* Project:

Which park to visit tomorrow?

* Motivation:

Personally, I’m a park lover, and when visiting new places in the United States, I always wonder which park shall I visit nearby. Therefore, by this motivation, I worked on this project. The project is a ranking system which ranks the parks in the United States and recommended

the top 5 to the user given their current location.

* Dataset:

In this project, I collected information about parks. More specifically I focused on 5 different data:

Distance: the distance between a park and my current location.

Weather: the weather condition in the park, it can be sunny, windy, cloudy, rainy etc.

Temperature: the temperature in the park

Wind Speed: wind speed in the park

Rating: rating of the park by one of the Google reviews

I'm specifically focusing on "next day" because weather data is temporary, so we cannot use that as a factor to conclude which park is the best in a relatively long time period.

These 5 data come from 3 different datasets:

1. Park name and location data: [Scrape]

This dataset is scraped from https://www.latlong.net/ where we can find a huge list of parks in the United States and their locations. Latitude and longitude of these parks can be obtained. The “Distance” factor is based on this dataset.

2. Weather data: [API]

This dataset is from api: https://api.weather.gov/gridpoints/lat,long, where given a latitude and longitude, we can get the real-time data for that area. The “Weather” “Temperature” “Wind Speed” factors are based on this dataset.

3. Rating data: [Scrape]

This dataset is scraped from google, we search the google using the parks' names as keywords, we also add a "review" keyword so that we will always find ratings for the parks. In the cases where a rating for a park is not found, we assign an average score to the park to prevent errors. (This is actually very rare in our project, and almost every park has a score which can be scraped.)

[Note] The weather data and rating data will be collected after park location data. This is because in the program, after we get all the park location data, we will only consider 10 nearest parks for user to visit, or otherwise the parks could be too far from the user. Therefore, we only obtain weather data and rating data for these 10 selected parks instead of getting these data for all parks. This can save time and scraping a small amount of data means that we are not likely to be detected by some anti-scraping mechanics, which we also set a small-time delay (0.5s to 1s) after each scrape to prevent that. Also the weather data’s API call might fail occasionally but I noticed that this is just temporary, and a solution to this problem is to keep calling that API method after a small time interval, and eventually we can get the results.

Examples of the dataset can be found here:

python hw4.py --static park\_location.csv:  
  
=========================================================

Reading file: park\_location.csv

=========================================================

Field names: name, location, lat, long

Displaying 10 of 394 rows of data:

--------------------------------------------------------

[name]: Barnsdall Art Park

[location]: CA, USA

[lat]: 34.100376

[long]: -118.294640

--------------------------------------------------------

[name]: Luna Park in Coney Island

[location]: NY, USA

[lat]: 40.575245

[long]: -73.978577

--------------------------------------------------------

[name]: Liberty State Park

[location]: NJ, USA

[lat]: 40.703693

[long]: -74.052315

--------------------------------------------------------

[name]: Honeymoon Island State Park

[location]: FL, USA

[lat]: 28.063587

[long]: -82.831558

--------------------------------------------------------

[name]: Kings Island

[location]: OH, USA

[lat]: 39.345097

[long]: -84.272026

--------------------------------------------------------

[name]: LEGOLAND Florida Resort

[location]: FL, USA

[lat]: 27.988932

[long]: -81.691345

--------------------------------------------------------

[name]: Stone Mountain Park

[location]: GA, USA

[lat]: 33.805359

[long]: -84.145882

--------------------------------------------------------

[name]: Williamsbridge Oval

[location]: NY, USA

[lat]: 40.877319

[long]: -73.878029

--------------------------------------------------------

[name]: Memorial Skatepark

[location]: Colorado Springs, CO, USA

[lat]: 38.832462

[long]: -104.796875

--------------------------------------------------------

[name]: Lummus Park

[location]: Miami Beach, FL, USA

[lat]: 25.781084

[long]: -80.130119

--------------------------------------------------------

python hw4.py --static weather.csv

=========================================================

Reading file: weather.csv

=========================================================

Field names: name, day, temperature, wind, forecast

Displaying 10 of 10 rows of data:

--------------------------------------------------------

[name]: MacArthur Park

[day]: Sunday

[temperature]: 69

[wind]: 0 to 10 mph

[forecast]: Mostly Sunny

--------------------------------------------------------

[name]: Grand Park

[day]: Sunday

[temperature]: 70

[wind]: 0 to 10 mph

[forecast]: Sunny

--------------------------------------------------------

[name]: Echo Park Lake

[day]: Sunday

[temperature]: 70

[wind]: 0 to 10 mph

[forecast]: Sunny

--------------------------------------------------------

[name]: Hancock Park

[day]: Sunday

[temperature]: 69

[wind]: 0 to 10 mph

[forecast]: Mostly Sunny

--------------------------------------------------------

[name]: Barnsdall Art Park

[day]: Sunday

[temperature]: 70

[wind]: 0 to 10 mph

[forecast]: Mostly Sunny

--------------------------------------------------------

[name]: Elysian Park

[day]: Sunday

[temperature]: 70

[wind]: 0 to 10 mph

[forecast]: Sunny

--------------------------------------------------------

[name]: Kenneth Hahn State Recreation Area

[day]: Sunday

[temperature]: 66

[wind]: 0 to 10 mph

[forecast]: Patchy Fog then Mostly Sunny

--------------------------------------------------------

[name]: Lake Hollywood Park

[day]: Sunday

[temperature]: 70

[wind]: 0 to 10 mph

[forecast]: Sunny

--------------------------------------------------------

[name]: Griffith Park

[day]: Sunday

[temperature]: 71

[wind]: 0 to 10 mph

[forecast]: Sunny

--------------------------------------------------------

[name]: Hermon Park (Arroyo Seco Park)

[day]: Sunday

[temperature]: 72

[wind]: 0 to 5 mph

[forecast]: Sunny

--------------------------------------------------------

python hw4.py --static rating.csv

=========================================================

Reading file: rating.csv

=========================================================

Field names: name, rating

Displaying 10 of 10 rows of data:

--------------------------------------------------------

[name]: MacArthur Park

[rating]: 3.5

--------------------------------------------------------

[name]: Grand Park

[rating]: 4.5

--------------------------------------------------------

[name]: Echo Park Lake

[rating]: 4.5

--------------------------------------------------------

[name]: Hancock Park

[rating]: 4.5

--------------------------------------------------------

[name]: Barnsdall Art Park

[rating]: 4.0

--------------------------------------------------------

[name]: Elysian Park

[rating]: 4.0

--------------------------------------------------------

[name]: Kenneth Hahn State Recreation Area

[rating]: 4.5

--------------------------------------------------------

[name]: Lake Hollywood Park

[rating]: 4.5

--------------------------------------------------------

[name]: Griffith Park

[rating]: 4.5

--------------------------------------------------------

[name]: Hermon Park (Arroyo Seco Park)

[rating]: 3.5

--------------------------------------------------------

* UI

This project is complicated because we are requiring a lot of user inputs and it would be very ugly if we try to get all those inputs from the command line. It is doable though, just I don’t like it. Instead, there is a UI for this project and when you run python.exe main\_app.py, a python UI will pop up and it looks like this:

Graphical user interface, application

Description automatically generated

You may notice in the UI there are a few parts and I’ll give a more detailed explanation below.

Graphical user interface, application

Description automatically generated

In area 1:  
You can find several checkboxes; you can check the box to indicate that you want a factor to be

considered, or you can also uncheck the box to indicate that you don't want this factor.

In area 2:

You can input the numbers in the text input to rank the preference. You should rank all factors

from 1 to 5, where 1 means the most important factor and 5 means the least important factor.

In area 3:  
You can enter your longitude and latitude to represent your current location. These can be either integers or float values. If you enter nothing here, the default location will be used, which is the location of USC.

In area 4:

This is a button to click when you finish editing the above input, for grading purpose or for easy testing, you can just do nothing above and simply click this button.

In area 5:

Results will eventually be shown here.

* Analysis

After we collect all data for the 10 nearest parks. We perform some analysis.

We assign a score to each park.

The score is calculated based on 5 factors:

[Distance]

[Weather]

[Temperature]

[Wind]

[Rating]

For [Distance] factor, the score is based on the distance between the park to our current location. The closer the park is, the higher the score the park gets.

For [Weather] factor, sunny or clear days get the most score and rainy days get the least scores.

For [Temperature] factor, the most comfortable weather gets the highest score, a too-low or too-high temperature will get a low score.

For [Wind] factor, smaller wind generally gets a better score than heavy wind.

For [Rating] factor, rating is directly related to score with some mathematical conversions.

All the scores above will be assigned a weight and the final weighted score will be added to the final score of the park. Eventually, we recommend parks from the highest scores to the lowest scores. The score is hidden from the user, but if you really like to see the score you can print the score in get\_rank() function.

* Result and visualization

There are two results the project is going to generate: a text result and a graph result. Note that in the project I comment out the graph part because this is not going to work on all devices because of the lack of maintenance of the graph library I’m using.

For instance, given user input:  
  
Graphical user interface, text, application, email

Description automatically generated  
  
You can see the text result in above, also a graph result will be generated:   
  
Chart

Description automatically generated  
  
This graph result is plotted using basemap library where the locations are projected onto a map. User’s location is marked yellow, parks’ locations are marked green. And I implemented a strategy to zoom in the map to just able to fit in all the parks. Also you’ll notice the longitude and latitude lines in the map, this is also an awesome feature of the library.  
  
Let’s look at another example:  
This time we don’t care too much about the distance but focus more on the rating and weather aspects.  
  
Graphical user interface, text, application, email

Description automatically generated

Graph result:  
Chart

Description automatically generated

You can notice this time, the parks we are recommending including some parks far away.

* Script

Here is a picture explaining how scraping data, storing data, loading data and printing data works:  
  
Graphical user interface, application

Description automatically generated

You may need to zoom in to view everything in more details.   
  
Apart from that, main\_app.py is responsible for the UI and plot\_geo.py is responsible for plotting the graph.

* Maintainability

There are a few possible ways the model can break:

1. If the API stops working or any website that we scraped makes big structure changes. Our model heavily relies on real time data rather than static data. We are getting the rating and weather data for a park. These can change daily, and they should not just be read from a static local data source. If this is a commercial product rather than a class project, it would be better to use paid API services to obtain these data rather than scraping, because paid APIs are usually maintained well.
2. When new park data is added to https://www.latlong.net, that park might be very new that there is no google review for it. This is okay, since we have a 4.0 rating by default to prevent this from happening. However, the newly added data might be a malformed data. I've noticed that the park "MacArthur Park" added to the website has name "Santa Monica" instead of "MacArthur Park" and I've manually corrected this in my code. However, if newly added parks have this issue, there will be no catastrophic outcomes since our models will not throw an error. However, the data we are displaying to the user might be wrong. Take the above example, they will see a park with name "Santa Monica".
3. The graph library needs to be replaced with a better one.

* Extensibility

This model is extensible in a few ways:

1. We can recommend other places to visit other than parks.
2. For now, we only recommend which park to visit the next day. We could extend the model to let user input a time within the current week and recommend parks on that day.
3. Currently, we are assuming people want to visit parks during the day. However, some might prefer to visit at night, so we can probably add a night choice.
4. 4. We can collect other factors people consider when visiting a park, such as crime rate near the park.

* Conclusion

I like this project a lot and this is what I want to do for a long time, thanks for this class to give me the chance to do this. Actually, I have started using this tool and have decided which park I’d like to visit during the weekends.   
  
This project has a few drawbacks though and I’ve made too many assumptions based on my personal preference: for instance, I’m assuming nobody wants to visit a park during nighttime and nobody wants to visit a park when there is heavy rain. However, this might not be always true and I should really be more inclusive in this app if I want others to use it.