# Examine labels:

import pandas as pd  
import matplotlib.pyplot as plt  
  
  
def plot(yearNum):  
 df = pd.read\_csv('TMO\_weekly\_label.csv')  
 year = df[df['Year'] == yearNum]  
 green = year[year['Label'] == 'g']  
 red = year[year['Label'] == 'r']  
  
 x = green['Mean Return']  
 y = green['Volatility']  
 plt.scatter(x, y, c='green')  
  
 x = red['Mean Return']  
 y = red['Volatility']  
 plt.scatter(x, y, c='red')  
  
 plt.title(str(yearNum))  
 plt.xlabel('Avg. of Daily Returns')  
 plt.ylabel('Volatility')  
 plt.grid(True)  
 plt.savefig(str(yearNum)+'.png')  
 plt.show()  
  
  
plot(2021)  
plot(2022)

# Separate points:

import pandas as pd  
import matplotlib.pyplot as plt  
  
*# 1. take year 1 and examine the plot of your labels.  
# Construct a reduced dataset by removing some green and red points  
# so that you can draw a line separating the points.  
# Compute the equation of such a line (many solutiuons are possible)*def Task1(yearNum):  
 df = pd.read\_csv('TMO\_weekly\_label.csv')  
 year = df[df['Year'] == yearNum]  
 green = year[year['Label'] == 'g']  
 red = year[year['Label'] == 'r']  
  
 if yearNum == 2021:  
 filter = green[green['Mean Return'] >= 0]  
 y = filter['Volatility']  
 x = filter['Mean Return']  
 plt.scatter(x, y, c='green')  
  
 filter = red[red['Mean Return'] <= 0]  
 y = filter['Volatility']  
 x = filter['Mean Return']  
 plt.scatter(x, y, c='red')  
  
 if yearNum == 2022:  
 x = green['Mean Return']  
 y = green['Volatility']  
 plt.scatter(x, y, c='green')  
  
 x = red['Mean Return']  
 y = red['Volatility']  
 plt.scatter(x, y, c='red')  
  
 x1, y1 = [0.05, 0.05], [0.3, 3]  
 plt.plot(x1, y1, marker='o')  
  
 plt.title(str(yearNum))  
 plt.xlabel('Avg. of Daily Returns')  
 plt.ylabel('Volatility')  
 plt.grid(True)  
 plt.savefig(str(yearNum)+'\*.png')  
 plt.show()  
  
  
*# 2. Take this line and use it to assign labels for year 2*def Task2():  
 df = pd.read\_csv('TMO\_weekly\_label.csv')  
 year = df[df['Year'] == 2022]  
 filter = year[year['Mean Return'] >= 0]  
 filter = filter.replace('r','g')  
  
 filter2 = year[year['Mean Return'] <= 0]  
 filter2 = filter2.replace('g', 'r')  
  
 x = filter['Mean Return']  
 y = filter['Volatility']  
 plt.scatter(x, y, c='green')  
  
 x = filter2['Mean Return']  
 y = filter2['Volatility']  
 plt.scatter(x, y, c='red')  
 plt.title('Year 2 (Assign labels based on last year)')  
 plt.xlabel('Avg. of Daily Returns')  
 plt.ylabel('Volatility')  
 plt.grid(True)  
 plt.savefig('2022\_new.png')  
 plt.show()

# Retail Price Data:

import pandas as pd  
import matplotlib.pyplot as plt  
import numpy as np  
from math import log10, log  
import math  
  
  
def getPrice():  
 df = pd.read\_csv('online\_retail.csv')  
 df.drop(df[df['UnitPrice'] < 1].index, inplace=True)  
 df2 = df['UnitPrice'].astype(str).str[:1]  
 df2 = df2.astype(int)  
 return df2  
  
  
def benford(n):  
 return log10(n+1) - log10(n)  
  
  
def model1(p):  
 temp = int(len(p) / 9)  
 res = []  
 for i in range(1, 10):  
 res = res + [i] \* temp  
 return res  
  
  
def model2(p):  
 x = [i for i in range(1, 10)]  
 num = [benford(i) for i in x]  
 res = []  
 for i in range(1, 10):  
 temp = len(p) \* num[i - 1]  
 res = res + [i] \* int(temp)  
 return res  
  
  
*# 1. plot 3 histograms for the frequencies for real distribution,  
# equal-weight and Bernford (for each digit)*def Task1():  
 *# Real Distribution* p = getPrice()  
 p = list(p)  
  
 *# Equal-weight Distribution  
 # Benford Distribution* data = [0]\*3  
 data[0], data[1], data[2] = p, model1(p), model2(p)  
  
 colors = ['purple', 'limegreen', 'tomato']  
 labels = ['Real Distribution', 'Equal-weight', 'Benford']  
 plt.hist(data, bins=np.arange(0.5, 10.5), histtype='bar', color=colors, label=labels)  
 plt.legend(prop={'size': 12})  
 plt.xlim(0.5, 9.5)  
 plt.title('Histogram for Task 1')  
 plt.xlabel('Digits')  
 plt.ylabel('Frequency')  
 plt.savefig('hist1.png')  
 plt.show()  
  
  
*# 2. plot 3 histograms for the relative errors for Models 1 and 2 (for each digit)*def Task2():  
 p = getPrice()  
 measured\_val = list(p.value\_counts())  
 p0 = list(p)  
  
 *# Models1* p2 = int(len(p0)/9)  
 true\_val = [p2]\*9  
  
 abs\_err, relative\_err = [0]\*9, [0]\*9  
 for i in range(9):  
 abs\_err[i] = abs(true\_val[i] - measured\_val[i])  
 relative\_err[i] = abs\_err[i]/true\_val[i]  
  
 data = [0]\*2  
 data[0] = relative\_err  
  
 true\_val, abs\_err, relative\_err = [0] \* 9, [0] \* 9, [0] \* 9  
 *# Models 2* p3 = model2(p)  
 for i in range(1, 10):  
 true\_val[i-1] = p3.count(i)  
  
 for i in range(9):  
 abs\_err[i] = abs(true\_val[i] - measured\_val[i])  
 relative\_err[i] = abs\_err[i] / true\_val[i]  
 data[1] = relative\_err  
  
  
 labels = [i for i in range(1, 10)]  
 x = np.arange(len(labels))  
 width = 0.25  
 plt.bar(x - width / 2, data[0], width, label='Model 1')  
 plt.bar(x + width / 2, data[1], width, label='Model 2')  
 plt.xlabel('Digit')  
 plt.ylabel('Relative Errors')  
 plt.title('Histogram for Task 2')  
 plt.xticks(x, labels=labels)  
 plt.legend()  
 plt.savefig('hist2.png')  
 plt.show()  
  
  
*# 3. compute RMSE (root mean squared error) for model 1, 2.  
# Which model is closer to the real distribution?*def Task3():  
 p = getPrice()  
 actual\_val = list(p.value\_counts())  
 p0 = list(p)  
 predicted\_val = [0]\*2  
  
 p2 = int(len(p0) / 9)  
 predicted\_val[0] = [p2] \* 9  
  
 p3 = model2(p)  
 predicted\_val[1] = [0] \* 9  
 for i in range(1, 10):  
 predicted\_val[1][i - 1] = p3.count(i)  
  
 for i in range(2):  
 MSE = np.square(np.subtract(actual\_val, predicted\_val[i])).mean()  
 RMSE = math.sqrt(MSE)  
 print('RMSE of Model ' + str(i+1) + ' is', RMSE)  
  
  
*# 4. take 3 countires of your choice:  
# one from Asia, one from Europe and one from the Middle East.  
# For each of these countries do the following:*def Task4():  
 df = pd.read\_csv('online\_retail.csv')  
 df.drop(df[df['UnitPrice'] < 1].index, inplace=True)  
 df['UnitPrice'] = df['UnitPrice'].astype(str).str[:1]  
 df['UnitPrice'] = df['UnitPrice'].astype(int)  
 Israel = df[df['Country'] == 'Israel']  
 Japan = df[df['Country'] == 'Japan']  
 Germany = df[df['Country'] == 'Germany']  
 Israel = Israel['UnitPrice']  
 Japan = Japan['UnitPrice']  
 Germany = Germany['UnitPrice']  
  
 data, data\_Israel, data\_Japan, data\_Germany = [0]\*3, [0]\*3, [0]\*3, [0]\*3  
 *#(a) compute F, P and π* data\_Israel[0], data\_Israel[1], data\_Israel[2] = list(Israel), model1(Israel), model2(Israel)  
 data\_Japan[0], data\_Japan[1], data\_Japan[2] = list(Japan), model1(Japan), model2(Japan)  
 data\_Germany[0], data\_Germany[1], data\_Germany[2] = list(Germany), model1(Germany), model2(Germany)  
 data[0], data[1], data[2] = data\_Israel, data\_Japan, data\_Germany  
 f, p, pi = [0]\*9,[0]\*9,[0]\*9  
 for i in range(3):  
 if i == 0:  
 country = 'Israel'  
 if i == 1:  
 country = 'Japan'  
 if i == 2:  
 country = 'Germany'  
 for j in range(9):  
 f[j] = data[i][0].count(j + 1)  
 p[j] = data[i][1].count(j + 1)  
 pi[j] = data[i][2].count(j + 1)  
  
 print('F, P, and pi for ' + country + ' is (from 1 - 9)')  
 print('F:', f)  
 print('P:', p)  
 print('pi:', pi)  
  
 *# (b) using RMSE as a ”distance” metric, for which of these chosen three countries  
 # is the distribution ”closest” to equal weight P?* MSE = np.square(np.subtract(f, p)).mean()  
 RMSE = math.sqrt(MSE)  
 print('RMSE of equal-weight in ' + country + ' is', RMSE)  
 print('\*' \* 30, '\n')  
  
  
  
 for i in range(3):  
 colors = ['purple', 'limegreen', 'tomato']  
 labels = ['Real Distribution', 'Equal-weight', 'Benford']  
 plt.hist(data[i], bins=np.arange(0.5, 10.5), histtype='bar', color=colors, label=labels)  
 plt.legend(prop={'size': 12})  
 plt.xlim(0.5, 9.5)  
  
 if i == 0:  
 country = 'Israel'  
 if i == 1:  
 country = 'Japan'  
 if i == 2:  
 country = 'Germany'  
  
 plt.title('Histogram for ' + country)  
 plt.xlabel('Digits')  
 plt.ylabel('Frequency')  
 plt.savefig(str(i) + '.png')  
 plt.show()  
  
  
def Driver():  
 Task1()  
 Task2()  
 Task3()  
 Task4()  
  
  
Driver()