Assignment: Sentiment Classification by AdaBoost and Random Forest

This dataset was chosen to contain similar numbers of positive and negative reviews. Here is our objective: **for a given review, we want to predict whether its sentiment is positive or negetive**. In this assignment, you need to establish both AdaBoost model and Random Forest model to solve this problem.

We have processed the dataset for you. Load file `sentiment_classification_processed_data.csv` to access the processed data. In the processed dataset, the column `sentiment` is our target label, with 1 means positive and -1 means negative. All the other columns except `sentiment` are our input features, and each column is corresponding to one important word.

You need to do the following tasks:

1. Split dataset into training set and testing set using an 80/20 split.
2. Generate a logistic regression model, fit the model by training set and calculate the accuracy on testing set.
3. Establish an AdaBoost model with the following setting: `n_estimators=5`, `random_state=1`. Calculate the accuracy on training set and test set.
4. Establish a Random Forest model with the following setting: `n_estimators=5`, `random_state=1`. Calculate the accuracy on training set and test set.
5. Do crossvalidation for AdaBoost. Generate 4 different AdaBoost models by setting `max_depth=2, 5, 10 and 20`. Fix `random_state=1` and `n_estimators=50` for these 4 models. Calculate the accuracy on training set and testing set for all these 4 models.
6. Do crossvalidation for Random Forest. Generate 4 Random Forest models by setting `n_estimators=5, 10, 50 and 100`. Fix `random_state=1` and `max_depth` to be default value for these 4 models. Calculate the accuracy on training set and testing set for all these 4 models.

```
import pandas as pd
data = pd.read_csv('sentiment_classification_processed_data.csv')
data.head()
```

```
   baby  one  great  love  use  would  like  easy  little  seat  ...  
picture  \
0     0    0      1     0    0      0     0     1       0     0  ...  
0
1     0    0      0     0    0      0     0     0       0     0  ...  
0
2     1    0      0     0    0      0     0     0       0     0  ...  
0
3     0    0      0     0    0      0     1     0       0     0  ...  
0
4     0    0      1     0    0      0     0     0       0     0  ...  
0
```

```
   completely  wish  buying  babies  won  tub  almost  either
sentiment
0           0     0       0       0    0    0       0       0
1
1           0     0       0       0    0    0       0       0
1
2           0     0       0       0    0    0       0       0
1
3           0     0       0       0    0    0       0       0
1
4           0     0       0       1    0    0       0       0
1

[5 rows x 194 columns]
```

Split data

- Split dataset into training set and testing set using an 80/20 split.

```
from sklearn.model_selection import train_test_split
import numpy as np


# define input features
features = data[np.delete(data.columns.values,
np.argwhere(data.columns.values=='sentiment'))]

# define output feature (label)
label = data['sentiment']

# split data into training set and testing set
train_X, test_X, train_Y, test_Y =  train_test_split(features, label,
test_size= 0.2, random_state=0 )
```

Logistic regression model

- Fit a logistic regression model by training set.
- Calculate the accuracy on testing set.

```
from sklearn import linear_model

# initialize a logistic regression model
logistic = linear_model.LogisticRegression()

# fit the logistic regression model
logistic.fit(train_X, train_Y)

# calculate the accuracy on testing set by function
linear_model.LogisticRegression.score(test_X, test_Y)
print('Logistic Regression accuracy on the training set: %.2f'
%logistic.score(test_X, test_Y))
```

```
Logistic Regression accuracy on the training set: 0.79
```

AdaBoost model

- Establish an AdaBoost model with the following setting: `n_estimators=5`, `random_state=1`.
- Calculate the accuracy on training set and test set.

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

# initialize a AdaBoost model with DecisionTreeClassifier as the
estimator, set random_state=1, n_estimators=5
adaboost = AdaBoostClassifier(DecisionTreeClassifier(),
random_state=1, n_estimators=5)

# fit the AdaBoost model
adaboost.fit(train_X, train_Y)

# calculate the accuracy on training set by function
sklearn.ensemble.AdaBoostClassifier.score(train_X, train_Y)
print ('AdaBoost accuracy on the training set: %.2f' %
adaboost.score(train_X, train_Y))

# calculate the accuracy on testing set by function
sklearn.ensemble.AdaBoostClassifier.score(test_X, test_Y)
print ('AdaBoost accuracy on the testing set : %.2f' %
adaboost.score(test_X, test_Y))
```

```
AdaBoost accuracy on the training set: 0.99
AdaBoost accuracy on the testing set : 0.71
```

```
adaboost_5 = AdaBoostClassifier(DecisionTreeClassifier(max_depth=5),
random_state=1, n_estimators=5)
adaboost_5.fit(train_X, train_Y)

print ('AdaBoost set max_depth=5, accuracy on the training set: %.2f'
%adaboost_5.score(train_X, train_Y))
print ('AdaBoost set max_depth=5, accuracy on the testing set: %.2f'
%adaboost_5.score(test_X, test_Y))
```

```
AdaBoost set max_depth=5, accuracy on the training set: 0.76
AdaBoost set max_depth=5, accuracy on the testing set: 0.75
```

Random Forest model

- Establish a Random Forest model with the following setting: `n_estimators=5`, `random_state=1`.
- Calculate the accuracy on training set and test set.

```
from sklearn.ensemble import RandomForestClassifier

# initialize a Random Forest model, set random_state=1, n_estimators=5
```

```
random_forest = RandomForestClassifier(n_estimators=5, random_state=1)

# fit the Random Forest model
random_forest.fit(train_X, train_Y)

# calculate the accuracy on training set by function
sklearn.ensemble.RandomForestClassifier.score(train_X, train_Y)
print ('Random Forest accuracy on the training set: %.2f' %
random_forest.score(train_X, train_Y))

# calculate the accuracy on testing set by function
sklearn.ensemble.RandomForestClassifier.score(train_X, train_Y)
print ('Random Forest accuracy on the testing set : %.2f' %
random_forest.score(test_X, test_Y))
```

```
Random Forest accuracy on the training set: 0.96
Random Forest accuracy on the testing set : 0.73
```

Crossvalidation for AdaBoost

- Generate 4 different AdaBoost models by setting max_depth=2, 5, 10 and 20. Fix random_state=1 and n_estimators=50 for these 4 models.
- Calculate the accuracy on training set and testing set for all these 4 models to check whether the model is overfitting.

```
# initialize and fit a AdaBoost model, set max_depth=2,
random_state=1, n_estimators=50
adaboost_2 = AdaBoostClassifier(DecisionTreeClassifier(max_depth=2),
random_state=1, n_estimators=50)
adaboost_2.fit(train_X, train_Y)

# initialize a AdaBoost model, set max_depth=5, random_state=1,
n_estimators=50
adaboost_5 = AdaBoostClassifier(DecisionTreeClassifier(max_depth=5),
random_state=1, n_estimators=50)
adaboost_5.fit(train_X, train_Y)

# initialize a AdaBoost model, set max_depth=10, random_state=1,
n_estimators=50
adaboost_10 = AdaBoostClassifier(DecisionTreeClassifier(max_depth=10),
random_state=1, n_estimators=50)
adaboost_10.fit(train_X, train_Y)

# initialize a AdaBoost model, set max_depth=20, random_state=1,
n_estimators=50
adaboost_20 = AdaBoostClassifier(DecisionTreeClassifier(max_depth=20),
random_state=1, n_estimators=50)
adaboost_20.fit(train_X, train_Y)

# calculate the accuracy on training set by function
sklearn.ensemble.AdaBoostClassifier.score(train_X, train_Y)
```

```
print ('AdaBoost max_depth=2,  n_estimators=50,  accuracy on training
set:  %.2f' % adaboost_2.score(train_X, train_Y))
print ('AdaBoost max_depth=5,  n_estimators=50,  accuracy on training
set:  %.2f' % adaboost_5.score(train_X, train_Y))
print ('AdaBoost max_depth=10, n_estimators=50,  accuracy on training
set:  %.2f' % adaboost_10.score(train_X, train_Y))
print ('AdaBoost max_depth=20, n_estimators=50,  accuracy on training
set:  %.2f' % adaboost_20.score(train_X, train_Y))

# calculate the accuracy on testing set by function
sklearn.ensemble.AdaBoostClassifier.score(train_X, train_Y)
print ('AdaBoost max_depth=2,  n_estimators=50,  accuracy on testing
set:  %.2f' % adaboost_2.score(test_X, test_Y))
print ('AdaBoost max_depth=5,  n_estimators=50,  accuracy on testing
set:  %.2f' % adaboost_5.score(test_X, test_Y))
print ('AdaBoost max_depth=10, n_estimators=50,  accuracy on testing
set:  %.2f' % adaboost_10.score(test_X, test_Y))
print ('AdaBoost max_depth=20, n_estimators=50,  accuracy on testing
set:  %.2f' % adaboost_20.score(test_X, test_Y))
```

```
AdaBoost max_depth=2,  n_estimators=50,  accuracy on training set:
0.78
AdaBoost max_depth=5,  n_estimators=50,  accuracy on training set:
0.83
AdaBoost max_depth=10, n_estimators=50,  accuracy on training set:
0.94
AdaBoost max_depth=20, n_estimators=50,  accuracy on training set:
0.99
AdaBoost max_depth=2,  n_estimators=50,  accuracy on testing set:
0.78
AdaBoost max_depth=5,  n_estimators=50,  accuracy on testing set:
0.77
AdaBoost max_depth=10, n_estimators=50,  accuracy on testing set:
0.71
AdaBoost max_depth=20, n_estimators=50,  accuracy on testing set:
0.71
```

Crossvalidation for Random Forest

- Generate 4 different Random Forest models by setting n_estimators=5, 10, 50 and 100. Fix random_state=1 and max_depth to be default value for these 4 models.
- Calculate the accuracy on training set and testing set for all these 4 models to check whether the model is overfitting.

```
# initialize and fit a Random Forest model, set n_estimators=5,
random_state=1
random_forest_5 = RandomForestClassifier(n_estimators=5,
random_state=1)
random_forest_5.fit(train_X, train_Y)

# initialize and fit a Random Forest model, set n_estimators=10,
random_state=1
random_forest_10 = RandomForestClassifier(n_estimators=10,
random_state=1)
random_forest_10.fit(train_X, train_Y)

# initialize and fit a Random Forest model, set n_estimators=50,
random_state=1
random_forest_50 = RandomForestClassifier(n_estimators=50,
random_state=1)
random_forest_50.fit(train_X, train_Y)

# initialize and fit a Random Forest model, set n_estimators=100,
random_state=1
random_forest_100 = RandomForestClassifier(n_estimators=100,
random_state=1)
random_forest_100.fit(train_X, train_Y)

# calculate the accuracy on training set by function
sklearn.ensemble.RandomForestClassifier.score(train_X, train_Y)
print ('RF n_estimators=5,   accuracy on training set:  %.2f' %
random_forest_5.score(train_X, train_Y))
print ('RF n_estimators=10,  accuracy on training set:  %.2f' %
random_forest_10.score(train_X, train_Y))
print ('RF n_estimators=50,  accuracy on training set:  %.2f' %
random_forest_50.score(train_X, train_Y))
print ('RF n_estimators=100, accuracy on training set:  %.2f' %
random_forest_100.score(train_X, train_Y))

# calculate the accuracy on testing set by function
sklearn.ensemble.RandomForestClassifier.score(train_X, train_Y)
print ('RF n_estimators=5,   accuracy on testing set:  %.2f' %
random_forest_5.score(test_X, test_Y))
print ('RF n_estimators=10,  accuracy on testing set:  %.2f' %
random_forest_10.score(test_X, test_Y))
print ('RF n_estimators=50,  accuracy on testing set:  %.2f' %
random_forest_50.score(test_X, test_Y))
print ('RF n_estimators=100, accuracy on testing set:  %.2f' %
random_forest_100.score(test_X, test_Y))
```

```
RF n_estimators=5,   accuracy on training set:  0.96
RF n_estimators=10,  accuracy on training set:  0.98
RF n_estimators=50,  accuracy on training set:  0.99
RF n_estimators=100, accuracy on training set:  0.99
RF n_estimators=5,   accuracy on testing set:  0.73
RF n_estimators=10,  accuracy on testing set:  0.75
RF n_estimators=50,  accuracy on testing set:  0.78
RF n_estimators=100, accuracy on testing set:  0.78
```