import seaborn as sns

import matplotlib.pyplot as plt

import os

import pandas as pd

import numpy as np

from sklearn.preprocessing import LabelEncoder

from sklearn.cluster import KMeans

filename = "StudentsPerformance.csv"

score\_df = pd.read\_csv(filename)

score\_df.head()

score\_df.rename(columns={"race/ethnicity":"ethnicity","parental level of education":"parent\_education","math score":"math","reading score":"reading","writing score":"writing","test preparation course":"pre"},inplace=True)

score\_df.head()

score\_df.dtypes

fig, ax = plt.subplots()

fig.subplots\_adjust(hspace=0.8, wspace=0.8, left = 0.2, right = 1.5)

for idx in range(3):

plt.subplot(1,3, idx+1)

gender\_df = score\_df.groupby("gender")[list(score\_df.columns[-3:])[idx]].describe()

sns.barplot(gender\_df.index,gender\_df.loc[:,"mean"].values)

plt.ylabel("score")

plt.title(list(score\_df.columns[-3:])[idx])

plt.show()

fig, ax = plt.subplots()

fig.subplots\_adjust(hspace=0.8, wspace=0.8, left = 0.2, right = 1.5)

for idx in range(3):

plt.subplot(1,3, idx+1)

ethn\_df = score\_df.groupby("ethnicity")[list(score\_df.columns[-3:])[idx]].mean()

sns.barplot(x=ethn\_df.index, y = ethn\_df.values, palette = "Greens")

plt.xlabel("Group")

plt.ylabel("mean score")

plt.xticks(rotation=90)

plt.title(list(score\_df.columns[-3:])[idx])

plt.show()

for item in score\_df.columns[-3:]:

sns.boxplot(x=score\_df["pre"], y=score\_df[item])

plt.title(item+" vs pre test", loc="left")

plt.show()

for item in score\_df.columns[-3:]:

sns.boxplot(x=score\_df["lunch"], y=score\_df[item])

plt.title(item+" vs lunch", loc="left")

plt.show()

labelencoder = LabelEncoder()

train\_df = score\_df.copy()

train\_df["parent\_education"] = labelencoder.fit\_transform(train\_df["parent\_education"])

train\_df["pre"] = labelencoder.fit\_transform(train\_df["pre"])

train\_df["lunch"] = labelencoder.fit\_transform(train\_df["lunch"])

train\_df.head()

kmeans\_dis =list()

for idx in range(2, 25):

kmeans = KMeans(init = "k-means++", n\_clusters = idx, n\_init = 20)

kmeans.fit\_transform(train\_df.iloc[:, 2:])

kmeans\_dis,append(kmeans.inertia\_)

plt.plot(list(range(2,25)), kmeans\_dis, marker = "o")

plt.xlabel("Number of clusters")

plt.ylabel("Summation of distance")

plt.show()

kmeans = KMeans(init = "k-means++", n\_clusters = 8)

kmeans.fit\_transform(train\_df.iloc[:, 2:])

kmeans\_label = kmeans.labels\_

score\_df["classification"] = kmeans\_label

score\_df.head(10)

class\_df = score\_df.group("classification")[score\_df,columns[-4:-1]].mean()

class\_df

ind = np.arange(8)

width = 0.35

fig, ax = plt.subplots()

rects1 = ax.bar(ind - width/2, class\_df.math, width, label='Math')

rects2 = ax.bar(ind, class\_df.reading, width, label='Reading')

rects3 = ax.bar(ind + width/2, class\_df.writing, width, label='Writing')

ax.set\_xlabel('Classification')

ax.set\_ylabel('Scores')

ax.set\_xticks(ind)

ax.legend()

plt.show()

class\_df["total\_ave\_score"] = (class\_df.math + class\_df.reading + class\_df.writing)/3

rank = class\_df["total\_ave\_score"].sort\_values(ascending = Flase)

rank.index

rank

def plot\_pie\_chart(column):

fig, ax = plt.subplots(figsize=(20,16))

color = ["orange","lightblue","green","yellow","red","pink","brown","gray"]

for idx in range(8):

plt.subplot(3, 3, idx+1)

num = "class"+ str(idx)

num = score\_df[score\_df["classification"]==rank.index[idx]]

percentage\_of\_parent\_edu = num[column].value\_counts()

percentage\_of\_parent\_edu.sort\_index()

label = percentage\_of\_parent\_edu.index

value = percentage\_of\_parent\_edu.values

plt.pie(value, labels = label, autopct = "%1.1f%%",

startangle=90, radius = 4, colors = color[:len(label)])

plt.axis("equal")

plt.title("Rank "+str(idx))

plt.show()

plot\_pie\_chart("parent\_education")

def plot\_bar\_chart(column):

fig, ax = plt.subplots(figsize (8,6))

index\_dict = dict()

width = 0.35

ind = np.arange(8)

for idx in range(8):

num = "class"+ str(idx)

num = score\_df[score\_df["classification"]==rank.index[idx]]

percentage\_of\_column = num[colomn].value\_counts()

percentage\_of\_column = percentage\_of\_column.sort\_index()

for key in percentage\_of\_column.index:

if key not in index\_dict.keys():

index\_dict[key] = []

index\_dict[key].append(percentage\_of\_column[key])

else:

index\_dict[key].append(percentage\_of\_column[key])

percentage\_of\_column = score\_df[score\_df["classification"]==rank[4]][column].value\_counts().sort\_index()

for i in range(len(percentage\_of\_column.index)):

rects = ax.bar(ind - width/(i+1),index\_dict[percentage\_of\_column.index[i]],

width, label=percentage\_of\_column.index[i])

ax.set\_xlabel('Rank')

ax.set\_ylabel('# of students')

ax.set\_title("Percentage of " + column)

ax.set\_xticks(ind)

ax.legend()

plt.show()

plot\_bar\_chart("pre")

plot\_bar\_chart("lunch")

plot\_bar\_chart("gender")