Los Angeles Neighborhood Score
Final Report

Jameson Thai, Matheus Schmitz, Yeonsoo Park

The City of Los Angeles, home to nearly 4 million residents, is a trailblazer in many aspects, including data transparency. The city's open data portals contain publicly accessible datasets covering a multitude of aspects of life in the city, such as environment, economy, libraries and park, public safety, public services and more, with many datasets being updated weekly or even daily. This abundance of data was the inspiration for the teams project, which envisions the creation of a comprehensive neighborhood quality of life comparison score, and an accompanying visualization tool. The team behind the project is composed by Matheus Schmitz[1], with a degree in Industrial Engineering and work experience as Product Manager, along with Jameson Thai[2] and Emma Park[3], each with a background in Computer Science specializing in Data Science.

Along with the datasets provided by the official LA City webpage, other official sources were used to complement the projects portfolio of neighborhood evaluation metrics. In aggregate, the datasets used in this project come from the following websites: data.lacity.org, data.lacounty.gov, Geohub, cde, Zillow, USC's Socrata, and US Census Bureau. The extensive number of topics covered by these websites allows for countless dimensions in which to analyse the data, however for this project the primary focus will be on three themes: crime, education and housing. The main source for education information is Geohub, a data visualization tool from the LA City open data portal, whereas the crime data comes directly from the City of Los Angeles open data portal, and housing data is sourced from Zillow and USC's Socrata.

Throughout the academic semester the team was able to stay ahead of schedule in developing the project, which by the midterm report was already 80% complete. The team's effective progress can be attributed to diligent management of the project's schedule and high commitment by the team's members.

---

[1] "Matheus Schmitz - Innovation Associate - FCC | LinkedIn." https://www.linkedin.com/in/matheusschmitz. Accessed 14 Nov. 2020.
[2] "Jameson Thai - Course Producer - University of Southern ...." https://www.linkedin.com/in/jamesonthai. Accessed 14 Nov. 2020.
[3] "Emma (Yeonsoo) Park - Course Grader - Machine ... - LinkedIn." https://www.linkedin.com/in/-emma-park. Accessed 14 Nov. 2020.

Task	Person	Deadline	Status
Proposal slides	Emma	09/12	OK
Select datasets	Matheus	09/09	OK
Clean crime dataset	Matheus	10/12	OK
Clean school dataset	Emma	10/12	OK
Clean housing dataset	Jameson	10/12	OK
Upload data to MySQL	Jameson	10/16	OK
Write midterm report	Jameson	10/14	OK
Aggregate all datasets with Spark	Matheus	10/18	OK
Load data to Firebase	Matheus	10/18	OK
Create search function	Matheus	11/16	OK
Create web view	Emma	11/16	OK
Write final report	ALL	11/22	OK

Following the team's assembly, many potential project themes were debated, with the consensus being on the Los Angeles Neighborhood Score vision as the team's favorite. Next, each of the three topics of focus was assigned to one team member, who was responsible for downloading the selected dataset, preprocessing it on a as-needed basis, then uploading it to a MySQL database hosted on AWS' EC2. Below is a sample CSV containing crime data after it was processed and made ready for upload into MySQL.

	A	B	C	D	E	F	G	H
1	DR_NO	DATE OCC	Crime_Code	LAT	LON	Neighborhood	Crime_Weighted	Crime_Weighted_No
2	191907191	2019-03-08	510	34.2991	-118.4211	Sylmar	0.1960784314	0.1137069485
3	190125334	2019-10-17	330	34.0363	-118.2314	Downtown Los Angel	0.303030303	0.2466509062
4	191920961	2019-12-22	510	34.2283	-118.4611	North Hills East	0.1960784314	0.1137069485
5	190604395	2019-01-09	510	34.0944	-118.318	Central LA	0.1960784314	0.1137069485
6	191310615	2019-04-30	510	33.993	-118.2411	South Los Angeles	0.1960784314	0.1137069485
7	191419522	2019-09-03	510	33.9759	-118.3897	Westchester	0.1960784314	0.1137069485
8	190123550	2019-09-21	330	34.0467	-118.2411	Downtown Los Angel	0.303030303	0.2466509062
9	191915118	2019-08-18	510	34.2985	-118.4205	Sylmar	0.1960784314	0.1137069485

One critical task in this project was obtaining the neighborhood associated with each registry in all gathered datasets, as neighborhoods were envisioned as the project's final indexer for the obtained quality of life scores. The housing data happened to already come with neighborhood information in all its data points, dispensing any preprocessing to obtain them. The education dataset's only georeference was school names, which were processed in the google maps API to obtain each school's neighborhood. Because some schools had identical names to other schools outside of Los Angeles, this approach also generated data from neighborhoods outside of Los

Angeles, which had to be removed with a second filter that allowed only LA neighborhoods through. To avoid data loss this filter was applied only later in the process, at the Spark stage. Lastly, the crime data came with the latitude and longitude associated with each record, which were also converted into neighborhoods through the google maps API.

Neighborhoods can be extracted from the google maps python API using the reverse_geocode() method, which takes a tuple of (latitude, longitude) and extracts all information related to that point in the globe. This information includes country, state, county, city, neighborhood and well as additional information for certain locations. As the data varies from one place to another, so does the structure of the API's output, and hence it's necessary to explore all elements in it until the one containing neighborhood data is found. Then the neighborhood name can be extracted and appended to the record from whose (latitude, longitude) it originated. Below is an excerpt of the code used to obtain the neighborhoods using the google maps API.

```
In [*]: # Loop through each row in the crime dataframe
for idx, row in tqdm(df.iterrows(), total=len(df)):

    # Make a temporary dict (json) with data extracted with google maps' reverse_geocode
    tmp = gmaps.reverse_geocode((row.LAT, row.LON))

    # Take the length at [0]['address_components'] and create a range of attempts to find the neighborhood
    # [0]['address_components'] contains all address elements, eg: county, city, neighborhood
    # Each in one position in a list whose positions are denoted by the values in attempt_range
    try:
        attempt_range = range(len(tmp[0]['address_components']))

    # Some empty data points (lat and long = 0) will have len(tmp[0]) == 0, in those cases just skip the data point
    except:
        continue

    # Loop through each possible position where the neighborhood data can be
    for attempt in attempt_range:

        # For each possible neighborhood location, check [0]['address_components'][attempt_range]['types']
        # To see if type == neighborhood
        if 'neighborhood' in tmp[0]['address_components'][attempt]['types']:

            # If yes, then extract the neighborhood name, putting it in a new column in the dataframe
            df.at[idx, 'neighborhood'] = tmp[0]['address_components'][attempt]['long_name']

            # And go to next iteration (no need to check the other positions in the tmp, as the answer has been found)
            continue

        # Else continue the loop until neighborhood is found
        else:
            continue

    # After each 20000 rows, save the dataframe as csv
    if idx%20000 == 0:
        df.to_csv('Crime_Data_2019_Neighborhoods.csv')

0% |                                                  | 356/216102 [01:12<10:43:24, 5.59it/s]
```

Feature extraction was a complex part of this project, as not only was it necessary to obtain neighborhoods for all records, it was also necessary to create scoring metrics for each of the prioritized topics, each of which required a customized and thorough approach, an example of which is given in the following paragraph. For the housing and

education data, after the scoring features were obtained, the data points were then aggregated by neighborhood, with the scores being averaged. With the preprocessing finished, the data was then saved into a CSV that is later read by a Python program into a MySQL database. Below is an example of the housing dataset prior to upload to MySQL.

id	neighborhood	size_rank	housing_score
1	South Los Angeles	4	442013.573529411750000
2	Southeast Los Angeles	8	367733.691176470600000
3	Hollywood	18	847758.705882352900000
4	Mid City	62	986030.014705882400000
5	Van Nuys	64	535259.676470588200000
6	Sun Valley	68	513386.485294117650000
7	Northridge	69	670616.470588235300000
8	Sylmar	74	483452.161764705900000
9	Boyle Heights	75	424736.161764705900000
10	North Hollywood	78	613276.544117647100000
11	San Pedro	79	583256.838235294100000
12	Koreatown	82	629124.544117647100000
13	Woodland Hills	83	780429.264705882400000
14	Sherman Oaks	85	951315.764705882400000
15	Pacoima	86	434265.852941176450000

The pipeline for getting the crime data to MySQL consisted of the following steps: First the data was downloaded as a CSV from the https://data.lacity.org page. The whole dataset contained 2 million records, covering years 2010-2019, using all of which proved too burdensome, hence only records for the year of 2019 were kept, totalling about 200k records. Those records already contained latitude and longitude, hence the second step consisted of using the google maps API reverse geocode and obtaining the neighborhood in which the crime happened. Unfortunately the API has limitations, which stopped the queries after around 115k had been sent. This forced the team to resort to using a KNN algorithm to predict the neighborhoods for the rest of the data, which was step three. A train-test split showed this approach to be highly effective, as neighborhoods are by definition linearly separable classes in a low (two) dimensional space, about as ideal a setting for a KNN algorithm as one could hope for, resulting in a classification accuracy of 98%. Step four was the creation of a feature called “Crime_Weighted_Norm”, which was obtained by using the “Crime Code” associated with each register. “Crime Code” indicates the crime committed, with lower crime code numbers representing more serious offenses. Thus “Crime_Weighted_Norm” was obtained by first dividing 100 (the lowest/most serious crime code) by the crime code of each data point, and then normalizing, resulting in a feature in the [0,1] range, with higher values representing more serious crimes. Other columns on the original dataset amounted essentially to the full public police log on each crime, but contained no other

relevant information for this project and were discarded. Lastly, step five was storing the final dataset into MySQL.

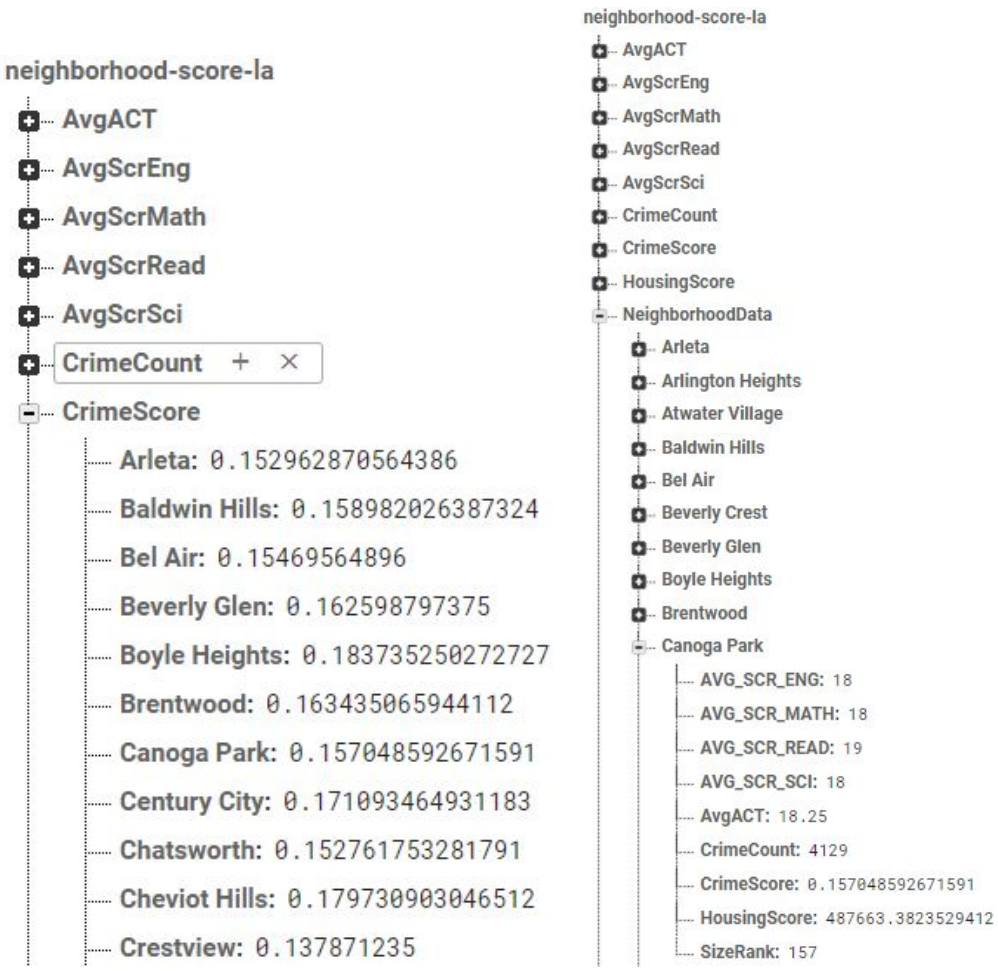
The MySQL database is hosted on an AWS EC2 instance. ID is a primary key and neighborhood names are a secondary key, with the neighborhood scores and other extraneous information as table attributes, all of which are shown in the image below. The extraneous information varied by dataset due to the what types of information that could be extrapolated from.

PRINT OF THE MYSQL DATABASE

In order to diversify the number of Spark use cases in the project, with the goal of enhancing the team's learning, we opted to have the crime data be saved on MySQL as individual data points (200k data points), instead of storing data already aggregated by neighborhood. This allows for Spark to be used to MapReduce the crime data and aggregate it by neighborhood in a more scalable fashion (as well as allowing for extra learning). The Spark MapReduce pipeline took the "Crime_Weighted_Norm" feature that had already been extracted from the data prior to upload into MySQL, and averaged it by neighborhood, while also obtaining a second feature: the count of crimes per neighborhood.

With the processing of the crime data finished, the two other datasets, which had already been cleaned prior to being stored in MySQL, were loaded into Spark. At this stage all datasets were already aggregated by neighborhood, and thus could be merged using Spark SQL. Once the aggregation pipeline grouped all datasets using the join function, the resulting tabular dataset was then converted into JSON (dictionary) format, making the data suitable for upload to Firebase.

In order to ease querying into the database, the Firebase architecture created was inspired on the homework 1 architecture, with each neighborhood score feature being stored twice: First following the index of *{Feature: {Neighborhood_1: Value, Neighborhood_2, Value, …}}* and second using the Neighborhood as the main index on a NeighborhoodData node, as in *{NeighborhoodData: {Neighborhood_1: {Feature_1: Value, Feature_2: Value, …}, Neighborhood_2: {Feature_1: Value, Feature_2: Value, …}...}}.* Both indexing structures can be seen in the figures below:

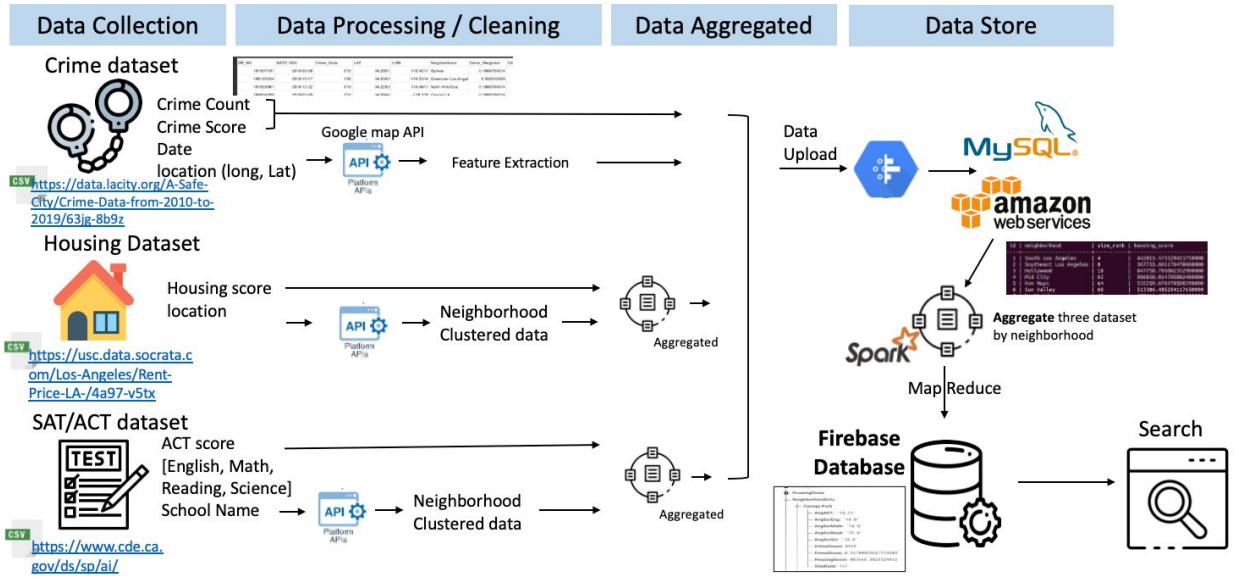


Benefiting from the dual-storage architecture of the Firebase database, three different search functions were developed, to most efficiently handle different scenarios. The first function is designed to deftly retrieve all information from one or multiple neighborhoods using the NeighborhoodData node in Firebase. The second function focuses on the inverse use case, whereby one or more metrics are informed and the score of each neighborhood is returned. The third function allows for dual filtering, where a set of neighborhoods and a set of features are given, and the matrix of resulting values is obtained. With regards to the backend implementation, all functions return a pandas dataframe containing the features as columns and the neighborhoods as rows. Such structure was chosen to simplify the ensuing data visualization step.

PARAGRAPH ON VISUALIZATION

A brief simplification of the project's overall workflow is: first datasets are cleaned; second neighborhoods and other features are extracted; third datasets are individually aggregated by neighborhood; fourth datasets are uploaded to MySQL; fifth data is read

from MySQL to Spark; sixth neighborhoods are used as index to group all datasets; seventh a filter is applied to ensure only neighborhoods in the City of Los Angeles move forward; eight data is loaded to Firebase; ninth a search program processes a set of neighborhoods and features from a user; tenth data associated with the query is displayed to the user. The complete data pipeline for this project can be seen in the architecture diagram below.



This project proved to be a great source of learning for the team, as it's focus on multiple datasets, from multiple sources, with multiple data processing and storage requirements allowed for experimentation with a variety of tools. Selecting a realistic use case strongly contributed to the team's engagement and continued interest, as well as to the applicability of the learnings. An outstanding dynamic among team members as well as the team's zealous organization of tasks allowed all requirements to be met without any peak in workload - a common source of lower quality deliverables. Although the formal delivery of the project is complete, the initial vision points to potential further developments, as more quality of life aspects can be easily incorporated into the project using the now-fleshed-out data processing pipeline.

Below are links to all data sources used.

Housing

https://www.zillow.com/research/data/

https://usc.data.socrata.com/Los-Angeles/Rent-Price-LA-/4a97-v5tx

https://data.census.gov

School
https://www.cde.ca.gov/ds/sp/ai/
https://geohub.lacity.org/datasets/70baf6da243e40298ba9246e9a67409b_0/data
Crime
https://data.lacity.org/A-Safe-City/Crime-Data-from-2010-to-2019/63jg-8b9z
Our Firebase dataset can be found at:
https://neighborhood-score-la.firebaseio.com/.json