|  |
| --- |
| # %load train\_model.py |
| ''' |
| train\_model.py |
| @author: jubinsoni |
| Text analysis |
| ''' |
| import numpy as np |
| import pandas as pd |
| from sklearn import metrics |
| from sklearn.naive\_bayes import MultinomialNB |
| from sklearn.feature\_extraction.text import TfidfVectorizer |
| import re, csv, sys |
|  |
| reload(sys) |
| sys.setdefaultencoding('utf8') |
|  |
| def createDict(calorie\_file): |
| calorie = pd.read\_csv(c:\users\user\document\indian1.csv) |
| d = {} |
| i = 0 |
| for foodsubcategory in calorie.FoodSubcategory: |
| d[foodsubcategory.lower()] = calorie.Calories[i] |
| i = i + 1 |
| return d |
|  |
| def search(d, searchFor): |
| df = pd.DataFrame({'food': d.keys(), 'calorie': d.values()}) |
| return df[df['food'].str.contains(searchFor)] |
|  |
| def getActualValue(searchFor): |
| try: |
| d = createDict(calorie\_file='calorie\_dataset.csv') |
| a = search(d,searchFor.lower()) |
| return np.array([a.values[0][0]]) |
| except: |
| return 1 |
|  |
| def train\_calorie\_model(data\_file): |
| train = pd.read\_csv(data\_file) |
| vectorizer = TfidfVectorizer(min\_df=1, ngram\_range=(1, 10)) |
| X\_train = vectorizer.fit\_transform(np.array(train.Food)) |
| model = MultinomialNB().fit(X\_train, np.array(train.Calories)) |
| return model, vectorizer |
|  |
| def get\_score\_of\_calorie(text): |
| actual = getActualValue(text) |
| predicted = float(get\_calorie(text)[0]) |
| #accuracy\_score = metrics.accuracy\_score(predicted,actual) |
| accuracy\_score = predicted/actual |
| return accuracy\_score |
|  |
| def get\_calorie(text): |
| data\_file = 'calorie\_dataset.csv' |
| model, vectorizer = train\_calorie\_model(data\_file) |
| test = vectorizer.transform([text]) |
| return model.predict(test) |
|  |
| if \_\_name\_\_ == '\_\_main\_\_': |
| food = "Rava" #Pizza, Stawberry, Burger, Fries, Biriyani, Dosa, Egg, etc... |
| d = createDict(calorie\_file='calorie\_dataset.csv') |
| print food,"has %s calories." % get\_calorie(food.lower()) |
| print "Accuracy Score: %f" % get\_score\_of\_calorie(food.lower()) |
| print "Other healthy options: \n %s" % (search(d, food.lower())) |
| #print getActualValue(food) |