Outdoor Guides - Hiking Trail Analysis

Mike Gruzynski, Fall 2017

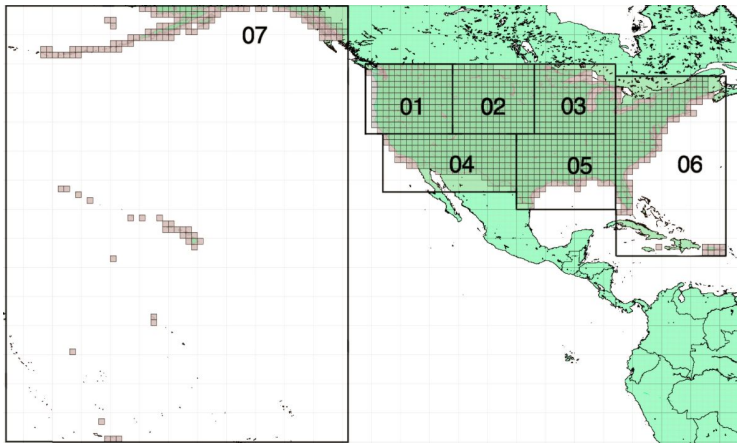
Background and Motivation

The goal of the final project was to expand knowledge and skill with pyspark along with managing a variety of data types in a high performance computing cluster. The original idea of the project was to have advanced data analytics on Yelp, utilizing time dependent data, geographical features, and text processing. However, data from Yelp was protected and we were quickly tagged as a bot when trying to parse the data. We then moved to open sourced hiking trail data, in order to also do some advanced analytics within the hiking world, linking multiple data types together. Trail data was gathered from a plethora of community based hiking apps along with blogs. It contains trail reviews, photos, maps, trip tracking, and check ins, etc. Inspiration for hiking analytics is the fact that hiking has increased in popularity over the last decade and has gone from around 29 million people going hiking/backpacking in 2008 to just over 47 million people in 2017 [1].

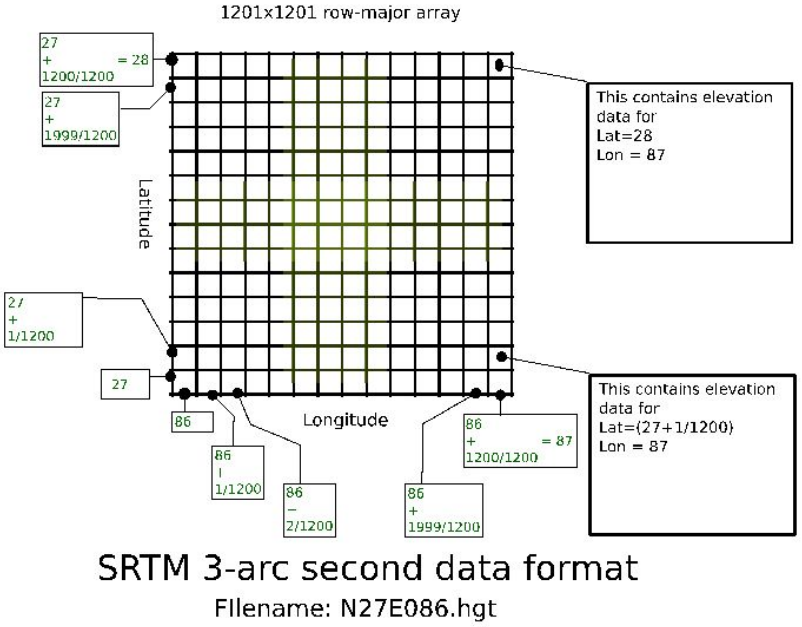
Data Sources

The first data type we extracted was with the use of Selenium web parsing in order to extract features from each hike in the United states along with links to other important information (i.e location of photos and gpx files). There are over 40,000 hikes in the United states when filtering out irrelevant or out of date hike information. Features were extracted from the data sources in order to be used for further data extraction later or use for converting each hike into euclidean space in order to perform cosine similarities in order to find hikes similar to favoriable hikes, this is discussed more in later section. In addition to features such as distance, location, and dummy variable extraction (i.e. hiking, fishing, backpacking, camping, or other tags associated with the hike), image location, user reviews, and gpx data was extracted in order to be used later. GPX or GPS exchange format is an XML schema that serves as the industry standard for storing GPS data (latitude, longitude, elevation). The GPX data will be used for both features for data analytics, and also linking 3-dimensional line plots (hike route information) with elevation data in order to better visualize hikes along with terrain it is associated with.

Elevation data was gathered by extracting Shuttle Radar Topography Mission (SRTM) data. The SRTM data was created in February of 2000 by the space shuttle Endeavour and collected interferometric radar which compared two radar images or signals taken at slightly different angles in order to calculate an elevation at each latitude and longitude scanned. The SRTM successfully collected radar data over 80% of the Earth's land surface between 60° north and 56° south latitude with data points posted every 1 arc-second which is approximately 30 meters. The image below [2] shows the regions of the US successfully scanned by the 1 arc-second data.



Each one of the sub-boxes within each region corresponds to a box with length 1-latitude and 1 longitude. The data was extracted from usgs.gov [2]. The data is efficiently packed in a zipped up binary file. The file can be extracted using numpy function fromfile(), with dtype i2 (or 16bit signed integer). The data is then an array of length 12,967,201 (which corresponds elevation data that needs to be reformatted in a 3601 x 3601 array. The latitude and longitude can be created from the file name, for example if a file name is: N27E086.hgt, then that file ranges from latitude 27 to 28 and longitude 86 to 87. An example of the structure is shown in the figure below for SRTM3 data (or 90 meter data) [3]:



This data was efficiently packed in because when trying to convert every table into a dataframe 3 x 12,967,201 (column for latitude, longitude, and elevation), the data quickly bloated and estimated to be over terabyte of data just for the US, when the binary format totaled

27 GB of data. The SRTM data was used as schema at read data and used in conjunction with GPX data in order to look at surrounding terrain to see if the hike is exactly what you were looking for.

In addition to the SRTM data, an interest to hiking is also weather. In addition to SRTM data, we also linked hikes up with global surface summary of the day (GSOD) weather data. This is a global dataset from all around the world and includes over 7500 weather stations in the United States. The data starts in 1901, however, we only took data from 1972 to present based on recommendation from NOAA on the health of the dataset. The data was extracted from [4], and the weather stations would be used to geographically filter by latitude and longitude of a hike in order to capture aggregates of weather of the area by month and day of the year.

Data Filtering

All trails data was a combination of unstructured and structured data in html format. We used selenium and Beautifulsoup as primary code software to parse the data. Once the data was parsed we converted the data into a structured data frame with varying data types in each column (i.e float for distance, list of strings for reviews, string for forest, etc..). We saved GPX data for each hike in the XML file format and wrote a parser for the format in order to get arrays for latitude, longitude, and elevation data. The final format of the data was spark sql dataframe format.

The SRTM data was structured data with little manipulation needed to convert it into final format. The data was parsed using a combination of numpy and pyspark in order to filter the data efficiently. Filters were applied in order to filter out elevation data a certain distance away from the bounding boxes of the hike and also any information with elevation below 300 feet was automatically filtered out, because the lowest elevation point in the United states is located in Death Valley, CA which is -282 ft below sea level [5]. The final format of the data was spark sql dataframe format, converted to numpy arrays for easy plots in plotly and matlibplot.

The GSOD data was structured data that needed no manipulation in order to get it into final format. The only thing that was needed was to write some pyspark filters and aggregation functions in order to create weather data features and to filter out erroneous data. There was some keyed in value for missing data that needed to be ignored for final aggregations and data analytics (i.e -999 values for temperature or precipitation). In addition there was some keyed in flags in the precipitation column that signaled if the data was a 6 hr average, 12 hr avg, 24 hr aggregate, etc.

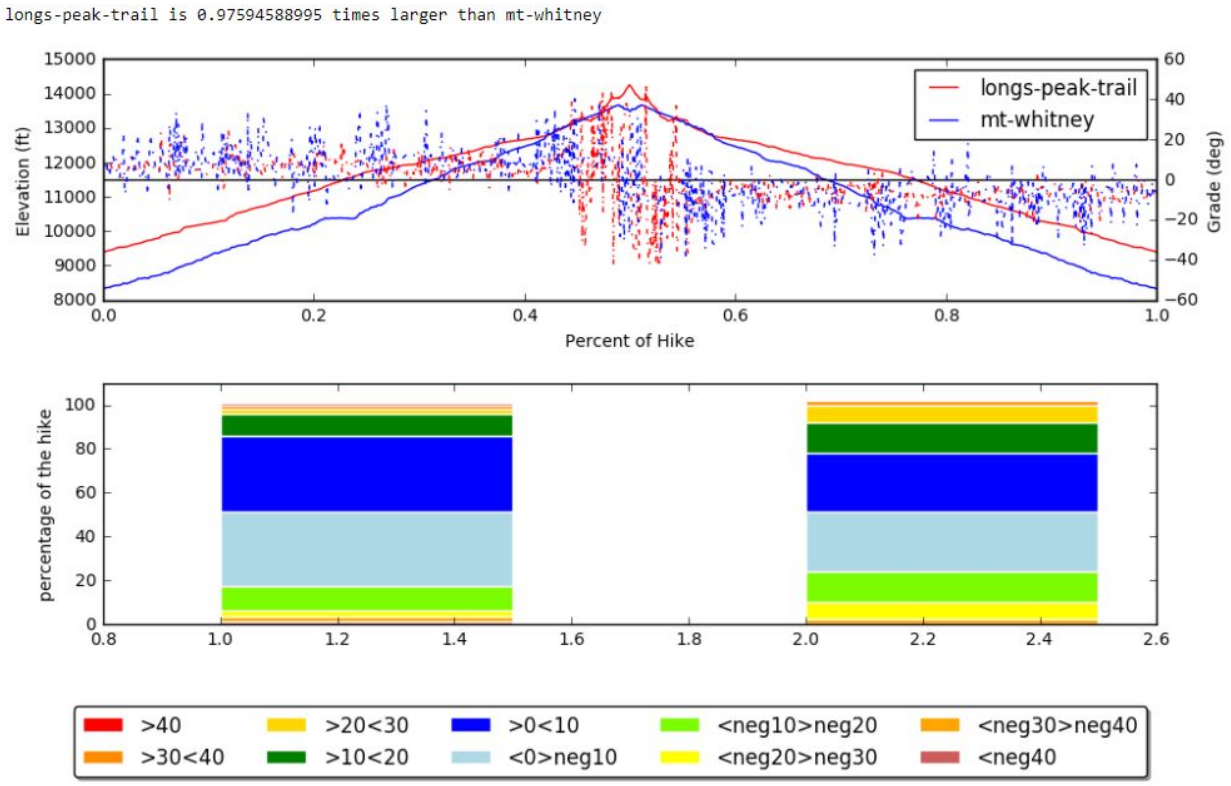
Data Processing and Analysis

Hiking data was taken and converted into pyspark sql data frames. All continuous and binary variables were kept in their raw state. All categorical data was converted into a matrix of dummy variables. For example if you had a column for state, and the column had three rows with one row equal to Washington and two rows equal to California, then the categorical data would be converted into a 2x3 matrix corresponding equal to [[1, 0], [0, 1], [0, 1]]. This was done with over 40 thousand hikes with 50 categorical data points for state, 3 categorical data points for typeof hike (i.e loop, out and back, etc..) and over 8000 columns for forrest name that hike was in. This converted the data frame of hikes to a very large number (well over 400 million data cells in the dataframe).

The hikes were then vectorized into euclidean space based on whatever hiking feature was important to the user. For example if total length, elevation, and loop hikes are important to the user then the data [[12.3], [4319], [1]] would be vectorized to [12.3, 4319, 1] in order to be feed into pyspark logic for cosine. The pyspark code would then calculate cosine similarities between the static hike and any hikes you feed it and returned with the top 5 hikes that you may like based on an input hike you liked based on euclidean space similarities. Example of the output is shown below (which barr-trail-to-pikes-peak is very similar to the enchantments (Mike has done both)):

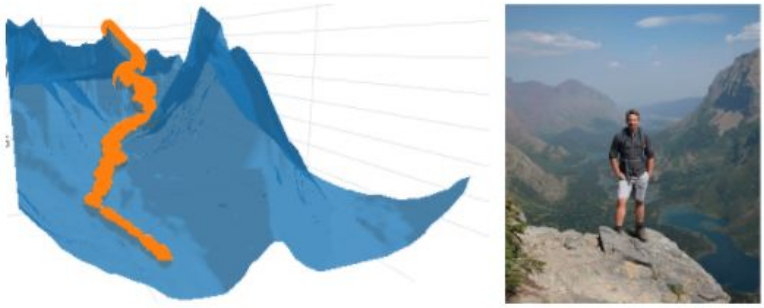
HIKE_NAME	eucledian_distance
the-loneliest-road-us-highway-50	0.014072820337093672
grand-mesa-scenic-byway	0.04487835061354428
barr-trail-to-pikes-peak	0.057116101219970794
colorado-trail-denver-to-salida	0.06388285414928285
phantom-canyon-gold-belt-tour-back-country-byway	0.07961028206829401
mount-evans-scenic-byway	0.08035867594506085

Another feature for hiking analysis was to compare hike difficulties. For example there are a lot of websites that list the hardest hikes in the United states, but never give reason as to why the ranking were created in that way. With the data analysis, we created quick looks at hikes in order to compare two hikes. For example, if you look at this top 10 website [6], it shows the Longs Peaks, keyhole Route is the hardest hike in the US followed by Mount Whitney, Whitney Portal in California. The results is fairly subjective with loose linkage to data. However, looking at the image below:

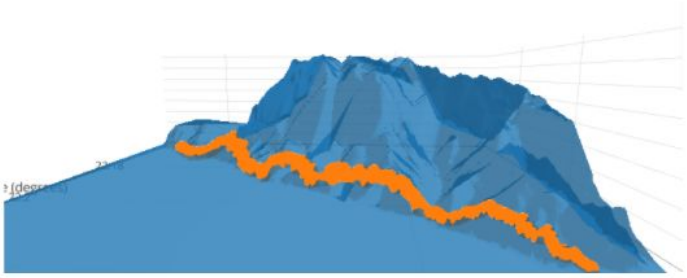


With the above image, we can see with the quick look of the date that Mt. Whitney (blue line graph and right bar graph data) appears to be the more difficult one based on the amount of the hike spent at higher grades (yellow, reds instead of blues and greens). In addition the area underneath the curve is calculated with the actual distance (instead of percent done with hike as shown in the top graph) in order to give an overall perspective of the size of the hike difference in order to take the aggregation under consideration. The area comparison can be seen as text before the matplotlib image

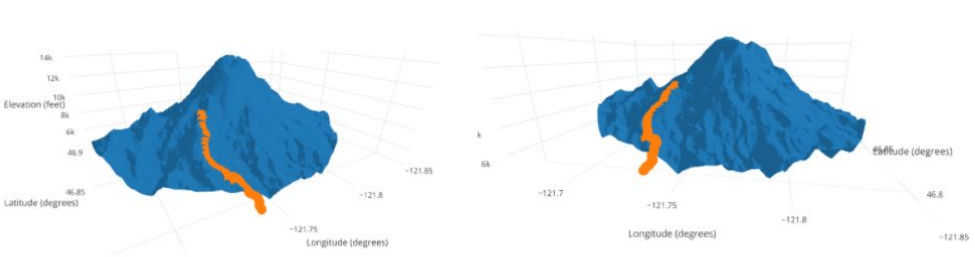
In addition to quick comparison of data, another important aspect of choosing a hike is to make sure that the terrain and hike are exactly what you are looking to do. In order to do this, we linked SRTM data with hiking GPX data in order to make 3-dimensional plots to view hiking data with elevation data. Shown below are hikes in GPX/SRTM data (left vs actual personal photo right)



Swiftcurrent Pass - Montana



Nā Pali Coast Trail - Hawaii



Mt. Rainier - Washington

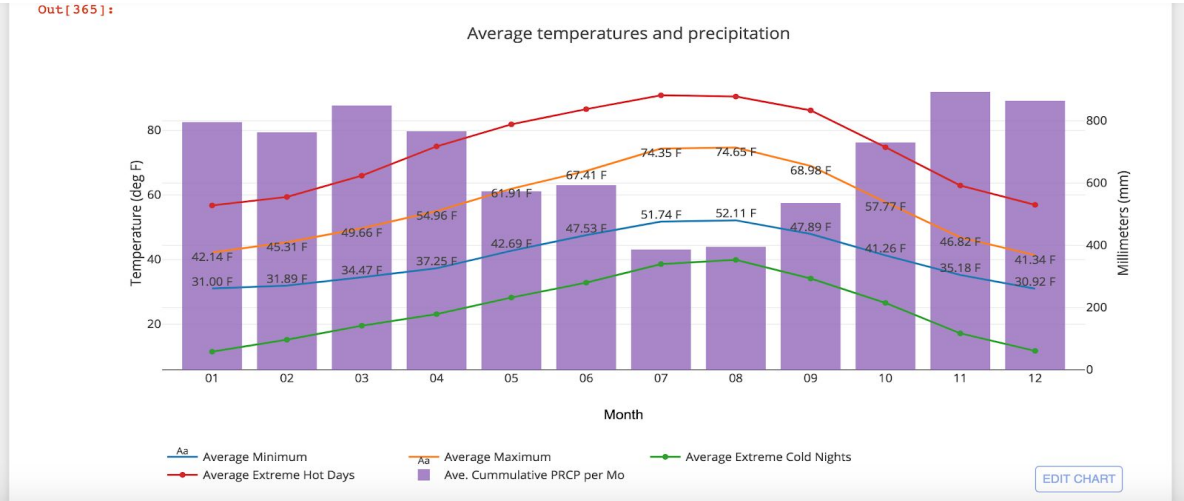
The user inputs a hike and a search radius (or how far to plot the SRTM data and plots the function and the above charts are observed. This will give you a more realistic view of the mountain and the surrounding terrain of the hike.

In addition to providing comparative information in order to suggest hiking trails to users by means of mapping selective features to euclidean space or scoring similarity of two trails by minimizing their differences. We also provide dashboard like views to paint a picture of the weather conditions by month to pain a picture of the ideal time of the year for a user to plan their trip for any given site. Additionally we created an optimization tool that allows the user to input ideal weather related conditions, and a desired time window, $(\Delta t_i, \Delta t_j)$, to find the the best N number of day ranges to visit this particular hiking trail.

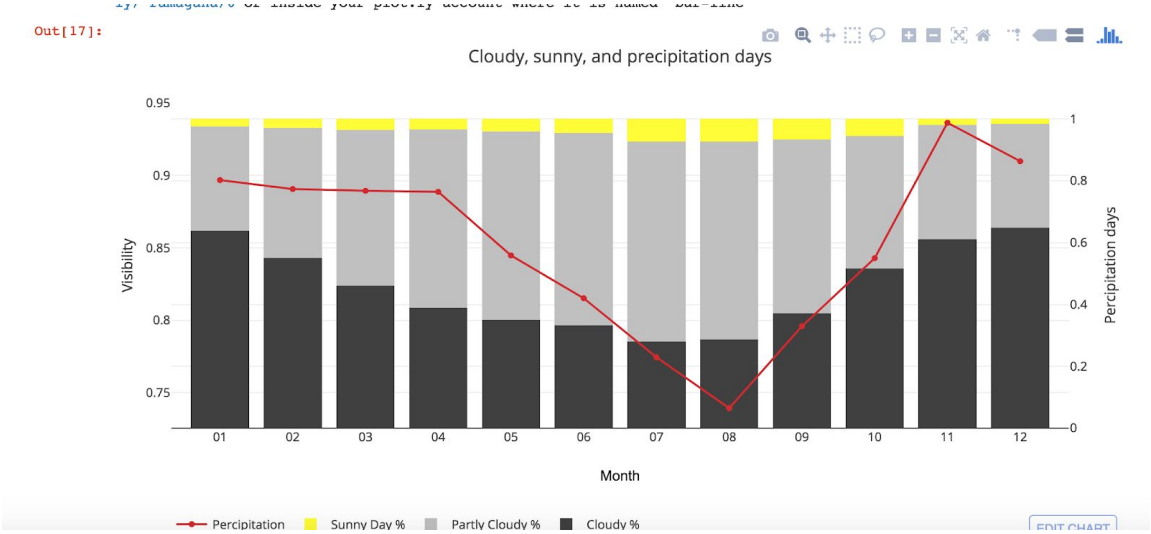
The GSOD data itself is collected globally gathering surface weather information such as: temperature, visibility, precipitation, air pressure, wind-speed, and more . All the data is collected at weather stations for which we have latitude and longitude information for. The weather for each site is represented by the data collected at the 10 nearest stations to the latitude/longitude coordinates determined to be the centroid for each hike. After putting the

data for the 10 representative weather stations in one spark dataframe we performed various aggregations on the data.

Using the MeteoBlue.com[8] website as an example to illustrate weather data usefully to the user planning their hiking trips for the year. Below we will are three different examples of charts. In the chart below, *Average temperatures and precipitation*, we show a bar chart of the average amount precipitation in mm for each month averaged across all years of the data (which extends back to 1972). Overlaid on top the bar chart is a secondary axis displaying the average maximum daily temperature in orange and the average minimum daily temperature in blue. Additionally the top red line shows the average hottest temperature for the month over then years while the bottom green line shows the coldest month temperature. This information give the user a range of possible temperatures.

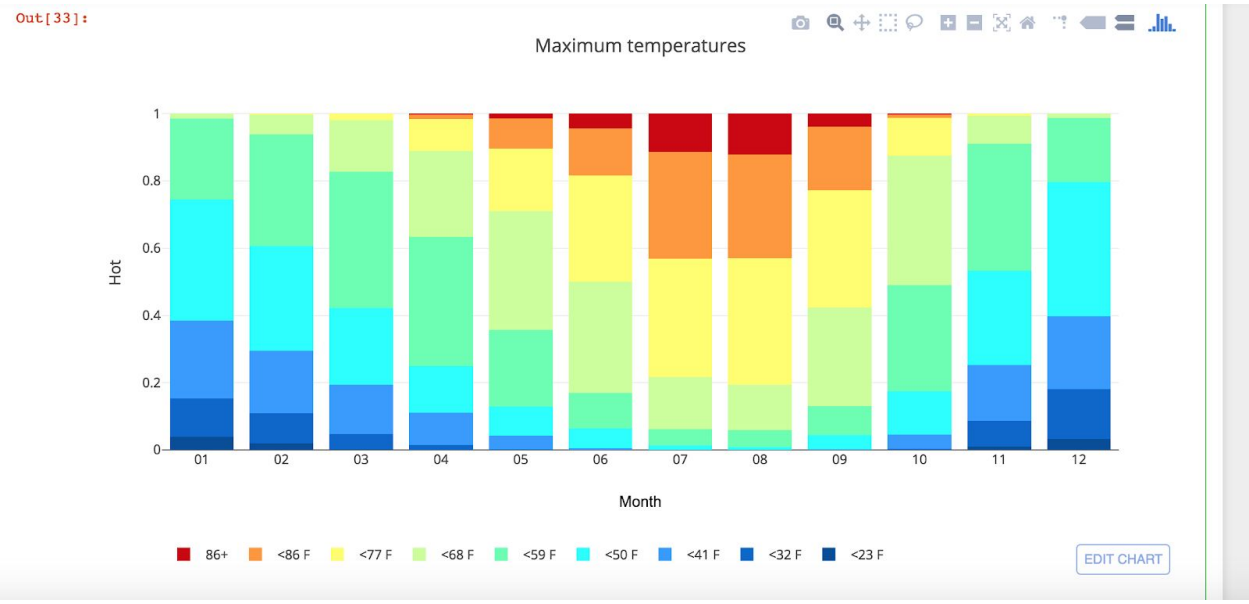


The chart, *Cloudy, sunny and precipitation days,* is also a line and bar chart overlaid on a dual axis. What this chart shows is the number of precipitation days, rather than the amount. Also the bar chart shows the average proportion of overcast, cloudy, and sunny days for each month across the year. Lastly the third chart is the is another bar chart that is displays the proportion on a stacked bar chart of days that fall in a particular range. Rather than showing the range of visibility, the third chart,



Maximum Temperatures , tells the user the average the proportion of peak daily temperatures as they fall into on of 9 buckets. This also provides visual cues to the user showing how warm across the year they can/should prepare for.

Out[33]:



Functions where created for all of the data frames and visualizations making the charts easily reproducible and reconfigurable by the as if they are incorporated into a more dashboard.

In addition to the chart providing the user with visualized information to plan their trip we created a tool that outputs a table from configured parameters from the user suggesting the top N number of dates with the window $-\Delta t_i$ and $+\Delta t_j$ optimizes to for an ideal condition all of which are defined by the user. The output is a table that give the month and day of a year for the user to use as a suggestion.

In [31]:
```
df_3 = continous_score(df = df,dayWindow = (-2,2), idealCondition= 70, conditionField = 'TEMP')
df_3.select(['MONTH','DAY']).orderBy('TEMP5DayScore',ascending = False).limit(100).distinct().show()
```

```
+-----+---+
|MONTH|DAY|
+-----+---+
|   08| 07|
|   07| 12|
|   08| 06|
|   07| 11|
|   07| 13|
|   08| 08|
|   07| 10|
|   07| 14|
|   07| 09|
|   08| 05|
|   07| 15|
|   07| 04|
|   07| 05|
|   08| 09|
|   07| 16|
|   07| 08|
|   07| 06|
|   07| 03|
|   07| 07|
|   07| 17|
+-----+---+
only showing top 20 rows
```

Future Work

We ran out of time, but was very close to linking in twitter data. We had a neural network CBOW and CNN model linked up with past review data in order to discover features of the reviews that correspond to the rating of the mountain. We wanted to link that data with a live twitter stream from popular hashtags (like #outdoor_project #optouside #getoutside) and also recent reviews from hikes the user was interested in and predict if it would be a good rating or bad rating or in other words worth the effort to go on this hike at this time based on recent reviews and comments about the hike. In addition to that, we would have liked to do analysis of variation of surface terrain and try and compare a specific locations elevation information to another location in a different state and even Mars data (or Mola [7] for fun and to get ready for when Elon Musk gets us there to set up national parks there) to see what other states have terrain similar to the one the user likes. Also, live streaming elevation data so with a certain filter distance. So essentially load the terrain and hikes and then pan around the map to stream new terrain data and overlay potential hikes that are in the area so that you can continuously look around the map at new terrain and potential hikes.

References:

[1] https://www.statista.com/statistics/227421/number-of-hikers-and-backpackers-usa/
[2] https://dds.cr.usgs.gov/srtm/version2_1/SRTM1/
[3] https://librenepal.com/article/reading-srtm-data-with-python/
[4] ftp://ftp.ncdc.noaa.gov/pub/data/gsod/
[5] https://www.nps.gov/deva/learn/nature/lowest-places-on-earth.htm
[6] https://www.marmot.com/destination-guides/x5gWsSsidGFEE2wpPhMZX1Vx.html
[7] http://pds-geosciences.wustl.edu/missions/mgs/mola.html
[8]https://www.meteoblue.com/en/weather/forecast/modelclimate/colchuck-lake_united-states-of-america_5790482