Experiment 6

Name	Ameya S. Daddikar
College I.D.	161070015
Course	Btech. Computer Engineering

Aim

To study classification of data using a Decision Tree algorithm.

Theory

Decision Tree is very very very simple model. It is widely known and used in many businesses to support decision making process and risk analysis. It is also one of legendary learning model which is heavily used in ’60–80’s to build expert systems. One of very popular expert systems which adopt decision tree (almost cs and informatics student knew about) is Mycin which developed at 1970 by Buchanan and Cohen. But, like another classic expert systems, Mycin is not fully automatic operated. At that years, human experts still needed to input hard coded rules into expert systems. After 80’s, this model has been lost popularity since it’s seems can not be extended using more sophisticated mathematics.

But now, Decision Tree learning start gaining popularity since some machine learning practitioners proved that inferior algorithm with bigger data may beats sophisticated algorithm. Beside that, it is worth to learn Decision Tree learning model at first place, before jump into more abstract models, such as, Neural Network and SVM (Support Vector Machine). By learning Decision Tree, you will have better insight how to implement basic probability theory and how to transform basic searching algorithm into machine learning algorithm.

Basically, we only need two mathematical tool to implement complete ID3 algorithms:

Entropy

It is a fundamental theorem which commonly used in information theory to measure important of information relative to its size. Let x is our training set contains positive and negative examples, then the entropy of x relative to this classification is:

$$H(x) = - \ p_+ \ log_2 \ p_+ - p_- \ log_2 \ p_-$$

Entropy Function

Information Gain

In multivariate calculus, we have learn how to use a partial derivative of each variable relative to all other variables to find local optimum. In information theory,

we used similar concept, we derive the original entropy of population to measure information gain of each attribute.
For training set x and its attribute y, the formula of Information Gain is:

$$G(x,y) = H(x) - \sum_{i \in value(y)} \frac{|\Delta y_i|}{|\Delta y|} H(y_i)$$

Decision Tree

Suppose we face with binary classification 'yes' or 'no', then we label of bit 1 for yes, and label of bit 0 for no. One of our feature's attributes is 'Outlook' (O) which has three possible values 'Sunny' (Os), 'Overcast' (Oo), and 'Rain' (Or). Then, the information gain of Outlook is:

$$G(x, Outlook) = H(x) - \frac{|Os_1 - Os_0|}{|O_1 - O_0|} H(Os) - \frac{|Oo_1 - Oo_0|}{|O_1 - O_0|} H(Oo) - \frac{|Or_1 - Or_0|}{|O_1 - O_0|} H(Or)$$

Code & Output

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load in

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_cs
import matplotlib as plt
import seaborn as sns
# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under t
he input directory

import os

DIRNAME = os.path.join('/', 'kaggle', 'input', 'titanic')

TRAIN_DATA = pd.read_csv(os.path.join(DIRNAME, 'train.csv'))
TEST_DATA  = pd.read_csv(os.path.join(DIRNAME, 'test.csv' ))
GENDER_SUBMISSIONS = pd.read_csv(os.path.join(DIRNAME, 'gender_submission.csv' ))


#print(TRAIN_DATA[:3])

# Any results you write to the current directory are saved as output.
```

```python
# TRAIN_DATA.replace({'Sex': gender_mapping}, inplace=True)
# TEST_DATA .replace({'Sex': gender_mapping}, inplace=True)

TRAIN_DATA['is_male'] = TRAIN_DATA.apply(lambda row: row.Sex == 'male', axis=1)
TEST_DATA ['is_male'] = TEST_DATA.apply(lambda row: row.Sex == 'male', axis=1)

TRAIN_DATA['Embarked_C'] = TRAIN_DATA.apply(lambda row: row['Embarked'] == 'C', axis=1)
TRAIN_DATA['Embarked_Q'] = TRAIN_DATA.apply(lambda row: row['Embarked'] == 'Q', axis=1)
TRAIN_DATA['Embarked_S'] = TRAIN_DATA.apply(lambda row: row['Embarked'] == 'S', axis=1)

TEST_DATA['Embarked_C'] = TEST_DATA.apply(lambda row: row.Embarked == 'C', axis=1)
TEST_DATA['Embarked_Q'] = TEST_DATA.apply(lambda row: row.Embarked == 'Q', axis=1)
TEST_DATA['Embarked_S'] = TEST_DATA.apply(lambda row: row.Embarked == 'S', axis=1)


default_values = {'Age': 20, 'Fare': 20}
TRAIN_DATA.fillna(value=default_values, inplace=True)
TEST_DATA.fillna(value=default_values, inplace=True)

print('Is Data Null', TRAIN_DATA.isna(), TEST_DATA.isna())

# TRAIN_DATA['Sex'].apply(lambda x: 1 if x == 'male' else 0 )
# TEST_DATA ['Sex'].apply(lambda x: 1 if x == 'male' else 0 )
#TRAIN_DATA['Embarked'].apply(lambda x: ord(x))
```

```
[418 rows x 15 columns]
* TRAIN_DATA.head()
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked  is_male  Embarked_C  \
0      0         A/5 21171   7.2500   NaN        S     True       False
1      0          PC 17599  71.2833   C85        C    False        True
2      0  STON/O2. 3101282   7.9250   NaN        S    False       False
3      0            113803  53.1000  C123        S    False       False
4      0            373450   8.0500   NaN        S     True       False
```

```
# features = ['Pclass', 'is_male', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked_C', 'Embarked_Q',
'Embarked_S']
features = ['Pclass', 'is_male', 'Age', 'Fare', 'Embarked_C', 'Embarked_Q', 'Embarked_S']



y = TRAIN_DATA['Survived']
X = TRAIN_DATA[features]

X.fillna(X.mean(), inplace=True)
print(X.dtypes)

print(X.head())
print(y.head())
print("Nans")
print(X)
```

```
Pclass          int64
is_male          bool
Age           float64
Fare          float64
Embarked_C       bool
Embarked_Q       bool
Embarked_S       bool
dtype: object
   Pclass  is_male   Age     Fare  Embarked_C  Embarked_Q  Embarked_S
0       3     True  22.0   7.2500       False       False        True
1       1    False  38.0  71.2833        True       False       False
2       3    False  26.0   7.9250       False       False        True
3       1    False  35.0  53.1000       False       False        True
4       3     True  35.0   8.0500       False       False        True
0    0
1    1
2    1
3    1
4    0
Name: Survived, dtype: int64
Nans
     Pclass  is_male   Age     Fare  Embarked_C  Embarked_Q  Embarked_S
0         3     True  22.0   7.2500       False       False        True
1         1    False  38.0  71.2833        True       False       False
2         3    False  26.0   7.9250       False       False        True
3         1    False  35.0  53.1000       False       False        True
4         3     True  35.0   8.0500       False       False        True
```

```
from sklearn import tree

dt1 = tree.DecisionTreeClassifier(min_samples_split=100, random_state=99, max_depth=5)
dt1.fit(X, y)
```

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=5,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=100,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=99, splitter='best')
```

```
dt2 = tree.DecisionTreeClassifier()
dt2.fit(X, y)
```

```
DecisionTreeClassifier(class_weight=None, criterion='gini', 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')
```

Conclusion

Thus we studied classification and applied the same on the given dataset.