Project 9

In addition to answering the bolded questions on Coursera, also attach your notebook, both as `.ipynb` and `.html`.

Please follow the instructions below (e.g., setting random state values) to ensure that your answers match the solutions.

In this assignment, we will be using PennGrader, a Python package built by a former TA for autograding Python notebooks. PennGrader was developed to provide students with instant feedback on their answer. You can submit your answer and know whether it's right or wrong instantly. We then record your most recent answer in our backend database. You will have 100 attempts per test case, which should be more than sufficient.

NOTE: Please remember to remove the

```
raise notImplementedError
```

after your implementation, otherwise the cell will not compile.

You currently work for a company that offers a subscription-based music streaming service. After an unsuccessful round of marketing efforts with low return on investments, your company is exploring other strategies to promote its services. Currently, its advertising strategy has been to bid on a set of keywords that someone on the marketing team put together, but the team has realized that they could improve on this strategy by refining the keywords based on different customer personas. Therefore, the company has asked you to help it identify a set of user segments so that the ads can be more customized.

For this task, you have been provided with the Music data which contains sociodemographic and music preference data on 4,914 users and your goal is to identify clusters of users based on this information.

Column	Description
Age	Age of the user
Gender	Gender of the user
Employment status	Employment status of the user
Annual income	Annual income of the user in USD
Usage per month	Average usage per month of the user measured in minutes
Top genre	The genre of music that is most streamed by the user
Num of days active	The number of days in the last 365 days that the user used the service

Getting Set Up

Meet our old friend - PennGrader! Fill in the cell below with your PennID and then run the following cell to initialize the grader.

Warning: Please make sure you only have one copy of the student notebook in your directory in Codio upon submission. The autograder looks for the variable `STUDENT_ID` across all notebooks, so if there is a duplicate notebook, it will fail.

In [1]:
```
#PLEASE ENSURE YOUR STUDENT_ID IS ENTERED AS AN INT (NOT A STRING). IF NOT, TH
E AUTOGRADER WON'T KNOW WHO
#TO ASSIGN POINTS TO YOU IN OUR BACKEND

STUDENT_ID = 49731093                    # YOUR 8-DIGIT PENNID GOES HERE
STUDENT_NAME = "Newman Ilgenfritz"      # YOUR FULL NAME GOES HERE
```

In [2]:
```
import penngrader.grader

grader = penngrader.grader.PennGrader(homework_id = 'ESE542_Online_Spring_2021
_HW9', student_id = STUDENT_ID)
```

In [3]:
```
# Let's import the relevant Python packages here
# Feel free to import any other packages for this project

#Data Wrangling
import pandas as pd
import numpy as np
```

Part A: Data Cleaning

Throughout this course, you have mostly been provided with relatively clean datasets but in the real world, data is usually a lot messier. This could be caused by data logging errors, bugs in the pipeline or even erroneous human input in surveys. Some common approaches of dealing with these observations are by dropping them, replacing the erroneous values with the mean/median, winsorizing, etc. Here, we will just drop them.

1. Load the Music data and drop any rows with `NaN/null` values. Assign the dataframe to the variable `music` .

In [4]:
```
music = pd.read_csv('music.csv')

print('music.shape before dropna = ', music.shape, '\n')

# Drop all rows with NaN/null inplace

music.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False)


# df.reset_index(drop=True)
music.reset_index(drop=True)
print('music.head: ')
print(music.head(), '\n')
print('music.tail: ')
print(music.tail(), '\n')
print('music.shape = ', music.shape, '\n')

#raise notImplementedError
```

```
music.shape before dropna =  (4914, 7)

music.head:
   age  gender   employment_status  annual_income  usage_per_month top_genre
\
0   20    Male  Employed full-time         115000              847      Rock
1   22    Male  Employed full-time          42100              256      Rock
2   18    Male  Employed full-time         114750             1232   Country
3   19  Female  Employed full-time         118250             2310      Rock
4   33    Male  Employed full-time         111450              568       Pop

   num_of_days_active
0                 210
1                 222
2                 161
3                 176
4                 185

music.tail:
      age  gender   employment_status  annual_income  usage_per_month  \
4909   31  Female          Unemployed          17150             1291
4910   41    Male  Employed full-time          19150              905
4911   37    Male  Employed full-time          39000              535
4912   43  Female             Student         138800               13
4913   44    Male  Employed full-time          27000             1184

     top_genre  num_of_days_active
4909      Rock                 220
4910       Pop                 166
4911      Rock                 203
4912      Rock                 126
4913       Pop                 230

music.shape =  (4914, 7)
```

In [5]:
```
grader.grade(test_case_id = 'test_read_music', answer = music.shape)
```

```
Correct! You earned 1/1 points. You are a star!

Your submission has been successfully recorded in the gradebook.
```

1. Plot bar charts and histograms of the data to visualize the distributions. Does anything seem unusual? Good to think about: can you encapsulate plotting data into a function?

    - Are there more male or female users? Store your answer in `male_or_female` ('M'/'F').
    - Which genre is the most popular among users? Store your answer in `pop_genre` .
    - How many users are students? Store you answer in `student_num` .

In [6]:
```
from matplotlib import pyplot as plt
%matplotlib inline

#music.plot.bar()
#plt.title("Magnitude of each data column:")
#plt.xlabel("Column type")
#plt.ylabel("Magnitude of each")
#plt.show()
#plt.close()

#music.plot.hist()
#plt.show()
#plt.close()

mpt = music.pivot_table(index=['gender'], aggfunc='size')
print('mpt: ')
print(mpt, '\n')

mpt = music.pivot_table(index=['top_genre'], aggfunc='size')
print('mpt: ')
print(mpt, '\n')

mpt = music.pivot_table(index=['employment_status'], aggfunc='size')
print('mpt: ')
print(mpt, '\n')
```

```
mpt:
gender
Female               2350
Male                 2482
Other                  22
Prefer not to say      60
dtype: int64

mpt:
top_genre
Classical     721
Country       556
Hip Hop        99
Indie         289
Jazz          265
Pop          2149
Rock          835
dtype: int64

mpt:
employment_status
Employed full-time    3037
Employed part-time     513
Student               1000
Unemployed             364
dtype: int64
```

In [7]:
```
male_or_female ='M'
```

In [8]:
```
grader.grade(test_case_id = 'test_visual1', answer = male_or_female)
```

```
Correct! You earned 1/1 points. You are a star!

Your submission has been successfully recorded in the gradebook.
```

In [9]:
```
pop_genre = 'pop'
```

In [10]:
```
grader.grade(test_case_id = 'test_visual2', answer = pop_genre)
```

```
Correct! You earned 1/1 points. You are a star!

Your submission has been successfully recorded in the gradebook.
```

In [11]:
```
student_num = 1000
```

In [12]:
```
grader.grade(test_case_id = 'test_visual3', answer = student_num)
```

```
Correct! You earned 1/1 points. You are a star!

Your submission has been successfully recorded in the gradebook.
```

1. Take a closer look at ‘age’ and ‘num of days active’. Does anything look peculiar? Store the number of invalid 'age' in `bad_age_num` , invalid number of 'num of days active' in `bad_active_num` . Drop these rows and also assign the new dataframe to the variable `music` . *Hint*: Recall the values that age and number of days active can take on.

In [13]:
```
# Enter your code here
mpt = music.pivot_table(index=['age'], aggfunc='size')
print('mpt: ')
print(mpt, '\n')

baNum = [-29, -27, -21, -10, -1]
bad_age_num = len(baNum) # 5
print('bad_age_num: ',bad_age_num, '\n')
```

```
mpt:
age
-29      1
-27      1
-21      1
-10      1
-1       1
 16    133
 17    128
 18    144
 19    146
 20    139
 21    144
 22    147
 23    152
 24    142
 25    140
 26    129
 27    131
 28    142
 29    124
 30    165
 31    154
 32    164
 33    187
 34    160
 35    158
 36    163
 37    130
 38    167
 39    160
 40    145
 41    118
 42    118
 43    117
 44    119
 45    112
 46    112
 47    124
 48    107
 49    106
 50    107
 51      4
 52      4
 53      6
 54      5
 55      3
 56      4
 57      8
 58      5
 59      5
 60      8
 61      6
 62      2
 63      6
 64      7
 65      2
```

```
dtype: int64

bad_age_num:  5
```

In [14]:
```
grader.grade(test_case_id = 'test_drop1', answer = bad_age_num)
```

```
Correct! You earned 2/2 points. You are a star!

Your submission has been successfully recorded in the gradebook.
```

In [15]:
```
# Enter your code here
mpt = music.pivot_table(index=['num_of_days_active'], aggfunc='size')
print('mpt: ')
#print(mpt.head(50), '\n')
print(mpt.tail(5), '\n')

baActNum = [380, 392]
bad_active_num = len(baActNum) # 2
print('bad_active_num: ',bad_active_num, '\n')
```

```
mpt:
num_of_days_active
329    1
346    1
350    1
380    1
392    1
dtype: int64

bad_active_num:  2
```

In [16]:
```
grader.grade(test_case_id = 'test_drop2', answer = bad_active_num)
```

```
Correct! You earned 2/2 points. You are a star!

Your submission has been successfully recorded in the gradebook.
```

In [17]:
```
# Drop rows with invalid entries in-place

music = music[music.age > 1]
mpt = music.pivot_table(index=['age'], aggfunc='size')
print('mpt: ')
print(mpt, '\n')

music = music[music.num_of_days_active <= 365]
mpt = music.pivot_table(index=['num_of_days_active'], aggfunc='size')
print('mpt: ')
print(mpt, '\n')

print('\nmusic.shape = ', music.shape, '\n')
```

```
mpt:
age
16    133
17    128
18    144
19    146
20    139
21    144
22    147
23    152
24    142
25    140
26    129
27    131
28    142
29    124
30    165
31    154
32    164
33    187
34    160
35    158
36    163
37    130
38    167
39    160
40    145
41    118
42    118
43    117
44    119
45    112
46    112
47    124
48    107
49    106
50    107
51      4
52      4
53      6
54      5
55      3
56      4
57      8
58      5
59      5
60      8
61      6
62      2
63      6
64      7
65      2
dtype: int64

mpt:
num_of_days_active
31     1
```

```
34     1
41     1
42     1
45     1
      ..
323    1
326    1
329    1
346    1
350    1
Length: 262, dtype: int64


music.shape =  (4907, 7)
```

In [18]:
```
grader.grade(test_case_id = 'test_drop3', answer = music.shape)
```

```
Correct! You earned 2/2 points. You are a star!

Your submission has been successfully recorded in the gradebook.
```

Part B: Clustering

Use K-means clustering to find a set of user groups using the following features as inputs: ‘age’, ‘annual income’, ‘usage per month’ and ‘number of days active’.

1. Standardize the data. *Hint*: `sklearn.preprocessing.scale` may be helpful.

In [19]:
```
from sklearn import preprocessing
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import scale


musicQuant = music.drop(['gender', 'employment_status', 'top_genre'], axis=1)
print('musicQuant.head: ')
print(musicQuant.head(), '\n')

musicQuantScld = preprocessing.scale(musicQuant, axis=0, with_mean=True, with_
std=True, copy=True)
print('musicQuantScld: ')
print(musicQuantScld, '\n')


X_scaled = musicQuantScld
```

```
musicQuant.head:
   age  annual_income  usage_per_month  num_of_days_active
0   20         115000              847                 210
1   22          42100              256                 222
2   18         114750             1232                 161
3   19         118250             2310                 176
4   33         111450              568                 185

musicQuantScld:
[[-1.26120736  1.70825627 -0.32086458  0.62887233]
 [-1.06361331  0.02875876 -0.94875035  0.89592062]
 [-1.45880141  1.70249667  0.08816421 -0.46157486]
 ...
 [ 0.41834209 -0.0426602  -0.65233727  0.47309416]
 [ 1.01112424  2.25656958 -1.20691657 -1.24046571]
 [ 1.10992127 -0.31912069  0.03716841  1.07395282]]
```

In [20]:
```
grader.grade(test_case_id = 'test_scale', answer = X_scaled)
```

```
Correct! You earned 2/2 points. You are a star!

Your submission has been successfully recorded in the gradebook.
```

1. Calculate the sum of squared distances of observations to their closest cluster center for $K \in [1, 10]$ and add these values to a list called `ssd`. Use the default hyperparameters and set `random_state=42`. Store the minimum value of ssd in `min_ssd`. Comment on your thoughts: shall we choose the model with minimum ssd?

Hint: You may want to write a for-loop and refer to the `sklearn.cluster.KMeans` (https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html) documentation (read through the Attributes section).

In [21]:
```
clusters = 3
maxIters =  300
kmeansArgs = {"init": "random", "n_init": 10, "max_iter": maxIters, "random_st
ate": 42}


ssd = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, **kmeansArgs)
    kmeans.fit(X_scaled)
    ssd.append(kmeans.inertia_)

print('ssd: ')
print(ssd, '\n')

min_ssd = min(ssd)
print('min_ssd: ',min_ssd , '\n')
```

```
ssd:
[19628.000000000004, 15279.939975664749, 12077.981652729, 10202.025406603878,
8889.815844988263, 8021.102051116282, 7353.001733920524, 6775.933069916614, 6
325.373776598275, 6014.832512563668]

min_ssd:  6014.832512563668
```

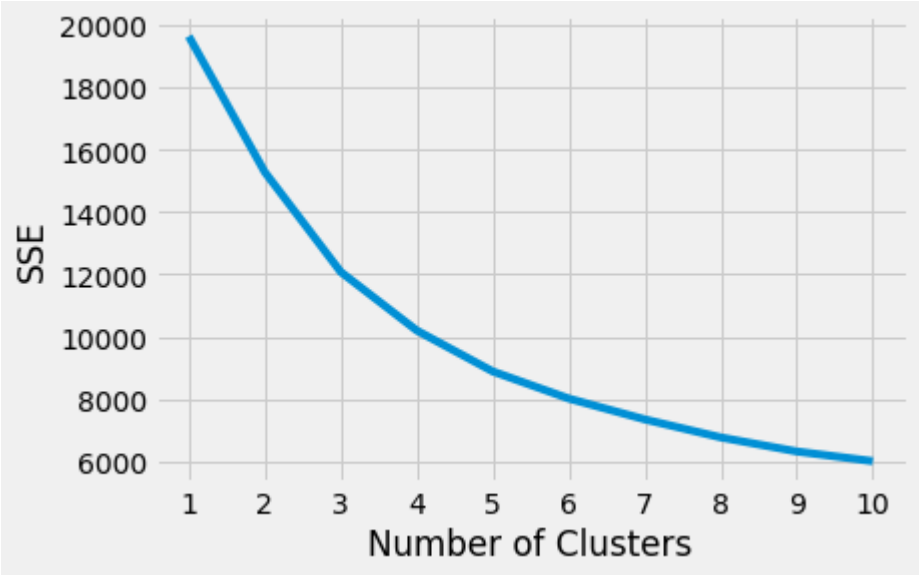
In [22]:
```
grader.grade(test_case_id = 'test_min_ssd', answer = min_ssd)
```

```
Correct! You earned 2/2 points. You are a star!

Your submission has been successfully recorded in the gradebook.
```

1. Plot the values in `ssd` against the number of clusters, K for $K \in [1, 10]$.

In [23]:
```
x = np.arange(1, 11)
y = ssd
plt.style.use("fivethirtyeight")
plt.plot(x, y)
plt.xticks(range(1, 11))
plt.xlabel("Number of Clusters")
plt.ylabel("SSE")
plt.show()
plt.close()
```



1. Based on this plot, what is the best number of clusters to set using the Elbow Method? The Elbow Method is the point where diminishing returns are no longer worth the additional cost. It will be the point before which the later points taper off or increase by very little. (More information on the Elbow Method can be found here (https://en.wikipedia.org/wiki/Elbow_method_(clustering)#:~:text=In\%20cluster\%20analysis\%2C\%20the\%2 Assign this value to the variable `optimal_K` (There might be multiple optimal k values, just enter one of those).

In [25]:
```
#import kneed
#from kneed import KneeLocator

#kl = KneeLocator( range(1, 11), ssd, curve="convex", direction="decreasing" )


# optimal_K = kl.elbow
optimal_K = 4
```

In [26]:
```
grader.grade(test_case_id = 'test_best_K', answer = optimal_K)
```

```
Correct! You earned 2/2 points. You are a star!

Your submission has been successfully recorded in the gradebook.
```

1. Retrain the K-means clustering algorithm with `n_clusters=optimal_K` . This will be your final model. Name your model `best_kmeans` and set `random_state=42` .

In [27]:
```
clusters = optimal_K
maxIters =  300
kmeansArgs = {"init": "random", "n_init": 10, "max_iter": maxIters, "random_st
ate": 42}
kmeans = KMeans(n_clusters=clusters, **kmeansArgs)
#kmeans.fit(X_scaled)

best_kmeans = kmeans.fit(X_scaled)

print('best_kmeans.cluster_centers_ :')
print(best_kmeans.cluster_centers_, '\n')
```

```
best_kmeans.cluster_centers_ :
[[ 9.54499771e-01 -3.96147495e-01 -4.38869616e-01 -1.05320102e-01]
 [-8.88934157e-01 -4.13304281e-01 -7.75288958e-01 -1.44681625e-03]
 [ 4.47434260e-01  1.94395051e+00  2.30392334e-01 -1.58390715e-02]
 [-7.08953631e-02 -4.08179560e-01  1.17531767e+00  1.05911249e-01]]
```

In [28]:
```
grader.grade(test_case_id = 'test_best_model', answer = (optimal_K, best_kmean
s.cluster_centers_))
```

```
Correct! You earned 2/2 points. You are a star!

Your submission has been successfully recorded in the gradebook.
```

Part C: Visualizing

As a data scientist, it is important to be able to translate your findings to colleagues who may not have the same level of technical knowledge as you do; visualizations help a lot! Therefore, you have decided to plot the clusters. However, since there are four dimensions that you want to plot – age, annual income, usage per month and number of days active – you will need to utilize dimensionality reduction techniques to be able to plot this on a 2-D plane.

1. Train another K-means model using the following features as inputs: `age` , `annual income` , `usage per month` and `number of days active` . Fit your model on `X_scaled` , name your predicted result as `pred` . Use `n_clusters=2` and `random_state=42` .

In [ ]:
```
# Enter your code here

raise notImplementedError
```

In [ ]:
```
grader.grade(test_case_id = 'test_kmeans_plot', answer = pred)
```

1. Find the first **two** principal components of the scaled data and save the new data to variable `X_reduced`. Name your model `pca`.

Hint: Use `sklearn.decomposition.PCA` with `random_state=42`.

In [ ]:
```
pca =

raise notImplementedError
```

In [ ]:
```
grader.grade(test_case_id = 'test_pca', answer = (pca.n_features_, pca.n_components_))
```

1. Plot a scatterplot with the 1^{st} principal component on the x-axis and the 2^{nd} principal component on the y-axis. Color each point by the cluster that it is in from (1). Comment on yout observations.

Hint: Check out `seaborn.FacetGrid`. To use it, you will need your principle components and predicted clusters in one dataframe.

In [ ]:
```
# Enter your code here

raise notImplementedError
```