*# CS677 assignment 5  
# by Zuowen Tang*import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn  
from sklearn.metrics import roc\_auc\_score  
  
  
*#2. for each category, count (and print) the number of food items in that category.  
# Take the 20 most numerous categories and display the results in a bar chart  
# (save bar chart in a pdf file ”items per category.pdf”)*def Task2():  
 df = pd.read\_csv('food.csv')  
 print("Task 2:")  
 print(df['Category'].value\_counts())  
 category = df['Category'].value\_counts()[:20]  
 print('\n', category)  
 category.plot(kind='barh')  
 plt.tight\_layout()  
 plt.savefig("items\_per\_category.pdf", format="pdf")  
 plt.show()  
  
  
*#3. choose two food categories randomly (let’s call them A and B).  
# Pick one vitamin category and one mineral category (for example, vitamin B12 and zinc).  
# Use scatterplot to plot values. Color points from category A as green and category B as red  
# (e.g. vitamin B12 and zinc). Save results in a pdf file ”vitamin mineral 2 categories.pdf”*def Task3():  
 df = pd.read\_csv('food.csv')  
 cheese = df[df['Category'] == 'CHEESE']  
 potatoes = df[df['Category'] == 'POTATOES']  
 x1 = list(cheese['Data.Vitamins.Vitamin E'])  
 y1 = list(cheese['Data.Major Minerals.Calcium'])  
 x2 = list(potatoes['Data.Vitamins.Vitamin E'])  
 y2 = list(potatoes['Data.Major Minerals.Calcium'])  
 plt.scatter(x1, y1, color='green', label='Cheese')  
 plt.scatter(x2, y2, color='red', label='Potatoes')  
 plt.xlabel('Vitamin E')  
 plt.ylabel('Calcium')  
 plt.title('Vitamin E vs Calcium')  
 plt.savefig("vitamin\_mineral\_2\_categories.pdf", format="pdf")  
 print("\nTask 3:")  
 plt.show()  
 *# task 4:* print("\nTask 4:")  
 print("Examine the two plots. Any interesting observations?")  
 print("The green dots, which represents cheese, contain varies calcium and vitamin E; "  
 "while the red dots, represents potatoes, have very little calcium, yet have varies vitamin E.")  
  
  
def getTrainTest(num=0.8):  
 pd.set\_option('display.max\_columns', None)  
 pd.set\_option('display.max\_rows', None)  
 pd.set\_option('display.width', 1000)  
 df = pd.read\_csv('food.csv')  
  
 df = df[['Category', 'Data.Major Minerals.Calcium', 'Data.Major Minerals.Iron', 'Data.Vitamins.Vitamin E',  
 'Data.Vitamins.Vitamin K']].copy()  
 df = df[(df['Category'] == 'BEEF') | (df['Category'] == 'CHEESE') | (df['Category'] == 'SOUP') | (  
 df['Category'] == 'EGG') | (df['Category'] == 'CHICKEN')]  
 m1 = 'Data.Major Minerals.Calcium'  
 df['labels'] = np.where(df[m1] >= 250, 1, 0)  
  
 train = df.sample(frac=num, random\_state=25)  
 test = df.drop(train.index)  
 train\_0 = train[train['labels'] == 0]  
 train\_0 = train\_0.drop(['labels'], axis=1)  
 train\_1 = train[train['labels'] == 1]  
 train\_1 = train\_1.drop(['labels'], axis=1)  
 return train\_0, train\_1, test, train  
  
  
def getName():  
 m1 = 'Data.Major Minerals.Calcium'  
 m2 = 'Data.Major Minerals.Iron'  
 v1 = 'Data.Vitamins.Vitamin K'  
 v2 = 'Data.Vitamins.Vitamin E'  
 return m1, m2, v1, v2  
  
  
def Task5():  
 train\_0, train\_1, test, train = getTrainTest()  
 print("\nTask 5:")  
 seaborn.pairplot(train\_0)  
 plt.savefig('zero\_food.pdf', format="pdf")  
 plt.show()  
 seaborn.pairplot(train\_1)  
 plt.savefig('one\_food.pdf', format="pdf")  
 plt.show()  
 seaborn.pairplot(train, hue='labels')  
 plt.savefig('food.pdf', format="pdf")  
 plt.show()  
  
  
def Task6\_9():  
 *# task 6:  
 # if m2 <= 10 and m1 <= 200 and v1 <= 25  
 # y = 0  
 # else: y = 1  
  
 # task 7,8:* train\_0, train\_1, test, train = getTrainTest()  
 m1, m2, v1, v2 = getName()  
 test['Y\_pred'] = np.where((test[m2] <= 10) & (test[m1] <= 200) & (test[v1] <= 25), 0, 1)  
 conditions = [  
 (test['labels'] == 1) & (test['Y\_pred'] == 1),  
 (test['labels'] == 0) & (test['Y\_pred'] == 1),  
 (test['labels'] == 0) & (test['Y\_pred'] == 0),  
 (test['labels'] == 1) & (test['Y\_pred'] == 0)  
 ]  
 values = ['TP', 'FP', 'TN', 'FN']  
 test['compare'] = np.select(conditions, values)  
 print("\nTask 7,8:")  
 print(test.head(), '\n')  
 print(test['compare'].value\_counts())  
  
 *# Task 9* FP, TN, TP, FN = 3, 165, 13, 0  
 TPR = TP / (TP + FN)  
 TNR = TN / (TN + FP)  
 ACC = (TP + TN) / (TP + TN + FP + FN)  
 d = {'TR': [TP], 'FP': [FP], 'TN': [TN], 'FN': [FN], 'Accuracy': [ACC], 'TPR': [TPR], 'TNR': [TNR]}  
 df = pd.DataFrame(data=d)  
 print("\nTask 9:")  
 print(df)  
  
 *# Task 10* y\_true = list(test['labels'])  
 y\_pred = list(test['Y\_pred'])  
 auc = roc\_auc\_score(y\_true, y\_pred)  
 print("\nTask 10:")  
 print("Does your simple classifier give you higher accuracy on identifying 0 or 1 than 50% (”coin” flipping)?")  
 print("Yes, the AUC is", auc)  
  
  
def Task11():  
 train\_0, train\_1, test, train = getTrainTest(1)  
 m1, m2, v1, v2 = getName()  
 df = pd.DataFrame(columns=['Class', 'μ(v1)', 'σ(v1)', 'μ(v2)', 'σ(v2)', 'μ(m1)', 'σ(m1)', 'μ(m2)', 'σ(m2)'])  
  
 for i in range(3):  
 if i == 0:  
 className = '0'  
 data = train\_0  
 elif i == 1:  
 className = '1'  
 data = train\_1  
 elif i == 2 :  
 className = 'all'  
 data = train  
 avg\_v1 = data[v1].mean()  
 std\_v1 = data[v1].std()  
 avg\_v2 = data[v2].mean()  
 std\_v2 = data[v2].std()  
 avg\_m1 = data[m1].mean()  
 std\_m1 = data[m1].std()  
 avg\_m2 = data[m2].mean()  
 std\_m2 = data[m2].std()  
 df.loc[len(df.index)] = [className, avg\_v1, std\_v1, avg\_v2, std\_v2, avg\_m1, std\_m1, avg\_m2, std\_m2]  
  
 print("\nTask 11:")  
 print(df)  
 print("\nTask 12:")  
 print("Examine your tables. Are there any obvious patterns in the distribution of "  
 "vitamin/mineral values in each class?")  
 print("Class 0 almost always contains less value than class 1.")  
  
  
def Task13():  
 pd.set\_option('display.max\_columns', None)  
 pd.set\_option('display.max\_rows', None)  
 pd.set\_option('display.width', 1000)  
 df = pd.read\_csv("food.csv")  
 df = df[df["Category"] == 'LAMB']  
 df2 = df.drop(list(df)[0:33], axis=1)  
 matrix = df2.corr()  
 print('\nTask 13:')  
 print(matrix)  
 orderedMatrix = df2.corr().unstack().sort\_values().drop\_duplicates()  
 print('\n', orderedMatrix)  
  
  
def Driver():  
 Task2()  
 Task3()  
 Task5()  
 Task6\_9()  
 Task11()  
 Task13()  
  
  
Driver()