RandomForest.py

December 17, 2018

```
In [ ]: import sys
        import warnings

        if not sys.warnoptions:
            warnings.simplefilter("ignore")

        import numpy as np
        import pandas as pd
        import re
        import tensorflow as tf
        from sklearn.model_selection import train_test_split
        import matplotlib.pyplot as plt
        import random
        from sklearn.metrics import r2_score, mean_squared_error, accuracy_score, log_loss
        import matplotlib.pyplot as plt

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



        def easydatagen():
            """
            This function generates training data
            It returns:
            x_train, x_test, y_train, y_test
            """

            # Reading in the training file
            data = pd.read_json('train.json')

            # The set of different cuisines
            cuisines = data.cuisine.unique()

            # To find the different ingredients, we need to clean them up a little.
            def clean(string) :
                s = string.replace('-',' ') # read low-fat the same as low fat
```

```
    s = string.replace('&', 'and') # read & and and as the same
    s = re.sub('\((.*?)\)', '', s) # remove everythin g in brackets
    s = re.sub('\d{1,2}\%', '', s) # remove things of the form d% or dd%, where d
    s = ' '.join(s.split()) # remove extra white spaces

    return s

ing_list = data.ingredients.values.tolist()
raw_ingredients = [clean(x) for ing in ing_list for x in ing]

ingredients = sorted(set(raw_ingredients))

# build a dictionary that to each ingredient assigns its index
ingredient_index = {}
for i in range(0,len(ingredients)) :
    ingredient_index[ingredients[i]] = i

# the same for cuisines
cuisine_index = {}
for i in range(0, len(cuisines)) :
    cuisine_index[cuisines[i]] = i

def ingredients_to_vector(ings) :
    vect = np.zeros(len(ingredients))
    for ing in ings :
        vect[ingredient_index[clean(ing)]] = 1

    return vect

def cuisine_to_vector(cus) :
    vect = np.zeros(20)
    vect[cuisine_index[cus]] = 1
    return vect

vect_list = [ingredients_to_vector(ing) for ing in ing_list]
target_list = [cuisine_to_vector(cus) for cus in data.cuisine.values.tolist()]

# Define training data
X = np.c_[vect_list]
Y = np.c_[target_list]

Y_num = np.zeros((Y.shape[0]))
for i in range(Y.shape[0]):
    Y_num[i] = np.argmax(Y[i])

x_train, x_test, y_train, y_test = train_test_split(X, Y_num, test_size = 0.2)

return x_train, x_test, y_train, y_test
```

```
if __name__ == '__main__':

    x_train, x_test, y_train, y_test = easydatagen()

    """
    Next code plots 32treesAcc.png
    """

    print('Starting training:')

    testscores = []
    trainscores = []
    index = []

    for i in range(1,32):
        clf = RandomForestClassifier(n_estimators=i, max_depth=None, max_features='auto
                                 verbose=True, n_jobs=8)
        clf.fit(x_train, y_train)

        index.append(i)
        testscores.append(clf.score(x_test, y_test))
        trainscores.append(clf.score(x_train, y_train))

    plt.plot(index, testscores, '-b' , label='Testing data')
    plt.plot(index, trainscores, '-r', label='Training data')
    plt.legend(loc='center right')
    plt.title('Accuracy of random forests')
    plt.xlabel('Trees')
    plt.ylabel('Accuracy')
    plt.show()
    # Plot tells us that we can use 10 trees to get a decent score




    """
    This next code plots Max_featuresplot.png
    """
    print('Starting training:')
    testscores = []
    trainscores = []
    index = []

    """
    # Auto = sqrt(classifiers) = 81      log2 = 12
    # Use then different values as max_features
    """
```

```
for i in range(1, 200, 10):
    clf = RandomForestClassifier(n_estimators=10, max_depth=None, max_features= i,
                                 verbose=True, n_jobs=8)
    clf.fit(x_train, y_train)

    index.append(i)
    testscores.append(clf.score(x_test, y_test))
    trainscores.append(clf.score(x_train, y_train))

plt.plot(index, testscores, '-b' , label='Testing data')
plt.plot(index, trainscores, '-r', label='Training data')
plt.legend(loc='center right')
plt.title('Accuracy of random forests')
plt.xlabel('Max_features')
plt.ylabel('Accuracy')
plt.show()

# Nothing conclusive
```