Lab 4

Jiarong Ye

September 13, 2018

Import packages

```
In [18]: import datascience
         import numpy as np
         import pandas as pd
         from datascience import *
         from sklearn.feature_extraction.text import CountVectorizer
         from sklearn.model_selection import train_test_split

         from sklearn.pipeline import Pipeline
         from sklearn import tree
         from sklearn.tree import DecisionTreeClassifier
         from sklearn import metrics

         import graphviz
```

Read the CSV file "Ben-NE-9- 10-2015-pass-6.csv" using Pandas

```
In [5]: Ben_pass= pd.read_csv('Ben-NE-9-10-2015-pass-6.csv', sep=",")
```

```
In [6]: print(Ben_pass)
```

```
    down  ydstogo  Yards.Gained.PrevPlay  AirYards  PassLocation  PassOutcome
0      1       10                     18        -4             1            1
1      1       10                      0         9             1            1
2      3       22                      6         1             1            1
3      1       10                      0         7            -1            1
4      1       10                     13         6            -1            1
5      1       10                     12         7            -1            1
6      1       10                      0         5             1            0
7      2       10                      0        25             1            0
8      3        5                     -1         6            -1            1
9      1       15                      4        -1             1            1
10     3       18                     -6        17            -1            1
11     1       20                      5         5            -1            1
12     2       11                      9         4            -1            1
13     2       13                     -3        -2            -1            1
14     3        6                      7         6             0            1
```

15	2	7	0	11	1	1
16	1	10	13	16	1	1
17	1	10	19	6	1	1
18	2	8	2	0	-1	0
19	3	8	0	39	-1	1
20	3	3	1	19	1	0
21	1	10	0	11	0	1
22	2	10	0	17	-1	1
23	1	10	18	7	0	1
24	1	9	2	4	-1	1
25	2	6	3	6	0	1
26	1	10	0	5	1	1
27	2	15	9	4	-1	1
28	3	6	9	26	1	1
29	2	8	2	0	-1	1
..	...	...	...	...	...	...
40	2	15	-5	7	0	1
41	3	6	-1	2	-1	1
42	1	10	0	25	1	1
43	2	5	5	2	1	1
44	2	8	0	1	1	1
45	1	1	0	1	0	0
46	1	10	0	18	1	1
47	1	10	0	27	0	0
48	3	5	5	4	0	1
49	2	1	0	1	1	0
50	3	1	0	1	-1	1
51	2	3	7	9	0	0
52	3	3	0	9	1	0
53	2	12	-2	23	1	0
54	3	12	0	6	0	0
55	1	10	0	18	1	1
56	1	10	18	39	-1	0
57	2	9	1	7	-1	1
58	3	4	3	-3	1	0
59	1	10	0	11	1	0
60	2	10	0	4	-1	1
61	3	1	9	6	0	1
62	1	10	0	29	0	0
63	2	10	0	1	-1	1
64	3	8	2	13	0	1
65	2	27	-7	14	0	1
66	3	9	0	12	-1	0
67	4	9	0	15	-1	1
68	1	10	0	11	-1	1
69	1	10	11	11	-1	1

[70 rows x 6 columns]

```
In [7]: X= Ben_pass.values[:,0:4]
        print(X)
```

```
[[ 1 10 18 -4]
 [ 1 10  0  9]
 [ 3 22  6  1]
 [ 1 10  0  7]
 [ 1 10 13  6]
 [ 1 10 12  7]
 [ 1 10  0  5]
 [ 2 10  0 25]
 [ 3  5 -1  6]
 [ 1 15  4 -1]
 [ 3 18 -6 17]
 [ 1 20  5  5]
 [ 2 11  9  4]
 [ 2 13 -3 -2]
 [ 3  6  7  6]
 [ 2  7  0 11]
 [ 1 10 13 16]
 [ 1 10 19  6]
 [ 2  8  2  0]
 [ 3  8  0 39]
 [ 3  3  1 19]
 [ 1 10  0 11]
 [ 2 10  0 17]
 [ 1 10 18  7]
 [ 1  9  2  4]
 [ 2  6  3  6]
 [ 1 10  0  5]
 [ 2 15  9  4]
 [ 3  6  9 26]
 [ 2  8  2  0]
 [ 3  3  5 23]
 [ 4  3  0  4]
 [ 2  5  5 26]
 [ 3  5  0  7]
 [ 1 10  0  4]
 [ 3  5  1  5]
 [ 2  1  9 -1]
 [ 2  1  0  0]
 [ 1 10 28 23]
 [ 1 10 13 -5]
 [ 2 15 -5  7]
 [ 3  6 -1  2]
 [ 1 10  0 25]
```

```
 [ 2  5  5  2]
 [ 2  8  0  1]
 [ 1  1  0  1]
 [ 1 10  0 18]
 [ 1 10  0 27]
 [ 3  5  5  4]
 [ 2  1  0  1]
 [ 3  1  0  1]
 [ 2  3  7  9]
 [ 3  3  0  9]
 [ 2 12 -2 23]
 [ 3 12  0  6]
 [ 1 10  0 18]
 [ 1 10 18 39]
 [ 2  9  1  7]
 [ 3  4  3 -3]
 [ 1 10  0 11]
 [ 2 10  0  4]
 [ 3  1  9  6]
 [ 1 10  0 29]
 [ 2 10  0  1]
 [ 3  8  2 13]
 [ 2 27 -7 14]
 [ 3  9  0 12]
 [ 4  9  0 15]
 [ 1 10  0 11]
 [ 1 10 11 11]]
```

```
In [8]: Y= Ben_pass.values[:,5]
```

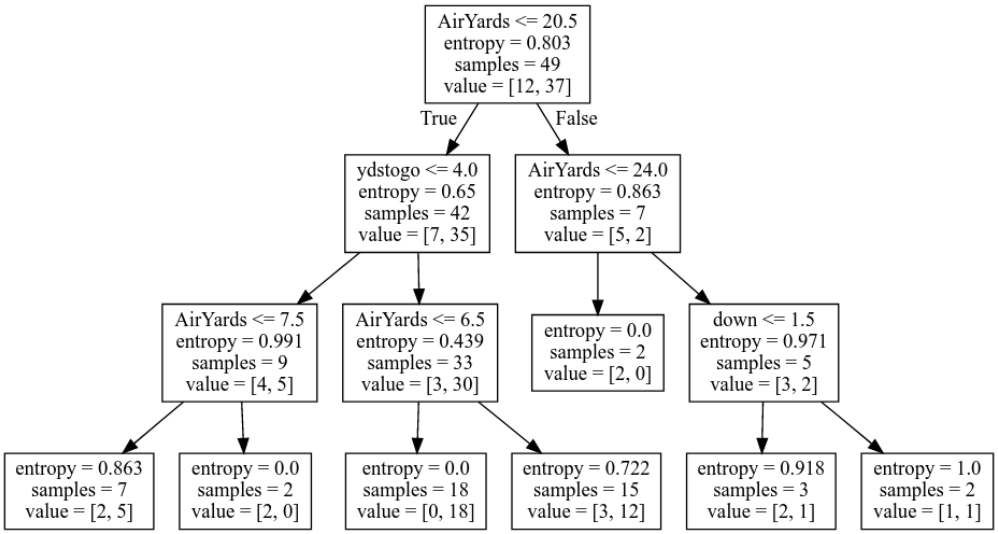
```
In [10]: X_train, X_test, y_train, y_test = train_test_split( X, Y, test_size = 0.3, random_stat
```

(1) Generate a visualization of the Decision Tree learned from the training data (set the max_depth to 3)

```
In [11]: clf = DecisionTreeClassifier(criterion = 'entropy', random_state = 100,
                                       max_depth=3, min_samples_leaf=2)
         clf.fit(X_train, y_train)
```

```
Out[11]: DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=3,
                     max_features=None, max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=2, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, presort=False, random_state=100,
                     splitter='best')
```

```
In [15]: dot_data= tree.export_graphviz(clf, out_file=None, feature_names=Ben_pass.columns[:4])
```




```
In [19]: graph = graphviz.Source(dot_data)
```

```
In [22]: graph.render('BenPassTree3')
```

```
Out[22]: 'BenPassTree3.pdf'
```

```
In [23]: X_train
```

```
Out[23]: array([[ 3,  3,  5, 23],
                [ 1, 10, 19,  6],
                [ 3,  5,  0,  7],
                [ 1, 10,  0, 11],
                [ 2,  9,  1,  7],
                [ 2, 15, -5,  7],
                [ 3,  1,  0,  1],
                [ 2,  6,  3,  6],
                [ 1, 10,  0, 27],
                [ 2,  3,  7,  9],
                [ 2,  1,  0,  0],
                [ 3,  5,  1,  5],
                [ 1, 10,  0, 11],
                [ 1, 10,  0, 25],
                [ 2, 10,  0, 25],
                [ 1, 10,  0,  7],
                [ 1, 10,  0, 18],
                [ 1,  1,  0,  1],
                [ 1, 10,  0, 29],
                [ 3,  8,  0, 39],
                [ 2, 13, -3, -2],
                [ 1, 10,  0,  9],
```

```
                [ 4,  3,  0,  4],
                [ 1, 10, 13,  6],
                [ 2,  8,  0,  1],
                [ 2, 15,  9,  4],
                [ 3,  1,  9,  6],
                [ 2, 10,  0, 17],
                [ 2,  8,  2,  0],
                [ 1, 15,  4, -1],
                [ 1, 10, 13, 16],
                [ 2,  5,  5,  2],
                [ 2,  1,  9, -1],
                [ 2,  7,  0, 11],
                [ 1, 10,  0, 11],
                [ 3,  9,  0, 12],
                [ 2,  1,  0,  1],
                [ 2, 10,  0,  4],
                [ 3,  6,  7,  6],
                [ 2, 10,  0,  1],
                [ 3, 22,  6,  1],
                [ 2, 12, -2, 23],
                [ 1, 10,  0,  4],
                [ 3,  3,  0,  9],
                [ 3, 18, -6, 17],
                [ 3,  5,  5,  4],
                [ 4,  9,  0, 15],
                [ 1,  9,  2,  4],
                [ 3,  5, -1,  6]])

In [24]: X_test

Out[24]: array([[ 1, 10, 13, -5],
                [ 1, 10,  0,  5],
                [ 1, 10, 18,  7],
                [ 1, 10, 18, 39],
                [ 1, 10, 28, 23],
                [ 3,  8,  2, 13],
                [ 1, 10,  0, 18],
                [ 1, 20,  5,  5],
                [ 2,  8,  2,  0],
                [ 3,  4,  3, -3],
                [ 2, 27, -7, 14],
                [ 1, 10, 12,  7],
                [ 3,  3,  1, 19],
                [ 1, 10, 11, 11],
                [ 1, 10, 18, -4],
                [ 3,  6,  9, 26],
                [ 2, 11,  9,  4],
                [ 1, 10,  0,  5],
```

```
                [ 3,  6, -1,  2],
                [ 2,  5,  5, 26],
                [ 3, 12,  0,  6]])
```

```
In [25]: predicted_completion = clf.predict(X_test)
```

```
In [26]: print(predicted_completion)
```

```
[1 1 1 0 0 1 1 1 1 1 1 1 0 1 1 0 1 1 1 0 1]
```

```
In [27]: print(y_test)
```

```
[1 1 1 0 1 1 1 1 0 0 1 1 0 1 1 1 1 0 1 0 0]
```

```
In [28]: np.mean(predicted_completion == y_test)
```

```
Out[28]: 0.7142857142857143
```

(2) Discuss the result of testing the model using confusion matrix

```
In [29]: print(metrics.classification_report(y_test, predicted_completion))
```

```
             precision    recall  f1-score   support

          0       0.60      0.43      0.50         7
          1       0.75      0.86      0.80        14

avg / total       0.70      0.71      0.70        21
```

```
In [30]: metrics.confusion_matrix(y_test, predicted_completion)
```

```
Out[30]: array([[ 3,  4],
                [ 2, 12]])
```

From the confusion matrix, we can collect the following stats:

$$Precision: \frac{3+12}{3+4+2+12} = \frac{15}{21} = 0.714$$

$$False\ Positive\ Rate = \frac{4}{4+12} = 0.25$$

$$False\ Negative\ Rate = \frac{2}{3+2} = 0.4$$

(3) Describe one rule (different from the one explained in the slides) extracted from the tree you generated

- Step 1: AirYards <= 20.5
- Step 2: ydstogo > 4.0
- Step 3: AirYards <= 6.5

Result:

value = [0, 18], the leaf node contains 0 incomplete samples, 18 complete samples in the training data. So the process is predicted to be complete.