MCIS6273 Data Mining (Prof. Maull) / Fall 2025 / HW1

Points Possible	Due Date	Time Commitment (estimated)
20	Monday September 15 @ Midnight	*up to* 20 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

- Perform basic data engineering on the light pollution dataset.
- Complete the online assessment
- BONUS: Normalize column names in working dataset.

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/hw0`. Put all of your files in that directory. Then zip or tar that directory, rename it with your name as the first part of the filename (e.g. `maull_hw0_files.zip`, `maull_hw0_files.tar.gz`), then download it to your local machine, then upload the `.zip` 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 zip 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

(80%) Perform basic data engineering on the light pollution dataset.

Continuing with our GaN dataset from the first assignment, we will now perform some data engineering.

As has been discussed in the slide content on data engineering most often we are handed datasets which are less than perfect. We might also have to deal with a more essential activity once it is perfect: determining if the dataset is sufficient to carry out the analysis we expect.

There are a few important things you may have noticed about the dataset from the first assignment:

1. the data files are separated; while this is sometimes desirable, often it is more desirable to have data in a single file *when possible*,
2. there are some categorical fields that are not numeric, but could be which might improve the outcomes of algorithms that do not accept non-numeric data,
3. there are some fields that need combining for easier access to date/time convenience functions in Pandas,
4. there are some complex fields which contain data that are may be better stored in there own field.

You will want to study a sample CSV file yourself to determine other issues which might be worthy of consideration.

For our data engineering we want to broadly complete the following:

1. combine all the GaN files into a single file and thus a single Pandas DataFrame,
2. convert specific data fields to numeric fields using Pandas,
3. combine date and time fields for easier time series evaluation,
4. split data out of specific fields and into new simpler fields,
5. use Pandas to understand the underlying distrubution of certain data.

Part of the assignment will require you to understand the following table for the Bortle Scale for assessing dark sky magnitude:

Bortle Class	SQM Value
1	>21.9
2	21.9-21.50
3	21.49-21.30
4	21.29-20.80
4.5	20.79-20.10
5	20.09-19.10
6+	<19.10

§ Task: Use the file of URLs from the first assignment to load the data from 2014-2024 and store it all into a single file.

Use Pandas `read_csv()` to load *all* the data into a single DataFrame, then store that DataFrame into a single file named `2014_to_2024_gan_data.csv`.

Note: You must `drop_duplicates()` **and** `reset_index()` before save the file!

§ Task: Convert categorical fields to numeric ones.

You might have noticed that the `CloudCover` column has 4 possible categorical values. If you execute `df['CloudCover'].value_counts()` (where `df` is the 2014-2024 GaN DataFrame), you will see "clear", "1/4 sky", "1/2 of sky" and "over 1/2 of sky" as the values.

We will convert these to numeric values using the following guidance:

- "clear" → 0
- "1/4 of sky" → 0.25
- "1/2 of sky" → 0.50
- "over 1/2 of sky" → 0.75

To do this, you will want to study `DataFrame.apply()` and you may optionally find `DataFrame.assign()` useful.

However, you choose to do it, the DataFrame should contain a new column called `CloudCoverPct` with the converted values.

§ Task: Convert `Constellation` to numeric binary values (binarize the column).

You will convert the 15 `Constellation` values to columns which contain a binary value. For example, `Leo` is present in over 30k data points. Thus a `1` will be present in the rows where `Leo` appeared and a `0` everywhere else.

To accomplish this, you will find `Pandas.get_dummies()` to be superior at accomplishing this goal in a single line of code. You must prefix the column name with `"const"` – doing so is easy with the `prefix` parameter of `get_dummies()`. For example, the constellation `Leo` would become a binary valued column `const_Leo`. Merge all the new columns into the 2014-2024 dataset.

§ Task: Create 5 binary columns `loc_urban`, `loc_suburban`, `loc_rural`, `loc_remote` and `loc_unknown`.

There is a tremedous amount of information in the `LocationComment` column – more than we have time to work on, but because it appears the field is an open response field, various data appear there (including useless and unwanted data).

I have provided a function for you to apply called `assign_location()`. The helper notebook on Github will show you the way.

Use the function in the context of the following operations:

(a) convert all `LocationComment` to a single string using `assign_location()`,
(b) use `fillna("unknown")` to use the string `"unknown"` as a placeholder,
(c) use `get_dummies()` to make the values binary, **note: you must prefix the new columns with `loc`**,
(d) merge the new columns from (c) in the DataFrame using `concat()`.

§ Task: Merge `LocalDate` and `LocalTime` and convert to a `datetime` object.

You can see that `LocalDate` and `LocalTime` are separate fields and also not seen as `datetime` objects to Python. This is a common task that is often necessary in order to ask questions that are time-related (and for plotting time series data).

You must merge the two columns, but luckily they are already strings, so Pandas will allow syntax like `df.col1 + df.col2` which is the same string concatenation behavior as in standard Python.

Also, luckily, `pd.to_datetime()` is a smart swiss army knife such that if you provide a *good enough looking date string* it will be smart enough to properly convert it to a dateime object.

In our case `"2021-01-01 23:33"` is a *good enough date string*.

Do not overthink this part. Simply concatenate the two local time/date columns with a space `" "` in between and call `pd.to_datetime()`. The result will be the new column to add to your dataset.

§ Task: Convert missing `SQMReading` values to `-1`.

Since there were no requirements in the data to provide SQM Readings, there are a lot of missing data. Instead of removing these rows, we will fill them with an invalid value of `-1`.

You may find `fillna()` useful.

§ Task: Store a new dataset file with a reduced set columns which include our numeric ones.

Now that we have engineered a decent dataset using the techniques above, it is now time to produce a file so it can be reused and reloaded, preserving the fruits of our labor.

You will use `to_csv()` to produce a new file called `"2014_to_2024_gan_data_working.csv"` to represent our working data file for future use and analysis. It will have only numeric and date date.

The new file will **only** have the following 27 columns:

```
'Latitude',
'Longitude',
'Elevation(m)',
'LocalDateTime',
'LimitingMag',
'SQMReading',
'CloudCoverPct',
'const_Bootes',
'const_Canis Major',
'const_Crux',
'const_Cygnus',
'const_Gemini',
'const_Grus',
'const_Hercules',
'const_Leo',
'const_None',
'const_Orion',
'const_Pegasus',
'const_Perseus',
'const_Sagittarius',
'const_Scorpius',
'const_Taurus',
'loc_remote',
'loc_rural',
'loc_suburban',
'loc_unknown',
'loc_urban'
```

(20%) Complete the online assessment

Please ZIP the folder and subfolder for your assignment and submit it directly to Blackboard.

Once you are done with the coding part of the assignment, you will need to complete the online assessment for the final 4 points (20%) of your grade.

§ Task: Turn in your solution and complete the online HW1 assessment.

(0%) BONUS: Normalize column names in working dataset.

You might notice a few inconsistencies in column naming in your final dataset.

While this is not desireable, for the assignment, I allowed for this variation.

However, a better way would be to normalize the names – that is make them consistent with one another so that they appear to represent a logic when dealing with names.

Some preferred guides to do this suggest:

- names that are all lowercase
- `"_"` (underscore) separating names (such as `LocalTime` being `local_time`
- using meaningful abbreviations when needed

You can update the column names for bonus points – make them consistent and follow the guides above OR if you choose a different style, include that in your submission.

You will turn in a notebook with the code that modifies the column names to the chosen style **and** stores that in a file `2014_to_2024_gan_data_final.csv`.

You may earn up to 2 bonus points depending on your solution.

§ Task: Normalize the column names and store the new dataset to a file called `2014_to_2024_gan_data_final.csv`.

Turn this in with your normal HW1 notebook code.