
clf_entropy = DecisionTreeClassifier(criterion='entropy'
,max_depth=4,random_state=0) # Create decision treeclassifier object
clf_entropy.fit(X_train, Y_train) # Train the classifier
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=4, random_state=0)
```

Task 2: Demonstrate how decision trees deal with missing values

```python
# Checking missing values in variables
dataset_df.isnull().sum()
```

```
Alcohol                         0
Malic acid                      0
Ash                             0
Alcalinity of ash               0
Magnesium                       0
Total phenols                   0
Flavanoids                      0
Nonflavanoid phenols            0
Proanthocyanins                 0
Color intensity                 0
Hue                             0
OD280/OD315 of diluted wines    0
Proline                         0
Class                           0
dtype: int64
```

According to this figure, it is clear that there were no missing values in the provided dataset.

We can use the following methods to deal with missing values if there are missing values.

- Method 1: Purification by skipping - Here we can use two main methods . They are,
    - Skip (remove) data points where any feature contains a missing value -make sure only a few data points are skipped
    - Skip an entire feature if it's missing for many data points - make sure only a few features are skipped

  If the dataset is very large it is better to use this method.

- Method 2: Purification by imputing - Here we can impute/substitute missing values.
    - Categorical features use mode: most popular value (mode) of non-missing xi
    - Numerical features use average or median: Average or median value of non-missing xi

  If the dataset is very small, it is better to use this method.

- Method 3: Adapt the learning algorithm to be robust to missing values. Every decision node includes a choice of response to missing values

Task 3: Evaluate the classifiers with suitable performance matrices.

- Evaluating the classifiers using accuracy score and confusion matrix.

```
print('Decision tree classifier with \
the Gini index criterion')
print()
print('Accuracy on test data (accuracy score): '
, metrics.accuracy_score(Y_test, Y_pred))
print('Accuracy on train data (accuracy score): '
, metrics.accuracy_score(Y_train, Y_train_pred))
print()
print('Accuracy (confusion_matrix):')
print(confusion_matrix(Y_test, Y_pred))
```

```
Decision tree classifier with the Gini index criterion

Accuracy on test data (accuracy score):  0.3333333333333333
Accuracy on train data (accuracy score):  0.6541353383458647

Accuracy (confusion_matrix):
[[11  3  4]
 [11  4  2]
 [ 7  3  0]]
```

```
[77]
print('Decision tree classifier \
with the entropy criterion')
print()
print('Accuracy Accuracy on test data (accuracy score):'
, metrics.accuracy_score(Y_test,Y1_pred))
print('Accuracy Accuracy on train data (accuracy score):'
, metrics.accuracy_score(Y_train,Y1_train_pred))
print()
print('Accuracy (confusion_matrix):')
print(confusion_matrix(Y_test, Y1_pred))
```

```
Decision tree classifier with the entropy criterion

Accuracy Accuracy on test data (accuracy score): 0.3333333333333333
Accuracy Accuracy on train data (accuracy score): 0.6541353383458647

Accuracy (confusion_matrix):
[[11  3  4]
 [11  4  2]
 [ 7  3  0]]
```

Task 4: Demonstrate how pruning can be applied to overcome the overfitting of decision tree classifiers.

According to the above figures we can see a huge difference between the accuracies of test and train data. (accuracy on train data = 0.6541353383458647, accuracy on test data

=0.3333333333333333) . This occurs due to overfitting of the dataset . To prevent overfitting we can use ‘Pruning’. There are 2 methods of pruning and they are shown bellow.

- **Method 1:** Post Pruning → takes a fully-grown decision tree and discard unreliable parts. Here we control the branches of the decision tree that is max_depth and min_samples_split using cost_complexity_puring. In the implementation we need to change the ccp_alpha. Finally we were able to improve the accuracy on test data set (0.37777777777777777) as shown bellow.

```python
clf_gini_after_postPruning=DecisionTreeClassifier(criterion='gini'
    ,random_state=0,ccp_alpha=0.04)
clf_gini_after_postPruning.fit(X_train,Y_train)
```

```python
print("Accuracy on test data set (gini index)\
 after applying post pruning method")
accuracy_score(Y_test,
              clf_gini_after_postPruning.predict(X_test))
```

```
Accuracy on test data set (gini index) after applying post pruning method
0.37777777777777777
```

- **Method 2:** Pre Pruning → stop growing a branch when information becomes unreliable. This can be done using Hyperparameter tuning. Finally we were able to improve the accuracy on test data set (0.4) as shown bellow.

```python
# use GridSearchCV for Hyperparameter tuning.

from sklearn.model_selection import GridSearchCV
grid_param={"criterion":["gini","entropy"],
            "splitter":["best","random"],
            "max_depth":range(2,50,1),
            "min_samples_leaf":range(1,15,1),
            "min_samples_split":range(2,20,1)
           }
grid_search=GridSearchCV(estimator=clf,param_grid=grid_param,cv=5,n_jobs=-1)
grid_search.fit(X_train,Y_train)
```

```python
# best parameters to feed to the Decision
# treee classifier
print(grid_search.best_params_)
```

```
{'criterion': 'entropy', 'max_depth': 13, 'min_samples_leaf': 2, 'min_samples_split': 9, 'splitter': 'random'}
```

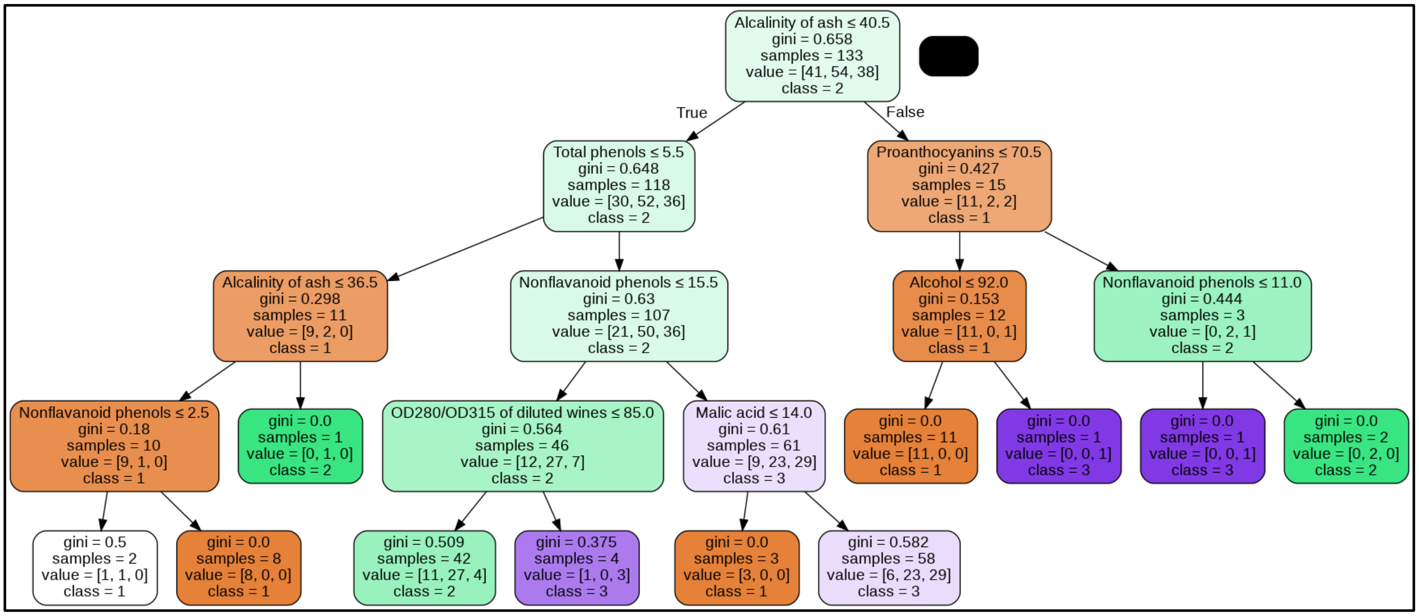
```python
# checkig the accuracy on test data

print("Accuracy on test data set (entropy criteria)\
 after applying pre pruning method")
accuracy_score(Y_test,clf_entropy_after_prePruning.predict(X_test))
```

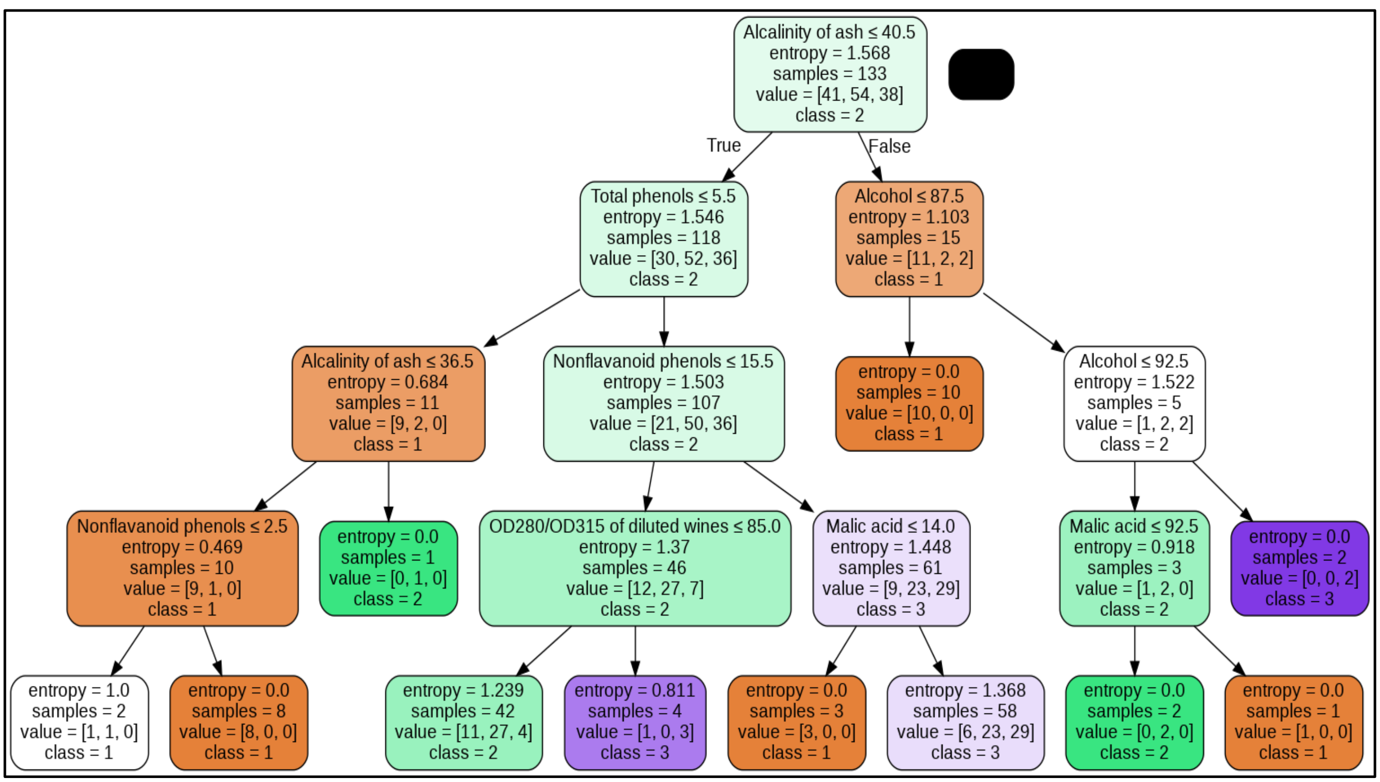
```
Accuracy on test data set (entropy criteria) after applying pre pruning method
0.4
```

Task 5: Visualize decision trees.

Gini Index method



Entropy method



Appendix:

Colab note book link :

https://colab.research.google.com/drive/1bGtyP6YSEygyGdD8XTh7ZMhe0LOnpKnK?usp=sharing

Code :

```
# import libraries

# for data analysis
import numpy as np
import pandas as pd


# to visualize data
from matplotlib import pyplot as plt
import seaborn as sns; sns.set(font_scale=1.2)

# to split the dataset
from sklearn.model_selection import train_test_split

# to access the Decision tree classifier model
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree

# to test the accuracy of the model
from sklearn.metrics import accuracy_score,confusion_matrix
from sklearn import metrics # scikit-
learn metrics module for computing accuracy

%matplotlib inline

# load dataset from the google drive
url='https://drive.google.com/file/d/1V_gDZfGLAsHHDFFlXhdbcuu_U3F3ZUvo
/view?usp=sharing'
url='https://drive.google.com/uc?id=' + url.split('/')[-2]
dataset_df=pd.read_csv(url)
```

```
# get the shape of the dataset
rows,cols=dataset_df.shape

print('No of rows = ',rows)
print('No of columns = ',cols)
# preview the dataset
dataset_df.head()

# Summary of dataset
dataset_df.info()

# store column names in a list
col_names=dataset_df.columns;
print(col_names)
print(col_names[0])

# Frequency distributions of values in variables

for i in col_names:
  print(dataset_df[i].value_counts())

# Exploring target variable
dataset_df['Class'].value_counts()

# Checking missing values in variables
dataset_df.isnull().sum()

X=dataset_df.drop(['Class'],axis=1) # drop the whole column 'Class'
Y=dataset_df['Class']

# splitting data
# 75% training and 25% tesT
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.25,random_stat
e=1)
# Shapes of X_train and X_test
X_train.shape, X_test.shape

encode_cols=[]
```

```
for i in range(len(col_names)-1): # ignore the target
  encode_cols.append(col_names[i])

print(encode_cols)


! pip install category_encoders

import category_encoders as ce # Import the relevant library

encoder = ce.OrdinalEncoder(cols=encode_cols)
X_train = encoder.fit_transform(X_train)
X_test = encoder.transform(X_test)


clf_gini = DecisionTreeClassifier(criterion='gini'
,max_depth=4,random_state=0) # Create decision treeclassifier object
clf_gini.fit(X_train, Y_train) # Train the classifier

Y_pred= clf_gini.predict(X_test)
Y_train_pred= clf_gini.predict(X_train)

print('Decision tree classifier with \
the Gini index criterion')
print()
print('Accuracy on test data (accuracy score): '
, metrics.accuracy_score(Y_test, Y_pred))
print('Accuracy on train data (accuracy score): '
, metrics.accuracy_score(Y_train, Y_train_pred))
print()
print('Accuracy (confusion_matrix):')
print(confusion_matrix(Y_test, Y_pred))

from six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus
dot_data = StringIO()
export_graphviz(clf_gini,
```

```
out_file=dot_data,
filled=True,
rounded=True,
special_characters=True,
feature_names=X.columns,
class_names=['1','2','3'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('wine1.png')
Image(graph.create_png())

#path variable gives two things ccp_alphas and impurities

path=clf_gini.cost_complexity_pruning_path(X_train,Y_train)
ccp_alphas,impurities=path.ccp_alphas,path.impurities
print("ccp alpha wil give list of values :",ccp_alphas)
print(" ")
print("Impurities in Decision Tree :",impurities)

#ccp_alphas gives minimum leaf value of decision tree
#and each ccp_aphas will create different - different classifier
#and choose best out of it.ccp_alphas will be added as a parameter in Decisio
nTreeClassifier()

clfs=[]   #will store all the models here
for ccp_alpha in ccp_alphas:
    clf=DecisionTreeClassifier(criterion='gini',random_state=0,ccp_alpha=ccp_
alpha)
    clf.fit(X_train,Y_train)
    clfs.append(clf)
print("Last node in Decision tree is {} and ccp_alpha for last node is {}".fo
rmat(clfs[-1].tree_.node_count,ccp_alphas[-1]))

#Visualizing the accuracy score for train and test set.
train_scores = [clf.score(X_train,  Y_train) for clf in clfs]
test_scores = [clf.score(X_test, Y_test) for clf in clfs]
fig, ax = plt.subplots()
ax.set_xlabel("Alpha")
ax.set_ylabel("Accuracy")
ax.set_title("Accuracy vs alpha for training and testing sets")
```

```
ax.plot(ccp_alphas, train_scores, marker='o', label="train",drawstyle="steps-
post")
ax.plot(ccp_alphas, test_scores, marker='o', label="test",drawstyle="steps-
post")
ax.legend()
plt.show()
clf_gini_after_postPruning=DecisionTreeClassifier(criterion='gini'
    ,random_state=0,ccp_alpha=0.04)
clf_gini_after_postPruning.fit(X_train,Y_train)

from six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus
dot_data = StringIO()
export_graphviz(clf_gini_after_postPruning,
out_file=dot_data,
filled=True,
rounded=True,
special_characters=True,
feature_names=X.columns,
class_names=['1','2','3'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('wine3.png')
Image(graph.create_png())


print("Accuracy on test data set (gini index)\
 after applying post pruning method")
accuracy_score(Y_test,
               clf_gini_after_postPruning.predict(X_test))


clf_entropy = DecisionTreeClassifier(criterion='entropy'
,max_depth=4,random_state=0) # Create decision treeclassifier object
clf_entropy.fit(X_train, Y_train) # Train the classifier

Y1_pred= clf_entropy.predict(X_test)
Y1_train_pred= clf_entropy.predict(X_train)
```

```python
print('Decision tree classifier \
with the entropy criterion')
print()
print('Accuracy Accuracy on test data (accuracy score):'
, metrics.accuracy_score(Y_test,Y1_pred))
print('Accuracy Accuracy on train data (accuracy score):'
, metrics.accuracy_score(Y_train,Y1_train_pred))
print()
print('Accuracy (confusion_matrix):')
print(confusion_matrix(Y_test, Y1_pred))

from six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus
dot_data = StringIO()
export_graphviz(clf_entropy,
out_file=dot_data,
filled=True,
rounded=True,
special_characters=True,
feature_names=X.columns,
class_names=['1','2','3'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('wine2.png')
Image(graph.create_png())

# use GridSearchCV for Hyperparameter tuning.

from sklearn.model_selection import GridSearchCV
grid_param={"criterion":["gini","entropy"],
             "splitter":["best","random"],
             "max_depth":range(2,50,1),
             "min_samples_leaf":range(1,15,1),
             "min_samples_split":range(2,20,1)
            }
grid_search=GridSearchCV(estimator=clf,param_grid=grid_param,cv=5,n_jobs=-1)
grid_search.fit(X_train,Y_train)
```

```python
# best parameters to feed to the Decision
# tree classifier
print(grid_search.best_params_)


clf_entropy_after_prePruning=DecisionTreeClassifier(criterion= 'entropy'
,max_depth= 13,min_samples_leaf= 2,min_samples_split= 9,splitter= 'random')
clf_entropy_after_prePruning.fit(X_train,Y_train)



from six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus
dot_data = StringIO()
export_graphviz(clf_entropy_after_prePruning,
out_file=dot_data,
filled=True,
rounded=True,
special_characters=True,
feature_names=X.columns,
class_names=['1','2','3'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('wine4.png')
Image(graph.create_png())



# checkig the accuracy on test data

print("Accuracy on test data set (entropy criteria)\
 after applying pre pruning method")
accuracy_score(Y_test,clf_entropy_after_prePruning.predict(X_test))
```