MCIS6273 Data Mining (Prof. Maull) / Fall 2021 / HW2

This assignment is worth up to 20 POINTS to your grade total if you complete it on time.

Points Possible	Due Date	Time Commitment (estimated)
20	Wednesday, October 20 @ Midnight	*up to* 24 hours

- **GRADING:** Grading will be aligned with the completeness of the objectives.
- **INDEPENDENT WORK:** Copying, cheating, plagiarism and academic dishonesty *are not tolerated* by University or course policy. Please see the syllabus for the full departmental and University statement on the academic code of honor.

OBJECTIVES

- Learn about the new data economy through Ocean Protocol and how Blockchain is fueling innovation in the data space
- Continue practicing exploratory data analysis and visualization
- Perform a clustering analysis using k-means

WHAT TO TURN IN

You are being encouraged to turn the assignment in using the provided Jupyter Notebook. To do so, make a directory in your Lab environment called `homework/hw2`. Put all of your files in that directory.

Then zip that directory, rename it with your name as the first part of the filename (e.g. `maull_hw2_files.tar.gz`), then download it to your local machine, then upload the `.tar.gz` to Blackboard.

If you do not know how to do this, please ask, or visit one of the many tutorials out there on the basics of using in Linux.

If you choose not to use the provided notebook, you will still need to turn in a `.ipynb` Jupyter Notebook and corresponding files according to the instructions in this homework.

ASSIGNMENT TASKS

(20%) Learn about the new data economy through Ocean Protocol and how Blockchain is fueling innovation in the data space

You have no doubt heard the phrase "Data is the new oil", which goes back to 2006, when British data scientist and mathematician Clive Humby first coined the phrase.

For an interesing chronology of the phrase over the past 15 years or so, please read the follow Medium.com post:

- "Data is the New Oil" — A Ludicrous Proposition by Michael Haupt

You can decide for yourself, after some contemplation, whether the "privatization" of data is right or wrong, and what one might do about it if they choose to.

Nonetheless, the parallels to oil, and the oil boom at the turn of the 20th century – which profoundly changed our relationship with energy and quite literally fueled innovation on a scale that humanity has never seen – are interesting. Data, however, is not a finite resource, and the "ownership" of data is quite different than that of oil (however controversial that may be). What should be obvious is that producing data and "enriching it" (adding "value") are two different things, and "value" is a term that requires a context.

Nonetheless, we are at an interesting point in the big data, data mining and data science conversation about what to do with and how to treat all the data now being produce, particularly as the kind of data that is becoming

interesting to companies is increasingly personal (i.e. personal health data) and quite possibly being put to use in contexts we have no transparency or control over.

Blockchain technologies are rapidly become the *de facto* platforms for data-related transactions, and interesting work is being done to use these distributed digital ledgers as data exchange platforms. Even more work is being done to bring "data ownership" to a new level on digital marketplaces and the "Internet of Value" or IoV (see also this 2017 writeup on the IoV by Ripple) building the "new data economy".

You will be listening to a podcast on this subject involving the Ocean Protocol and its founder Bruce Pon. You will listen the almost 43 minute podcast from The Crypto Conversation Podcast on Brave New Coin.

- "Data is the new oil - Ocean Protocol is building the data economy" - October 28, 2020 Podcast with Bruce Pon;
- download the mp3 file directly.

§ Listen to the entire podcast and answer the questions below:

1. What was the original idea that Pon tried in 2013?
2. Who (or what) did Pon say was the "consumer" of Blockchain and big data? How does this relate to "data mining"?
3. Who does Pon say are the gatekeepers of the current "data economy"? How does Ocean Protocol address this?
4. How does Pon suggest data might be "securitized" over Ocean Protocol? Use the example he gives directly.
5. What are the four landscape users (or targets) of Ocean Protocol?
6. What example does he give of a liquid dataset that might have tremendous value in a variety of current and future contexts relating to our world?
7. What does Pon predict the crypto marketplace will look like in 10 years?
8. After consuming this podcast, please list your reaction to it in a short paragraph of at a minimum of three **complete** sentences. Address what you already know about crypto and the data economy as well as the things that you learned that were new knowledge. You may also address whether you agree or disagree with Pon's positions, especially how it relates to Ocean Protocol and cryptocurrencies, in general, becoming a great equalizer. You can also include any personal experiences you have with the technology.

(25%) Continue practicing exploratory data analysis and visualization

In the last HW we explore some of the basic features of Pandas with graphic and data selection. This time we're going to go a but deeper into Pandas ans learn about MultIndices and grouping data in interesting and useful ways.

Power weightlifting (powerlifting) is an international sport that invites advanced amateurs and professionals alike. Fortunately, there are datasets for the multitude of powerlifting competitions around the world, and they are openly available for curious data scientists like ourselves who would like to ask interesting questions and find interesting relationships in the data. Whether you're into the sport or not, I think there are a variety of interesting phenomenon in the data that make it both tractable and interesting from just a data perspective.

DATA

OpenPowerLifting.org is a large set of data for a multitude of data related to powerlifting competitions around the world. The core data live at the following open source repository on gitlab.com/openpowelifting.

For the curious, there are a number of analyses that have already been performed on the data in a number of interesting ways. Please visit this page to further fill your intrigue.

Though the full dataset is available to us and will be used in the next part of the assignment, we want to get a little practice getting data from websites that require some HTML parsing and navigation. Unlike the last time where we used APIs to get data, we're going to build a dataset en mass from the CSV data on Gitlab. Remarkably, while this technique may seem antiquated, you will find many datasets are just sitting on servers as text files that

will require something similar to be accomplished. We're unfortunately not yet in a data environment where the most interesting data you want is easily obtained or accessible behind APIs. Often the resources to do so are beyond the capabilities of the data providers, though things are getting better each year with new tools and data access platforms.

We're going to use Python and the Beautiful Soup library to build a random dataset of just 2019 data by directly navigating the HTML of the Gitlab repository. It will be noted that Gitlab does have an API, and it would be the preferred mechanism if we were to do this exercise with APIs like the last assignment.

One of the things that we will learn from the data is that the majority of it are interesting over several dimensions. There are the years of competition, the sex of the competitors, the age the competitors, country of origin, among other things. With denser data like these, we want to understand some of the underlying groupings for easier access to the data. For example, one might want to understand how groupings by year and age bear out on the data to explore questions like "Has the number of competitors over 40 increased over the years?" This might be an interesting question to ask to explore if powerlifters continue to compete as they age since the sport is very difficult on one's body and requires intense continuous training to stay competitive.

Some questions like these are also very useful to explore visually, so we'll dive into a few more graphical techniques to get at these answers and more. We're going to end up with a DataFrame that will group our data by year, age class and sex, so we can see some of the interesting annual trends along each of these dimensions within the last two decades.

§ BUILD THE DATASET

We've learned CSV is common file format for data and we will be working the files large text files in Gitlab to do the work we need. The task is to explore the repository and build up a dataset of 15 random lifting meets from 2019 using BeautifulSoup and the tools in Pandas to put these datasets together.

This technique is often known as "crawling" and is consider by some to be a flagrant violation of good web etiquette. However, the technique is still often the only way to obtain data en mass from a single source. If this were an FTP server, the same pattern could be applied and would not be considered unusual to do so. Of course, you must use this with caution, as it can result in IP throttling and IP blocking, so please use it within the licensing terms of both the data and website you are obtaining data from. Good web citizens restore trust in providers and administrators alike, so throttling yourself after your own requests with code like time.sleep(2) (which will pause your code for 2 seconds), will show that you can behave responsibly.

Once you have loaded all the files, please re-index the rows using the Dataframe.reset_index() method.

§ FILTER AND EXPLORE THE DATA

Let's first get a feel for the data and filter it down. One of the main difficulties in dealing with large user-contributed data sets like these are *data consistency* and *data quality*. *Data consistency* refers to how data is represented over time. We can see how this becomes an issue when we look at the `Division` column of the dataset. We can see with a relatively untrained eye to the data, that something is very wrong with the consistency — there are over 1300 Division designations! When you look at it more closely, there are groupings that overlap. For example, you will see `Masters 45-49` and `Masters 40-49` when you perform a `.value_counts()` on the `Divisions` column of the data (see supplemental notebook). What is the difference between these two since they obviously overlap?

Filter the data down to a smaller subset of the data using `dropna()`. Specifically, drop all rows of data missing `Age` and `Division` data (e.g. they have `NaN` values). You will be able to do this by passing a `subset=` parameter into `dropna()`.

You should now have fewer than 200K data points, leading us to the second problem of *data quality*. When we look at the original data file, we have over 400K data points, yet after filtering for missing values, we end up with over 50% reduction in the data! This is actually a reasonable preservation of the original data set given that we `NaN` values from two different columns. In general, it isn't a bad idea when working with your data to develop an understanding of the holes in it. What if we reduced our data 90% just on `Age` indicating it was not recorded consistently over time? This would indeed limit what we could ask of the data in analyses requiring age data. Many of the tools in Pandas will ignore `NaN` data, but some required you to send specific instructions to the tool on what to do. Better to get ahead of things now.

In your notebook **you must include the following** to be considered a correct answer:

- correct use of `dropna()`,
- output of the original dimensions of the data using `DataFrame.shape` or something similar that shows the original and new dimensions of the data.

§ CLEAN THE DATA

One of the major issues with any data set when importing into Pandas is the that Pandas tries to infer the data types so that when you compare data you are comparing data that *can* be compared (i.e. you cannot compare strings with integers). One area that you'll need to be mindful of is with dates. For example, if you perform a `df.Date.head()` the data is of type `object`. We want it to be `Datetime` so we can benefit from the many useful features in Pandas to manipulate dates.

Doing this is straightforward with the `pandas.datetime()` method. Your notebook must show:

- use of `pandas.datetime` to convert the `Date` column of your data to `datetime` (hint: use `loc` to set the data),
- that the column has changed by using `DataFrame.dtype` or alternatively use `DataFrame.head()` which will show the data type in its output.

NOTE: you may need to use the `errors` parameter of `to_datetime` to handle any issues you may encounter.

§ GROUP THE DATA

Pandas provides superior capabilities to slice and group data. We would like to build a dataset that is composed of all remaining data from 1999 to 2018 that restricts age to those 21 and older.

Like the last homework, doing this is simple with `Dataframe.query()`. You will notice that a query over the `Date` column can be done naturally by just saying `Date > a_year` since it is a `datetime` object. Pandas does the inferential magic for you!

Next add to your query by grouping the data by `Date`, `AgeClass` and `Sex`. You will need to use the `DataFrame.groupby()` method to accomplish this. **A hint for grouping by year**: if you do not restrict the data in the `Date` column Pandas will naturally group by the full date meaning that each *day* will be grouped leading to the wrong result. You can convert a date to a year for the purposes of grouping by year using `dt.to_period('Y')`. Please see the documentation for `Series.dt.to_period()`.

You will pass to the first parameter of `DataFrame.groupby()` the list of the grouping in order of grouping, outer group first. Thus, `groupby(['Sex', 'AgeClass])` will return the MultiIndex DataFrame with `Sex` as the outermost group and `AgeClass` the inner group. See the supplemental notebook for some more clues.

Your notebook must show:

- use of `query()` to restrict your data to competitors 21 and older in competitions from 1999 to 2018,
- use of `groupby` showing the `mean()` values for each column with groups being (in order) `Date (year)`, `AgeClass` and `Sex`.

§ VISUALIZE

Now that we have the data segmented the way we'd like, let's visualize it in some interesting way.

With powerlifting there are a number of ways to express the *strength* of a competitor. There is *raw* strength, meaning how much total weight was lifted on a given lift, and there is *relative* strength. It is not fair to compare the raw lift of a 100lb 16 year old teenage to that of a 35 year old 300lb adult.

The 35 year old might lift ten times the weight yet the 16 year old may be *relatively* stronger, but how would we compare their *relative* strengths? Many competitors will be able to lift between 2 and 7 times their body weight depending on the lift, so we might expect a 100lb powerlifter to perhaps perform a 200lb bench press and maybe a 300lb squat, both impressive for their weight. To deal with comparing *strength* across age and weight variables a number of methods have been developed to create fair and accurate measures of *relative strength*. The OpenPowelifting dataset includes three such measures: *Wilks*, *McCulloch* and *Glossbrenner*, which give a numeric assignment of relative strength which factor age and weight into the computation. Exploring the details of each of these methods is beyond the scope of this homework, but the curious can learn more on the variety of sites which calculate these statistics.

We will restrict our interest to the *Glossbrenner* score, which takes into account age and weight to compute a normalized weight value. Consider three competitors, all 29 year olds, with one male and one female weighting 131.84lbs and the last male weighting 263.67 pounds. Assume they all lift 639.33 pounds total. The *Glossbrenner* score takes into account the age and weights and produces a relative score with the following:

competitor	sex	age (lbs)	weight (lbs)	lift (lbs)	score	γ-coefficient	γ_{age}-coefficient
1	F	29	131.84	639.33	287.187	0.9903	1
2	M	29	131.84	639.33	242.3095	0.83555	1
3	M	29	263.67	639.33	159.906	0.5514	1
4	M	49	263.67	639.33	218.951	0.5514	1.113
5	F	49	263.67	639.33	177.9754	0.67835	1.113

The *Glossbrenner* score is in the *score* column and the γ-coefficient is the constant calculated by the method. The γ_{age}-coefficient is computed constant which the method factors in for the relative impact age has on the competitor. Thus, the *Glossbrenner* score Γ is:

$$\Gamma(age, sex) = \gamma\text{-coefficient}_{age,sex} \times \gamma_{age} \times weight$$

If you perform the score calculation on the Openpowerlifting data, you'll notice the some values are off by a small amount and this is likely due to the *Glossbrenner* constants used, which have varied over time.

Now that we have that out of the way, let's visualize some data. Specifically, we'd like to plot the *Glossbrenner* score for the last 20 years over time. Are the scores going up, down or staying the same? One could expect any of these scenarios to occur, so let's dive in.

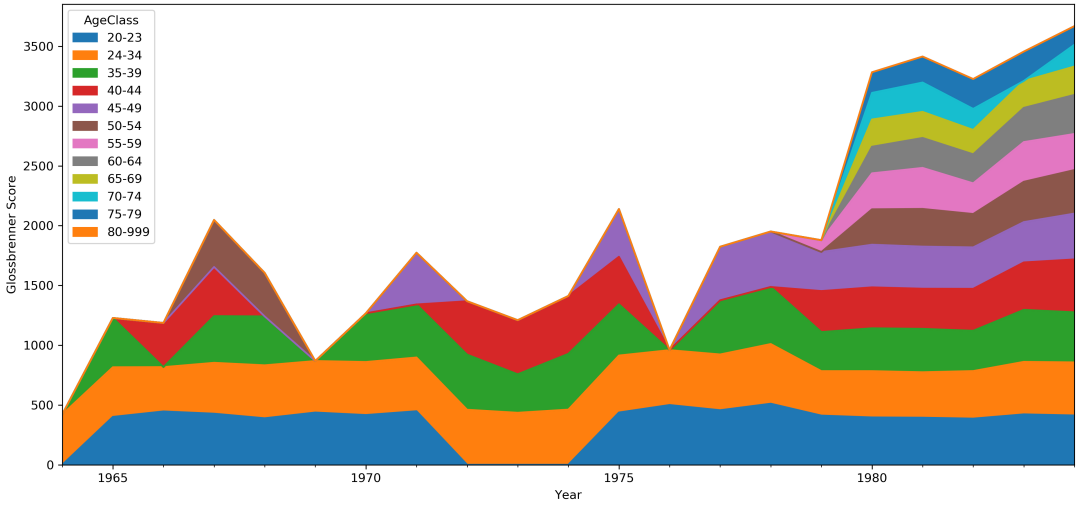
What we want to produce are two *area* plots of the annual *mean Glossbrenner* score from 1999 to 2018 for all age groups, one plot for males and the other for females as putting them all on one graph would most certainly be information overload. To do this we will need to slice the data in a way that makes a multi-index grouped by year, age group and sex. You will effectively use the data from the prior part and make a visual of it.

Your area plot code will be invoked by:

```
DataFrame.plot.area()
```

You may optionally pass in the `figsize=(15,7)` (or whatever dimensions you'd like) to stretch the data out a bit so you can visually see what is going on, since the legend may get in the way of viewing the data.

Your plot will look something like this:



Please see the `DataFrame.plot.area()` method for full information on the area plots.

A final important note: You will need to use `droplevel()` and `unstack()` in order to prepare your DataFrame for final presentation. Basically, you'll need to drop the `Sex` level of your index (level 2) and immediately before you plot the area plot you will use `unstack()`.

Your notebook must show:

- an area plot showing the male data grouped by age and year, that is the x-axis will show the year and the y-axis the *Glossbrenner* score,
- an area plot showing the female data grouped by age and year, that is the x-axis will show the year and the y-axis the *Glossbrenner* score.

You must also answer the following questions in your notebook:

- What's the general trend you see in the area plots? Your answer can be in one or two sentences.

(50%) Perform a clustering analysis using k-means

The simplicity and power of k-means algorithm makes it one of the best to start with when performing *unsupervised learning* — that is the class labels of your data are not known *a priori* and you that will not be training the algorithm on labeled data. While this is a powerful and oft useful technique, use it with care as the initial conditions of the algorithm do not guarantee a global maximum and as such, running the algorithm with a number of initialization points will produce better and more reliable results.

Continuing with our OpenPowerlifting data, we're going to do some exploratory data analysis to examine this dataset in some interesting ways using unsupervised learning, namely clustering. The original dataset has over 1 million data points, but in order to get a good idea of what's in it, we will not need to go back through the entire dataset, and in fact, we will restrict the focus of our energy on just the last 2 decades from 1999.

REMEMBER TO MAKE SURE TO SHOW ALL YOUR WORK IN THE NOTEBOOK SO YOU CAN RECEIVE PARTIAL CREDIT WHERE APPROPRIATE!

§ PREPARE FOR CLUSTERING

You will need to complete part 1 of this homework to filter the data to the necessary subset for this part. As we talked about in lecture, the subset of features will just be the following:

```
features = [
    'Sex',
    'Age',
```

```
    'BodyweightKg',
    'Best3SquatKg',
    'Best3BenchKg',
    'Best3DeadliftKg',
    'TotalKg',
    'Date'
]
```

Final preparation for clustering will require you to turn all of *categorical* variables into *numeric* one's. One way to do this from directly within Pandas is to use `Pandas.get_dummies(your_dataframe)`. You can also study the `sklearn.preprocessing.OrdinalEncoder()` which will do something very similar. Either way, with the reduced set of features above, the only categorical variable will be `Sex` as all the others should already be numerical features.

In your notebook, you should show:

- clearly how many features are now in your dataframe?

§ PERFORM SILHOUETTE ANALYSIS

In class we talked about the fact that the k number of clusters needs to be determined *a priori* — that is you will need to know how many clusters beforehand to run the algorithm. To find the optimal k, we will use a method called the *silhouette score*.

Adapt the following code to compute the silhouette scores on *only* the dataset filtered by the features from the prior step.

```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

Sum_of_squared_distances = []
K = range(2, 15)
for k in K:
    km = KMeans(n_clusters=k, n_init=20)
    km = km.fit(YOUR_OPENPOWERLIFTING_DATAFRAME_WITH_DUMMY_VARS)
    Sum_of_squared_distances.append(km.inertia_)

    silh_score = silhouette_score(YOUR_OPENPOWERLIFTING_DATAFRAME_WITH_DUMMY_VARS, km.labels_)
    print("k = {} | silhouette_score = {}".format(k, silh_score))
```

The largest score is typically the k you go with. If $k = 2$ is your largest score, we will ignore and use the next best score since 2 clusters is not usually an interesting number of clusters when dealing with a large set of data points.

Your notebook must show and answer the following:

- What is the optimal k according the silhouette score?
- What else is interesting about the scores?

§ CLUSTER INTERPRETATION

Now that you have clusters and optimal cluster, let's find out the characteristics of the features that dominate them.

Note that the k-means algorithm returns the cluster centers for each cluster, hence in that center each feature value is the *representative feature value* for that cluster. For example, the `TotalKg` would be the representative `TotalKg` for that cluster.

Using the optimal cluster size from the silhouette score in the prior section, please use adapt the following code to determine the cluster characteristics.

```
optimal_k = THE_OPTIMAL_SILH_K

km = KMeans(n_clusters=optimal_k, n_init=150)
```

```
km = km.fit(YOUR_OPENPOWERLIFTING_DATAFRAME_WITH_DUMMY_VARS)

for i in range(0, optimal_k):
    l = list(zip(YOUR_OPENPOWERLIFTING_DATAFRAME_WITH_DUMMY_VARS.columns, \
                km.cluster_centers_[i]))
    l.sort(key=lambda x: x[1], reverse=True)

    print('CLUSTER : {}\n'.format(i))
    for attr, val in l[:]:
      print('\t{} : {}\n'.format(attr, val))
```

Your notebook must show and answer the following:

- for each cluster, describe in real words what the cluster centers are telling you about the representative of that cluster. For example, your answer might look like: "for cluster 1, the representative for that cluster is a 24.7 year old female, with an average `Best3SquatKg` of 121 and a `TotalKg` of 721",
- show the output of the cluster centers above.

NOTE: The order of the features in `km.cluster_centers_` are the same order as they exist in the DataFrame.