ass3

October 26, 2018

Chiyu Chen

1 What impurity measures do you know?

Entropy, gini index, and classification error.

2 What is entropy and information gain? Explain feature selection using information gain/entropy technique.

Entropy is a measure of degree of randomness in a system. Information gain is the difference between two entropies or other impurity measures. Whever a feature is selected to split a tree, a new entropy can be calculated. We can split a tree by different features and then get many different entropies, as well as IG. By comparing those IG, we can find out the feature which brings us the highest IG and decreases impurtiy the most. The feature is the best candidate to split our tree in the next iteration.

3 What is Random Forest and why is it good?

Random forests are an ensemble learning method that operate by constructing a lot of decision trees at training time and outputting the class that is the mode of the classes output by individual trees. It trained on different parts of the same training set, with the goal of overcoming over-fitting problem of individual decision tree.

4 For cancer data set. Use three different methods to train a classifier that predicts whether cancer is Malignant or Bening,: a Naïve Bayes, Decision Tree and Random Forest. Look at the confusion matrices: for each method.

```
In [1]: import pandas as pd
        import numpy as np
        from sklearn import metrics
        from sklearn import preprocessing
        from sklearn.model_selection import train_test_split, GridSearchCV
        from sklearn.naive_bayes import GaussianNB
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

import matplotlib.pyplot as plt
%matplotlib inline
```

```
In [2]: wbcd = pd.read_csv("/Users/ChiYuChen/UCSC extension/Predictive Analysis/wisc_bc_data.cs
        wbcd = wbcd.drop(['id'], axis = 1)
        # wbcd.head()
```

```
In [3]: # wbcd.columns
```

```
In [4]: # wbcd.dtypes
```

```
In [5]: # wbcd.info()
```

```
In [6]: # wbcd.iloc[:,0:6].values
```

some data exploration

```
In [7]: # print('The shape of our features is:', wbcd.shape)
```

```
In [8]: # # Descriptive statistics for each column
        # wbcd.describe()
```

```
In [9]: # pd.isnull(wbcd).sum()
```

```
In [10]: y = wbcd['diagnosis']
         x = wbcd.drop('diagnosis', axis = 1)
```

```
In [11]: x = pd.get_dummies(x)
         y = pd.factorize(y)[0]
         # y
```

```
In [12]: # Split the data into training and testing sets, Stratify  ensures that trg
         # set looks similar to test set, making evalaution metrics more reliable
         x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_sta
```

```
In [13]: # Feature Scaling  Presently we are not using it.
         #from sklearn.preprocessing import StandardScaler

         #sc = StandardScaler()
         #x_train = sc.fit_transform(x_train)
         #x_test = sc.transform(x_test)
```

4.1 Decision Tree

```
In [14]: dt = DecisionTreeClassifier()
         model_dt = dt.fit(x_train, y_train)
         pred_dt = model_dt.predict(x_test)
         # print(metrics.confusion_matrix(y_test, pred_dt))
         pd.crosstab(y_test, pred_dt, rownames=['Actual Class'], colnames=['Predicted Class'])
```

```
Out[14]: Predicted Class    0   1
         Actual Class
         0                104   3
         1                  9  55
```

4.2 Random Tree Forest

```
In [15]: rf = RandomForestClassifier(n_estimators = 100, criterion = 'entropy', oob_score=True)
         model_rf = rf.fit(x_train, y_train)
         pred_rf = model_rf.predict(x_test)
         # print(metrics.confusion_matrix(y_test, pred_rf))
         pd.crosstab(y_test, pred_rf, rownames=['Actual Class'], colnames=['Predicted Class'])
```

```
Out[15]: Predicted Class    0   1
         Actual Class
         0                105   2
         1                  7  57
```

4.3 Naive Bayes

```
In [16]: gnb = GaussianNB()
         pred_nb = gnb.fit(x_train, y_train).predict(x_test)
         # print(metrics.confusion_matrix(y_test, pred_nb))
         pd.crosstab(y_test, pred_nb, rownames=['Actual Class'], colnames=['Predicted Class'])
```

```
Out[16]: Predicted Class    0   1
         Actual Class
         0                104   3
         1                  8  56
```

5 Which algorithm do you think performs better? What can you do to improve the performance?

According to my results, random forest performs slightly better than the others. I have tested with different parameters combination by GridSearchCV, and the results shows that it performs slightly better than the default values. This indicates that default values are the almost the best combination for building a tree. Besides, I think the accuracy of 96% is too good to be true. It might possibly overfitted. I think another possiblity of improving the performance is to decrease the differences between behaviors of the training set and the testing set. If the training set covers all behavior of the testing set, it might perform better.

```
In [17]: # rf = RandomForestClassifier(n_estimators = 100, criterion = 'entropy', oob_score =
         rf = RandomForestClassifier(oob_score=True)
         param_grid = {
             'n_estimators': [40, 60, 80, 100, 200, 300, 400, 500],
             'max_features': ['auto', 'sqrt', 'log2'],
             'max_depth' : [4, 5, 6, 7, 8],
             'criterion' :['gini', 'entropy']
```

```
        }
        rf = GridSearchCV(estimator=rf, param_grid=param_grid, scoring='accuracy', n_jobs=-1)
        model_rf = rf.fit(x_train, y_train)
        pred_rf = model_rf.predict(x_test)
        # print(metrics.confusion_matrix(y_test, pred_rf))
        pd.crosstab(y_test, pred_rf, rownames=['Actual Class'], colnames=['Predicted Class'])
```

```
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages/sklearn/model_sel
  warnings.warn(CV_WARNING, FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages/sklearn/model_sel
  DeprecationWarning)
```

Out[17]:

Predicted Class	0	1
Actual Class		
0	105	2
1	5	59