Data analysis and visualizations

Name of student

Name of professor

University

Course

Date

This analysis was based on the county of ten counts of baskets that were filled with fruits and then the concentration of vitamins and minerals selected from this selection of baskets. The also step of this task involves the prediction of the potential random fruit baskets. The analysis is done on Python and Jupyter Notebook. The first step is to load the dataset as below:

#Load the dataset and read it on Jupyter

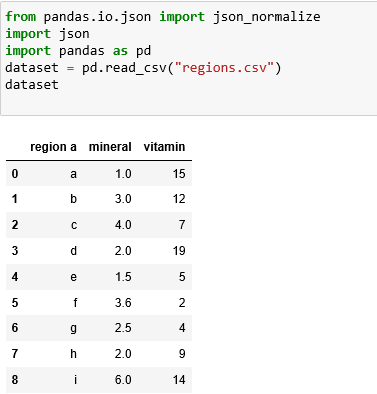
from pandas.io.json import json\_normalize

import json

import pandas as pd

dataset = pd.read\_csv("regions.csv")

dataset



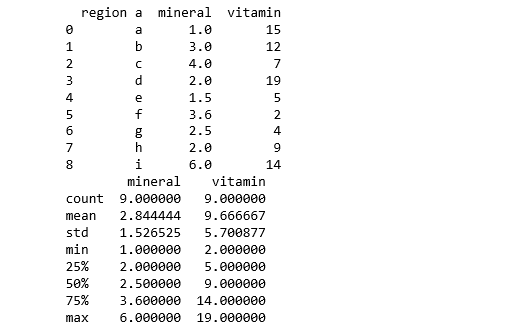
Getting a summary of the dataset looks like below:

#Get the summary of the data

df = pd.DataFrame(dataset)

print(df)

print(df.describe())



The summary of the datasets show the mean, std deviation, minimum 50th percentile, 75th percentile and maximum value. The next step is to visualize the dataset on Jupyter Notebook.

# creating some arrays relevant for the plot

region =np.array(['a','b','c','d','e','f','g','h','i'])

mineral =np.array([1,3,4,2,1.5,3.6,2.5,2,6])

x\_pos = np.arange(len(region))

CTEs = [mineral]

#Building the plot

fig, ax = plt.subplots()

ax.bar(x\_pos, CTEs, align='center', alpha=0.5)

ax.set\_ylabel('Region')

ax.set\_xticks(x\_pos)

ax.set\_xticklabels(the\_hour)

ax.set\_title('Minerals')

ax.yaxis.grid(True)

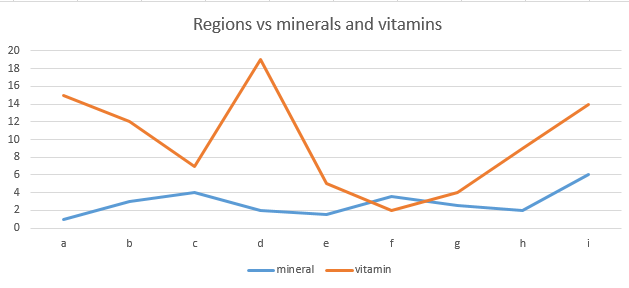
# Saving result as an image

plt.tight\_layout()

plt.savefig('bar-chart.png')

plt.show()

The graph fenerated below indicates that minerals were the highest count in region d and I as below.



In the next step, an analysis was done to predict the potential vitamin content in random sample of the 2.4 samples and then the resulting output shown below. This indicates a negative correlation covariance between these two data variables. Meaning that the change in a basket does not change the vitamin content or the mineral content of the fruits.

#Probabilistic predictions here:

From sklearn.linear\_model import LinearRegression

from sklearn.datasets import make\_regression

X, y = make\_regression(n\_samples=200, n\_features=2, noise=0.1)

model = LinearRegression()

model.fit(X, y)

# create a new data instance

Xnew = [[2,4]]

# the predict the outcome

ynew = model.predict(Xnew)

# show prediction

print("X=%s, Predicted=%s" % (Xnew[0], ynew[0]))

