DEVELOPING A PLUGIN IN QGIS FOR SELECTING THE LOCATION OF A NEW MANUFACTURING PLANT

by

Sanaz Ahmadzadeh Siyahrood

A thesis submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
degree of Master of Science in Architecture and
Master of Science in Information Technology

Charlotte

2021

Approved by:

_____

Prof. Jefferson Ellinger

_____

Prof. José L. S. Gámez

_____

Prof. Ming-Chun Lee

_____

Prof. Wenwen Dou

ABSTRACT

SANAZ AHMADZADEH SIYAHROOD. Developing a Plugin in QGIS For Selecting the Location of a New Manufacturing Plant
(Under the direction of PROF. JEFFERSON ELLINGER)

With the growth of the technology-driven world, today many designs and analyzes depend on smart software to address different computational concepts. In the meantime, locating and finding a suitable place for establishing a facility is one of them and is considered by urban designers, regional planners, and architects. Accordingly, the main goal of this study is developing a plugin in QGIS to aid in the decision-making of selecting the location of a new manufacturing plant by prioritizing the places that have the most renewable energies. Considering this logic has two main purposes; the first one is renewable resources, such as sunlight, wind, rain, tides, waves, and geothermal heat can supply all the energies needed for the productions of these factories while causing as little harm to the environment as possible. Second, we can locate these factories in locations with a low unemployment rate while providing maximum suitable conditions and facilities for the workers, thus helping to reduce unemployment rates in those areas. To reach these main goals, we developed a computational system titled the site selection decision making (SSDM | Site Selection Decision Making) plugin in QGIS3.12 software. The clustering method was used for clustering the important locations based on their accessibility to other facilities. Then binary classification which is a supervised machine learning algorithm, and its goal is to predict categorical class labels including discrete and unordered format was used for analysis and returning the final results. Pycaret library; pycaret.classification has been used for implementing the machine learning algorithm. In this regard, binary classification determines whether a site is suitable for establishing a new industrial factory or not. Therefore, its answer is yes or no considering several significant factors.

ACKNOWLEDGMENTS

While writing this dissertation I experienced a lot of support and assistance from my supervisor, and I would like to extend my sincere gratitude to him, Professor Jefferson Ellinger, for his invaluable advice, continuous guidance, and help. My academic research has been greatly influenced by his insightful comments, which helped me sharpen my thinking. His vast knowledge and abundant experience encouraged me throughout the years that I studied in UNCC, architecture, and information technology department. I want to thank him not only for supporting me during my master thesis but also because of all the opportunities he gave me for developing my ideas on different topics. It is my honor to have worked under his supervision.

I would like to thank all of my committee members; Professor José L. S. Gámez, Professor Ming-Chun Lee, Professor Wenwen Dou. I received their kind help, support, and valuable feedbacks during all sessions of my thesis reviews which have made my study a wonderful time.

I would also like to thank Professor Eric Sauda that guided me through the path of graduate studies at UNC Charlotte.

I would also like to thank Dr. Alireza Karduni, who guided me whenever that I needed help. With his guidance, I could pass more than 20 online courses from several universities to become more proficient in algorithmic thinking, computational design logic, different languages of code programming, and related skills.

I would also like to thank Professor Emily Gunzburger Makaš and Professor Mona Azarbayjani for their support during my study at UNC Charlotte, Descomp program.

DEDICATIONS

This thesis is dedicated to my husband, Dr. Saeed Bahrami, who has been a constant source of support and encouragement during the challenges of graduate school and life. It has been a pleasure having you in my life. Also, I dedicate this work to my mother and my father who have loved me unconditionally and who's their good examples inspired me to strive for achieving my goals and to work hard for what I want to accomplish. Also, to my lovely sisters; Dr. Nezhla Ahmadzadeh Siyahrood and Sahar Ahmadzadeh Siyahrood

And to my dearest Ilia.

This thesis work is also dedicated to my good friend which I have never seen, but to whom I owe and who reminds me of hardworking and kindness.; Sheida Hosseinzadeh. Graduate Research Assistant of UNCC which we cherish her memory. Dear Sheida, our hearts are always with you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

SSDM	Site Selection Decision Making
MEDLOC	MEDical LOCator
K-Mean	K-Means Clustering Algorithm (KNN)
DB Scan	Density-Based Clustering Algorithms
CEO	Chief Executive Officer
QGIS	Quantum Geographic Information System
SQLite	Structured Query Language, relational database management system
UI	User Interface
CCA	Community Choice Aggregators
RPS	Renewables Portfolio Standard
AUC	Area under the curve
ROC	Receiver operating characteristic
Lightgbm	Light Gradient Boosting Machine
dt	Decision Tree Classifier
rf	Random Forest Classifier

SECTION 1: INTRODUCTION AND OVERVIEW OF THE THESIS

1.1 – INITIAL CONCEPT

Studio Lab 2 with Professor Jefferson Ellinger sparked my interest in selecting this topic for my thesis. The course mainly was about uncovering and refining some details about different aspects of healthcare facilities; however, the intention was to design an artifact, dashboard, or application that could uncover urban spatial relationships through the collaboration between different stakeholders using unsupervised machine learning. In speculation, the initial concept of the MEDLOC (MEDical LOCator), ranged from a design interface for an "app" or tool to spatial organization. In that study, the final product became a collaborative dashboard for the stakeholders of the healthcare department which would include doctors, hospital deans or CEO, designers that could include architects and urban planners, real estate developers, and so on to collaborate with each other about several main factors of healthcare facilities such as access to public transportation, related infrastructure, etc. in different areas of Chicago city. Hence, MedLoc became a platform that visualized complex relationships between the different types of data geospatially and by applying the power of unsupervised machine learning algorithms, provided a vision for different stakeholders. So various stakeholders, based on their demands, or their expertise, could benefit from MEDLOC.

MEDLOC collaborated with a national architecture firm to develop a computational system to help developers address a variety of challenges. Hence, through the use of unsupervised machine learning, MEDLOC developed as an intuitive, user-friendly collaborative dashboard that helped identify the types of care needed by groups of people and where best to position services based on their needs. The system, which combines disparate types of data, such as demographics and air quality, produces a heat map of a city showing where there is a high demand for certain types of

care. Also, by using unsupervised machine learning algorithms for processing geospatial data, MEDLOC serves both as an interface and as a processing system for spatial organization, which provides the capability of visualizing complex relationships between various types of unrelated geospatial data. Synthesizing this information, a vision of stakeholder groups, and empirical data, leads to a visualization that quickly shows the city index while allowing easy access to all the underlying information to guide decision-making.

In this study unsupervised machine learning algorithms, in specific, clustering methods, K-Mean clustering, and DBScan models, were used to generate the visualizations to facilitate collaboration between the different domain experts. Additionally, silhouette analysis was used to determine the optimal number of clustering in the K-Mean algorithm. As a result, silhouette coefficients assessed the optimum number of clusters parameters for each point in the dataset. Thus, silhouette analysis was used as a way to ensure that the dataset was clustered well. To assess the feasibility of installing a new distributed healthcare infrastructure in Chicago, the project used the application to index the city. MEDLOC includes three original (and complementary) sections: explorer, preview, and magnifier, and finally synthesizer. The explorer displays all types of data (features) in the database. The preview and magnifier feature visualizes data visualization, while the synthesizer shows the heat map outcome. In this regard, an area in the city with similar values and/or criteria is colored based on a synthesis of the selected datasets (features). By using unsupervised machine learning models, MEDLOC allows users to see how complex interactions of data types are related. The machine learning algorithms synthesize input from domain experts and user-inspired research and identify locations that would best meet the needs of target users and all stakeholders.

Studio Lab 2 was an ideal start for the concepts and methods of human-centered design, and it gave me a fundamental understanding of interaction design, starting with user-research methods for determining the needs and desires of target users. Studio Lab 2 which was a computational design program took a computer science view of design, applying both the science and art of computing to design, visualization, analysis, evaluation, interaction, and aesthetic expression as well as applying machine learning algorithms, creating a comprehensive database, and finally implementing the required codes in HTML and JavaScript language. Also, Studio Lab 2, was a research and design-based course investigating new design opportunities and critical perspectives at the intersection of design and computation. By the end of the course, I gained experience with a broad scope of design principles and evaluation methodology leading to work with MongoDB, Python, and different types of machine learning algorithms especially unsupervised learning. As I had a perfect experience in all parts of that project, I was interested in continuing some main concepts of that project to be able to publish its result as an application or plug-in for another facility and with a different approach or model. So, I decided to select this topic as my thesis title. Besides, as I had the research method 2 courses in spring 2021, I could complete some research around this topic.

1.2 – INTRODUCTION

Selecting and finding the best location for factories and manufacturing plants with considering sustainability parameters are among the major goals of most local governments, planners, architects, and urban designers around the world since in recent years we have witnessed major changes in the location of the economic activities, with emphasis on geographical disperse of them[1]. That is why determining the best location to serve companies' profitability and sustainability is becoming more crucial every day. To find the optimal location for a new facility, the location-allocation model can be one of the appropriate tools to cover the majority of planners' demands and needs in the most effective way. In this regard, locating manufacturing plants like other facilities and allocating demands to these locations have been implemented within the environment of Geographic Information Systems (GIS) in different studies[2]. If we consider factory as one of this main economic activities, we can say that the location of this activity is of particular importance. Hence, establishing a factory is one of the main economic activities for which finding appropriate locations by considering diverse parameters is of great importance[3].

The location is defined as a position in space and an area defined for a particular purpose, by the webster's Dictionary. Therefore, the problems associated with the location are involved finding a particular position for a specific purpose, function, or activity[4]. Among diverse types of location problems related to the GIS, the measurement of the place of a particular thing can be considered as the most common problem[5]. The measurement of the location of almost everything on the earth can be performed as long as allocating enough time and energy for that and this kind of problem is defined as measurement problems[6]. Searching for the proper location for an activity or purpose is the second common type of problem in this subject which is named the location search problem[7]. Regarding this, it is common for referring to the location search

problem as a facility location problem[8]. To illustrate more about the type of problems in this category, defining a place for one activity, for example, a gas station, or defining places for a group of interrelated facilities or activities, such as fire station for a city, are called single or multiple location problems, respectively[9]. In addition to the previous explanation, It is important to mention that diverse geospatial parameters which return the capital investment, are considered during locating a factory site[10]. In particular, the site location analysis for a manufacturing plant can be analyzed in two steps: legal point of view and site-specific perspective[11]. The word "factory or manufacturing plant" in this work refers to the production facilities of all parts, junkyards, and related shops. To receive factory construction approval from a local government urban management, various standards and complex legal considerations and information must be considered[12]. The complexity and huge quantity are among important characteristics of legal considerations which also are constantly subject to change and update. Therefore, reducing the cost and time in considering corresponding legal issues in defining a site for a factory is of great importance in establishing a factory or manufacturing plant[13]. In the case of using spatial analysis algorithms regarding legal information for locating a factory, Open Geospatial Consortium (OGC) standards are among the great sources that can be used for this purpose[14]. One of the benefits of using OGC for approaching legal information is that many remote users can get access to legal-based processes with spatial perspectives[15]. In work conducted by Peter Schut (2007), after identification of related important processes, only analytical processes with geospatial properties were screened and implemented as OGC[16]. However, this part of the explanation was only for some clarification of the process, and it is out of the scope of this study.

As we know there are lots of visible lands, but the optimum sustainable location for establishing a new factory is Invisible[17]. As lots of parameters and factors are involved in finding

the best location for manufacturing plant, therefore in this thesis, we are developing a computational system and in specific, a plug-in in QGIS as a proposed strategy to assess the feasibility of installing a new factory with considering and investigating important factors in California State. Accordingly, this plugin refers to the algorithm used primarily in a geographic information system, and with the characteristics of a place (Longitude and Latitude for each location) and the analysis it performs, it determines whether the desired place can be a suitable place for building a factory or not?

The location of a manufacturing plant can be described as determining where it will operate most economically and effectively[18]. An ideal Location may not, by itself, guarantee and confirm success but it certainly contributes to the smooth and efficient working of the organization[19]. There may be a need for location selection in the following circumstances:

- When the business is newly started.
- An existing business has outgrown its original facilities.
- A lease expires.
- Other social-economic reasons[20]

and among all of the possible conditions, the main goal of this study is finding the location for the manufacturing plant when the business is newly starting.

SECTION 2: LITERATURE REVIEW

2.1 – FINDING THE PURE LOCATION

The locational search problem is considered one of the important issues in urban planning and urban design. In this section, one of the early changes which greatly affected the development of GIS to be involved with location search problems over large regions is discussed[21]. In the 1970s, the development and use of geographical databases for the storage of environmental data as well as planning them began in many states[22]. Land use and natural resource inventories like the LUNR from the State of New York and the MAGI (Maryland Automated Geographic Information) database of Maryland are examples of these kinds of systems[23]. However, it should be mentioned that from a historical standpoint, the Dangermond of ESRI (Environmental Systems Research Institute) was a principal developer of the MAGI database. During 1970, whether electrical utilities could meet the increasing electricity demand or not, was significantly debated. Also, there were some doubts over the existence of enough power plants[24]. To address these concerns, the project of power plant siting was funded by the state of Maryland, and the design and development of a simple grid-based GIS were created[25]. That should be mentioned that the planning office of the State of Maryland still is in charge of managing and updating this database, however, now that is performed in both raster and vector formats. Locating rights-of-way for roads and transmission power lines also has been evolved using GIS[26]. It is noteworthy to mention that most of these developments in computerized modeling for locating linear facilities have been managed by using some type of GIS, especially the raster type[27]. McHarg (1969) could develop a process that presents a precursor to the classic overlay process in GIS by using design with nature, for which a color acetate map sheet for each basic theme was developed which could be used in the location of a corridor[28]. This map had a shading property which was from light (high

appropriateness) to dark (low appropriateness); by laying all sheets on top of each other and placing them on a light table, those areas with a little color were considered to be the best places in terms of suitability score[29]. This process was looking for a finding the most direct route that is connecting the interested points together by considering the point that they should pass through as many lightly colored areas as possible.

Besides all these tries, four general classes are involved with the most location models[30]: median, covering, capacitated, and competitive. In this classification, a median model indicates locating a fixed number of facilities in a way that we get the minimized average distance from any selected user for their closest facility[31]. Locating facilities in such a manner to cover all or most within some chosen service distance (commonly named the maximum service distance) is involved with covering models. The general idea behind this model is that the more we serve users relatively close to a specific facility, the better our service condition would be[32]. As an example, in ambulance deployment projects, a common goal is to serve at least 90% of the targeted population within 8 minutes. Classic median models are involved with assuming that there is enough source available at each facility to manage all demands. Therefore, we assume that everyone is going to be served by the closest facility[33]. On the other hand, some limitations on possible accomplishments at each specific facility (such as the number of units that can be produced, the quantity of demand that can be assigned, the volume of garbage can be managed, etc.) are considered incapacitated facility models[34]. However, considering the cases in which a competitor can readjust to location decisions that other competitors are making over desired time frames all are discussed with the competition models. To illustrate this model more, for example, if a specific firm locates a new branch in a new location that can exploit a poorly served area of a particular competitor, that competitor may decide to relocate a branch to a new place so that it can

return some of its lost markets[35]. Therefore, in making any location decision, that is very vital to discuss the potential response of competitors trying to prevent loss in their business. From a general perspective, it should be mentioned that the median, covering, and capacitated models are usually addressed as classical optimization models, while competition models are often addressed by game theory and simulation[36]. In this regard, Beaumont (1981), Brandeau and Chiu (1989), Eiselt (1992), ReVelle (1987), and Schilling et al (1993) have provided great resources for location research[37]. For defining a specific location problem, the relationship between the defined points as demand and defined points as a facility is involved very closely. Although the demand is commonly spread over space (such as a census tract, or an apartment), it is often represented as a single point[38]. In more detail, most models are based on the assumption that points are used for representing demands, however, there are diverse significant exceptions to this general assumption in the literature, such as the work of Wesolowsky and Love (1971) in which demand is indicated by continuous rectangular areas[39]. In this regard, a facility site can be represented as points, lines, or areas. For example, in case of a corridor location problem, the facility usually represents a curvilinear facility like a roadway that is connecting two pre-defined points. However, for continuous surface problems, generally, facility locations can be placed anywhere. For network models, facility sites are commonly described as nodes, however, significant concepts have been developed addressing the location along arcs or links. In most cases, both demand and facility sites are represented as discrete points because solution algorithms have been developed for such cases[40]. Based on this logic, a classification of the location model based on the geometric representation of demand and the geometric representation of target facilities was discussed by Miller (1996). For instance, for defining a polygon-polygon location problem which involves the location of a set of polygons to serve a set of weighted polygons, the demands showed as polygons

and facilities presented by polygons can be utilized[41]. For example, this perspective can be found in very-large-scale integration design and production layout. That was argued that GIS can give us a good opportunity to represent location model characteristics such as the shape and size of the facility, and such improvements can give the potential to advance the relevance and flexibility of facility models[42]. GIS can become a vital part of many locations model approaches in the future.

2.2 – LOCATION ALLOCATION MODEL

The roots of Location Science can be found in the literature of location analysis that started with work by von Thunen (1826) about land use allocation and Launhardt (1872) and Weber (1909) about industrial location[43]. The location of industrial facilities was initially focused on locating facilities that needed raw materials, such as iron ore, from local deposits, as well as transportation of the products. As a result, the concept of a location triangle, like the one shown in Figure 1, was derived[44]. As we can see in the image, there was no allocation component in this initial view of the location of production. To supply the market with a known demand, the production facility would have to transport raw materials from a reliable and known source to the industrial facility and then make the product before shipping it to the market[45]. In this case, the only thing left to do was to locate the plant so that shipping costs of raw materials and finished products would be minimized. The main difference between Launhardt and Weber's studies was that Launhardt paid attention to the cost of building the transportation links between the factory and the raw material sources as well as the costs of material and product shipments, whereas Weber focused on transportation costs were related to the Euclidean distance and only assessed and measured the shipping costs for raw materials and products in his analysis[46]. Based on this regard, a true location problem, in this case, can be solved by knowing the interactions when locating the factory. According to Weber, the location triangle represents a simple example of a

more complicated problem of many markets and raw materials. Despite the fact that there could be more markets and specific material requirements, the flow of materials from the factory to its needed materials, and the flow of materials to multiple markets, remained fixed in quantity[47]. Hence, the flow of materials between the factory and the market, and the factory's interactions with multiple markets, remained fixed in numbers, despite the possibility of more markets and materials[48].

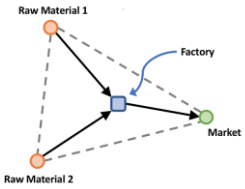


Figure 1: Triangle of Weber - Location

As opposed to pure location problems, location-allocation problems include factors such as the flow of raw materials and products, as well as assignment of services, that cannot be established beforehand and needs to be evaluated as part of the problem of location[49]. When more than one facility is being located, such cases occur. A Weber paradigm indicates that the value or cost of transportation or service is determined by the location of the facility relative to the markets it serves, so both facility placement and market allocation must be optimized simultaneously[50]. Location-allocation problems have been developed in a variety of ways over the past 50 years (Church and Murray, 2009). It is impossible to cover the wide range of location-allocation modeling in this introduction and this issue is out of the scope of this project[51]. Just it is enough to know that two important and commonly used location-allocation models include: the p-median problem (Hakimi, 1964, 1965) and the classic warehouse location problem (Geoffrion and McBride, 1978; Beasley 1988), and companies and public sector agencies solve many location problems related to one of these two issues[52].

SECTION 3: PROBLEM STATEMENT

3.1 – PROBLEM STATEMENT AND HYPOTHESES

Location modeling involves the search for the right location of the desired facilities to support some functions and demands. Examples range from retail site location to the location of multiple large facilities but in this study, we have designed a computational process to find the location which has the capability to build a new factory. As GIS has played a large role in the siting of single facilities, including rights-of-way for roads and transmission lines, therefore it will play a significant role in this study for finding the proper location. This thesis also describes some of the histories of location searches as supported by GIS. It also discusses some of the current impediments to the application of location models, issues associated with the integration of location models into QGIS, and future needs in QGIS functionality to support location models for manufacturing factories with priority given to sites that have the most renewable energies.

3.2 – RESEARCH QUESTIONS

1. Which factors are significant in designing this computational system?
2. How we should consider the factors and parameters which are involved in this analysis to reach an accurate result? What is the process of the analysis and weighting of the data?
3. Which types of machine learning are required to analyze the data which are stored in our database?

3.3 – OBJECTIVE OF THE STUDY

By combining Python code programming with supervised and unsupervised machine learning techniques, this study aims to create a computational system (plug-in) in QGIS that determines whether a location is suitable for building a factory or not by prioritizing the locations that are close to sites that have the most renewable energies in California State.

SECTION 4: DIFFICULTIES OF THIS STUDY

4.1 – COLLECTING DATA

One of the difficulties in this project is related to find some data for creating the database. As I should build a database from scratch for this project, so I need to collect the required data from different open resources. On the other hand, the selected site plan for this project is California state therefore, I should collect data from websites that provide some open-source data for this state. Based on my previous experience, sometimes these data are not available, or they are in the specific format that needs lots of modification, changes, or cleaning which this process usually takes a long time.

4.2 – WEIGHTING DATA

The other difficulty of this study is related to weighting all involved data for analysis. As mentioned before, there are many factors, parameters, and variables that are involved in this analysis but not all factors are equally important. Therefore, all variables which are involved in this analysis should be weighed based on their importance. Due to the initial concept of this thesis, since the priority is to find a location that has the most renewable, clean, and sustainable energy sources, energy resources factors should be considered with a higher weight (with the coefficient of greater importance) than other geographical factors. In addition to the above, according to the existing rules and regulations for locating urban land uses, the importance of economic factors such as employed and unemployed rate will be more important than some other statistical factors such as demographic information. Hence, all factors must be weighted according to the initial idea and to achieve this we consider the environmental layouts more than other layouts. However, the hard part about weighting is the multiplicity of factors involved in the analysis as well as classical mathematical calculations, for which sometimes no similar example was found.

SECTION 5: METHODOLOGY AND CODE IMPLEMENTATION

5.1 – TOPIC MODELING

Before explaining the mechanism and process of the analysis, to ensure the uniqueness of the topic, two steps were performed before finalizing the topic, including A) doing topic modeling B) searching in related articles as an existing state of knowledge and literature review. As the future target of this computational system is designing a plug-in for one of the architectural software, so primitive data, extracted from the food4Rhino website as the main source of these applications and software.

A) In the first step, an account was created on Zyte website for collecting raw data from the food4Rhino website.

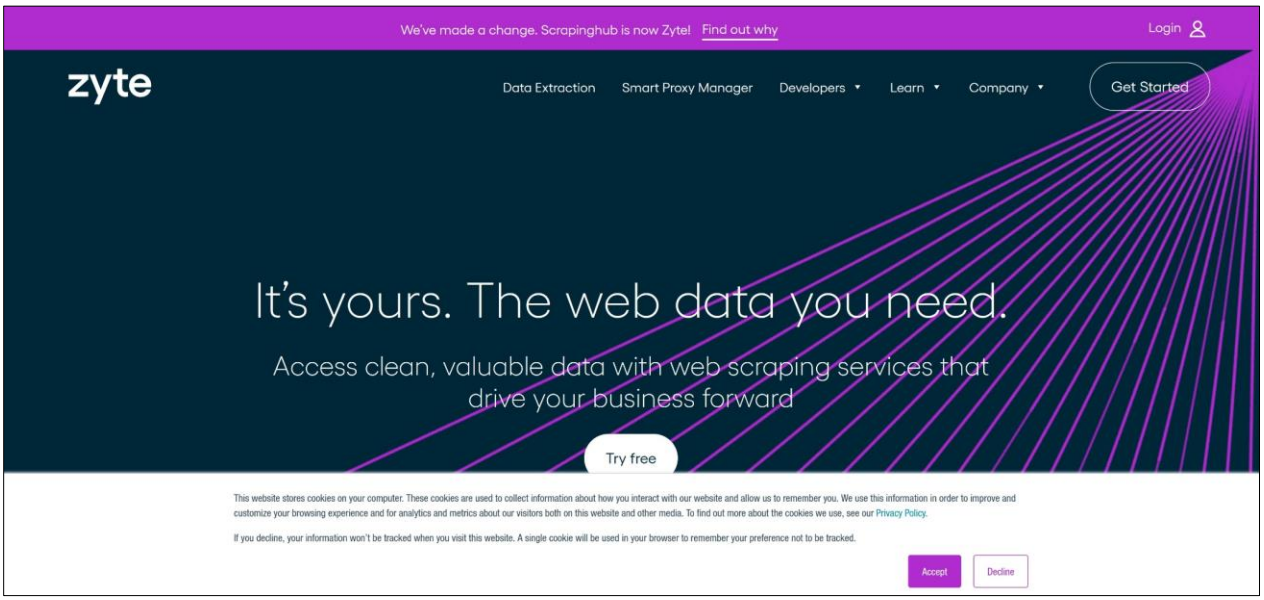


Figure 2: Main page of Zyte website for collecting raw data.

Then raw data (JSON format) was extracted from the food4Rhino website. Data includes all articles and explanations about related plug-ins. (application or plug-ins which have been designed to locate spaces, buildings, facilities, etc. in urban or rural space).

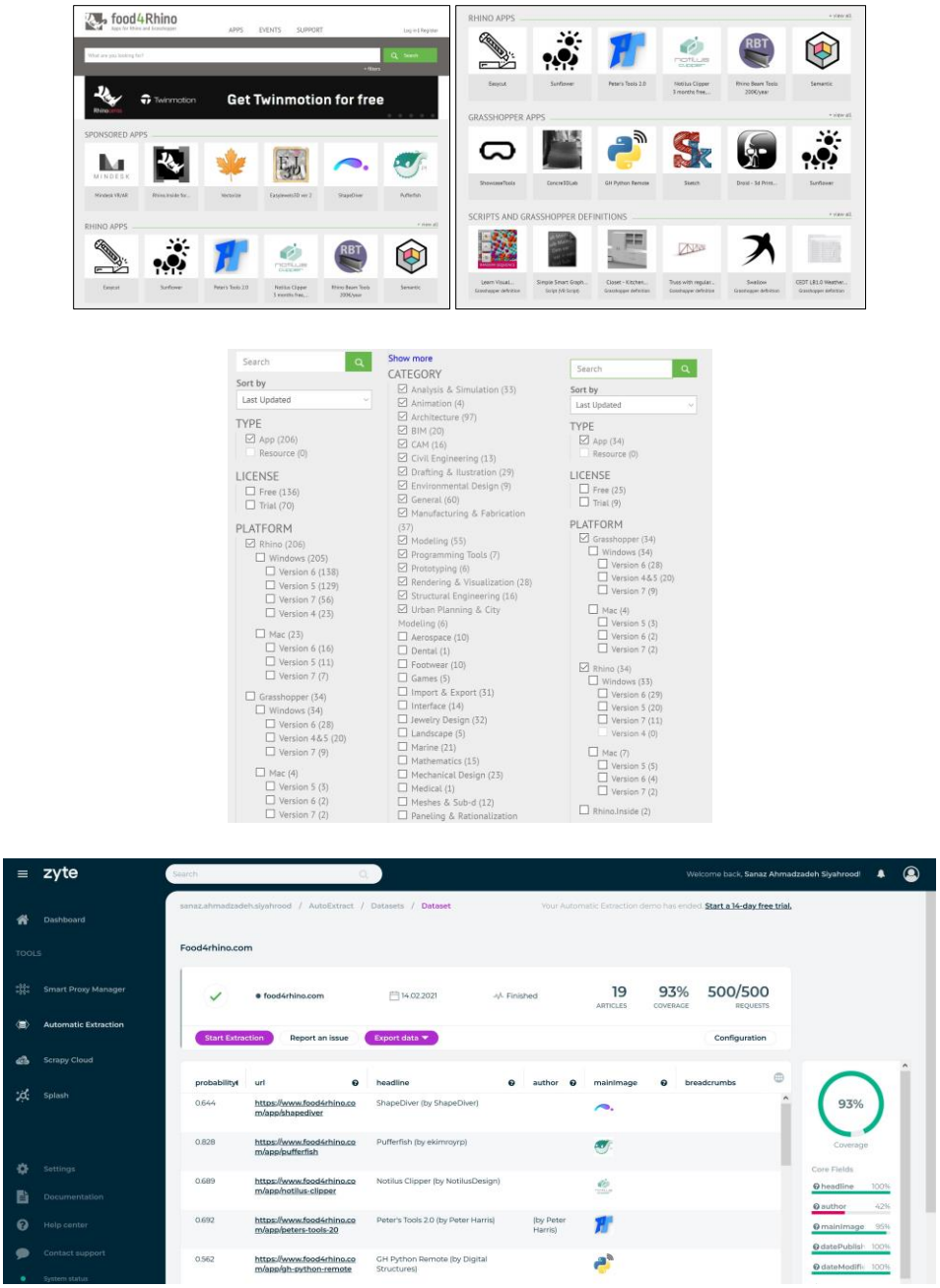


Figure 3: Extracting raw data (JSON format), from the food4Rhino website through Zyte website.

Then topic modeling was done in Jupyter notebook with data that had been extracted from the food4Rhino website - Latent Dirichlet Allocation (LDA) model was used for this purpose. There are two general assumptions applied in the LDA:

1- Documents that have similar words usually have the same topic

2- Documents that have groups of words frequently occurring together usually have the same topic.

Mathematically, two assumptions can be represented in this model as follows:

1- Documents probability distributes over latent topics.

2- Topics probability distributes over words.

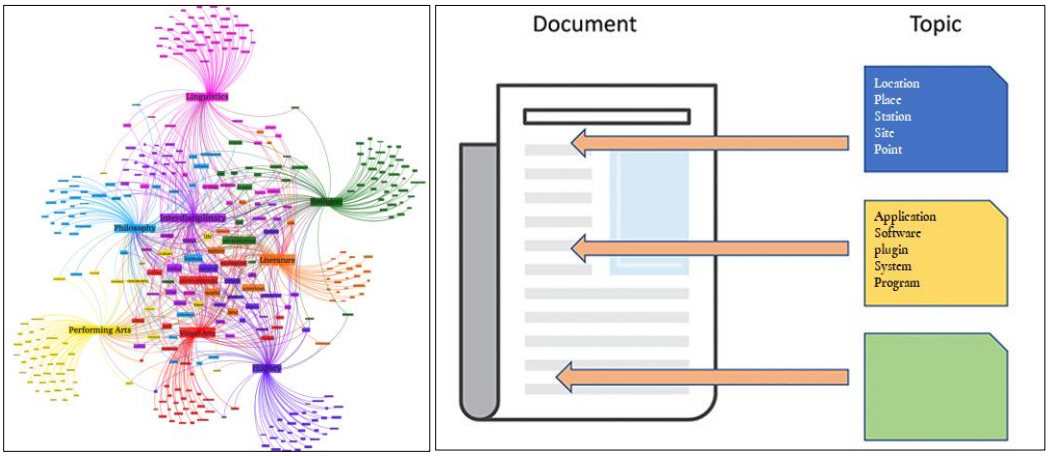


Figure 4: Schematic of topic modeling

In my results, I did not find any keywords related to location, place, automatic, etc.

```
In [1]: import pandas as pd
        import numpy as np
        import json
        import csv
        from pandas import DataFrame

In [2]: df = pd.read_json (r'E:\UNCC\Spring 2021\Directed Independent Study - Prof.Jefferson\Topic Modeling\Jupyter Notebook\raw
        df.to_csv (r'E:\UNCC\Spring 2021\Directed Independent Study - Prof.Jefferson\Topic Modeling\Jupyter Notebook\raw_data.cs

In [3]: reviews_datasets = pd.read_csv(r'E:\UNCC\Spring 2021\Directed Independent Study - Prof.Jefferson\Topic Modeling\Jupyter
        reviews_datasets = reviews_datasets.head(20000)
        reviews_datasets.dropna()

Out[3]:
```

	url	probability	headline	datePublished	datePublishedRaw	dateModified	dateModifiedRaw	inLanguage	
3	https://www.food4rhino.com/app/peters-tools-20	0.692364	Peter's Tools 2.0 (by Peter Harris)	2020-12-07T23:13:46+01:00	Peter's Tools 2.0 (by Peter Harris)	2021-02-10T17:53:52+01:00	2021-02-10T17:53:52+01:00	en	hi
5	https://www.food4rhino.com/app/droid-3d-print-...	0.593732	Droid - 3d Print Slicer and Path Plotter	2018-04-23T11:40:28+02:00	Droid - 3d Print Slicer and Path Plotter (by y...	2021-02-12T09:10:40+01:00	2021-02-12T09:10:40+01:00	en	hi
10	https://www.food4rhino.com/app/laminate-tools	0.505251	Laminate Tools (by Anaglyph)	2021-02-03T18:32:58+01:00	Laminate Tools (by Anaglyph)	2021-02-04T08:54:52+01:00	2021-02-04T08:54:52+01:00	en	hi
14	https://www.food4rhino.com/resource/fir-tanne	0.785236	fir - Tanne (by schnurzelpurz)	2016-04-20T13:43:00+02:00	2016-04-20T13:43:00+02:00	2020-08-14T13:56:30+02:00	2020-08-14T13:56:30+02:00	en	hi
15	https://www.food4rhino.com/resource/beech-buche	0.817336	beech - Buche (by schnurzelpurz)	2016-04-20T13:32:48+02:00	beech - Buche (by schnurzelpurz)	2020-08-14T13:57:02+02:00	2020-08-14T13:57:02+02:00	en	hi
17	https://www.food4rhino.com/resource/maple-ahorn	0.755602	maple - Ahorn (by schnurzelpurz)	2016-04-20T13:47:56+02:00	2016-04-20T13:47:56+02:00	2020-08-14T13:55:52+02:00	2020-08-14T13:55:52+02:00	en	hi
18	https://www.food4rhino.com/resource/mahogany-m...	0.773653	mahogany - Mahagoni (by schnurzelpurz)	2016-04-20T13:45:17+02:00	(by schnurzelpurz)	2020-08-14T13:55:59+02:00	2020-08-14T13:55:59+02:00	en	hi

In [1]:
```
import pandas as pd
import numpy as np
import json
import csv
from pandas import DataFrame
```

In [2]:
```
df = pd.read_json (r'E:\UNCC\Spring 2021\Directed Independent Study - Prof.Jefferson\Topic Modeling\Jupyter Notebook\raw
df.to_csv (r'E:\UNCC\Spring 2021\Directed Independent Study - Prof.Jefferson\Topic Modeling\Jupyter Notebook\raw_data.cs
```

In [3]:
```
reviews_datasets = pd.read_csv(r'E:\UNCC\Spring 2021\Directed Independent Study - Prof.Jefferson\Topic Modeling\Jupyter
reviews_datasets = reviews_datasets.head(20000)
reviews_datasets.dropna()
```

Out[3]:

dified	dateModifiedRaw	inLanguage	mainImage	images	description	articleBody	articleBodyHtml	canonicalUrl	_type
+01:00	2021-02-10T17:53:52+01:00	en	https://static.food4rhino.com /s3fs-public/user...	['https://static.food4rhino.com /s3fs-public/st...	A variety of tools including bill of materials...	This is a collection of scripts that I've writ...	<article>\n \n<p>This is a collection o...	https://www.food4rhino.com /app/peters-tools-20	dict
+01:00	2021-02-12T09:10:40+01:00	en	https://static.food4rhino.com /s3fs-public/user...	['https://static.food4rhino.com /s3fs-public/st...	3D printing related Library including model Sl...	UPDATE: NOW BUILT WITH RHINO 6 / GRASSHOPPER 2...	<article>\n \n<p>UPDATE: NOW BUILT WITH RHINO 6...	https://www.food4rhino.com /app/droid-3d-print-...	dict
+01:00	2021-02-04T08:54:52+01:00	en	https://static.food4rhino.com /s3fs-public/user...	['https://static.food4rhino.com /s3fs-public/st...	Laminate Tools for composite material structur...	is a stand-alone Windows application addressin...	<article>\n\n<p>is a stand-alone Windows appli...	https://www.food4rhino.com /app/laminate-tools	dict
+02:00	2020-08-14T13:56:30+02:00	en	https://static.food4rhino.com /s3fs-public/user...	['https://static.food4rhino.com /s3fs-public/st...	This is a pure solid, procedural material. It ...	This is a pure solid, procedural material. It ...	<article>\n \n<p>This is a pure solid, procedur...	https://www.food4rhino.com /resource/fir-tanne	dict
+02:00	2020-08-14T13:57:02+02:00	en	https://static.food4rhino.com /s3fs-public/user...	['https://static.food4rhino.com /s3fs-public/st...	This is a pure solid, procedural material. It ...	This is a pure solid, procedural material. It ...	<article>\n \n<p>This is a pure solid, procedur...	https://www.food4rhino.com /resource/beech-buche	dict
+02:00	2020-08-14T13:55:52+02:00	en	https://static.food4rhino.com /s3fs-public/user...	['https://static.food4rhino.com /s3fs-public/st...	This is a pure solid, procedural material. It ...	This is a pure solid, procedural material. It ...	<article>\n \n<p>This is a pure solid, procedur...	https://www.food4rhino.com /resource/maple-ahorn	dict
+02:00	2020-08-14T13:55:59+02:00	en	https://static.food4rhino.com /s3fs-public/user...	['https://static.food4rhino.com /s3fs-public/st...	This is a pure solid, procedural material. It ...	This is a pure solid, procedural material. It ...	<article>\n \n<p>This is a pure solid, procedur...	https://www.food4rhino.com /resource/mahogany-m...	dict

In [8]:
```
reviews_datasets['articleBody'][1]
```

```
a custom curve graph mapper, and a multi-threaded morph to twisted box. In addition, there are extra components wh
ich simplify some common grasshopper operations such as testing for equality within a tolerance and rounding to ne
arest numbers. Please email me if you find any bugs. Works with Grasshopper for Rhino 5, Rhino 6, Rhino 7 WIP, and
Rhino Mac. Make sure to read the credits below. Instagrams: @ekim.royrp & @designmorphine\n\nJoin the Pufferfish G
rasshopper forum group here: www.grasshopper3d.com/group/pufferfish\n\nCredits:\n\nI would like to make a special
thanks to David Stasiuk and Mateusz Zwierzycki for their continual support and input on the Pufferfish code and le
tting me constantly bother them. Check out their plugins: Conduit, Cocoon, Tree Sloth, Owl, Anemone, Starling, & S
quid.\n\nDavid Rutten for aide in implementing his Twisted Box Library.\nDaniel Piker for his reference definition
on Quaternion rotation.\nDaniel Abalde for his reference on optimized corner finding.\nPetras Vestartas for his re
ference on RTree mesh welding.\nMahdiyar Esmailbeigi for his reference on mass transformations.\nAndrew Heumann fo
r his reference definition on rectangles by area.\nAldo Sollazzo for his reference definition on discrete variable
s.\n\nI would also like to thank Pavlina Vardoulaki for introducing me to shape blending techniques in Autodesk Ma
ya which inspired many of Pufferfish's components.\n\n-Michael Pryor\n\nGeneral notes\n\nGrasshoppers native "Inte
rpolate Data" component can "Tween" simple data types such as numbers, colors, vectors and points already. Pufferf
ish\'s Tweens of these types differ in 2 ways. The first being that Pufferfish has 3 types of tweens for each: Twe
en Two, Tween Consecutive, and Tween Through which perform the tweens in different ways with the input lists. The
second difference is that Pufferfish uses interpolation types which match nurbs interpolation for simple data type
s. Those types are Linear, Chord, Square Root, and Uniform. Grasshopper\'s "Interpolate Data" component uses Bloc
k, Linear, Cubic, and Catmull. Pufferfish also adds the ability to tween Planes, Surfaces, Meshes, and Twisted Box
es as well as average them.\nGrasshopper already has a native "Tween Curve" component however, it gets odd results
```

In [7]:
```
# Before we can apply LDA, we need to create vocabulary of all the words in our data.
# we could do so with the help of a count vectorizer

from sklearn.feature_extraction.text import CountVectorizer

count_vect = CountVectorizer(max_df=0.8, min_df=2, stop_words='english')
doc_term_matrix = count_vect.fit_transform(reviews_datasets['articleBody'].values.astype('U'))
```

In [15]:
```
# document term matrix
doc_term_matrix
```

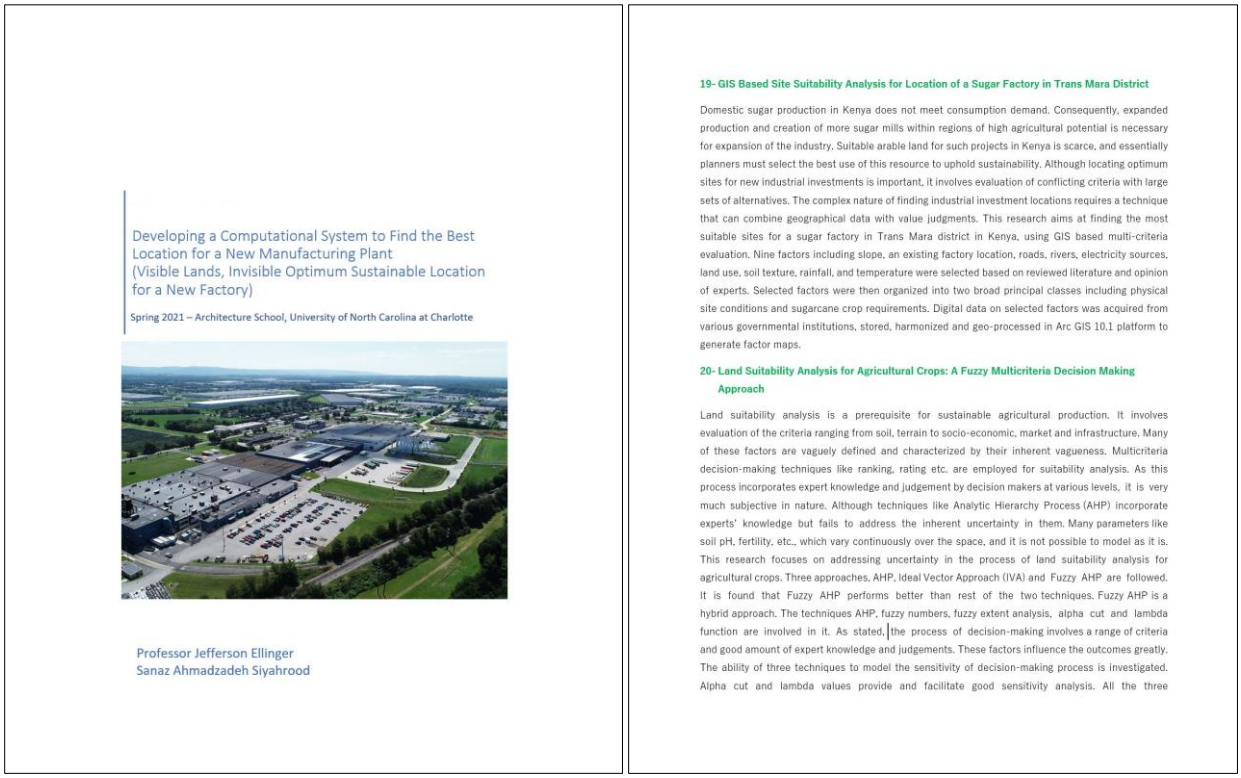
Out[15]:
```
<19x394 sparse matrix of type '<class 'numpy.int64'>'
        with 1709 stored elements in Compressed Sparse Row format>
```

In [16]:
```
# We will use LDA to create topics along with the probability distribution for each word in our vocabulary
# for each topic

from sklearn.decomposition import LatentDirichletAllocation

LDA = LatentDirichletAllocation(n_components=5, random_state=42)
LDA.fit(doc_term_matrix)
```

Out[16]:
```
LatentDirichletAllocation(n_components=5, random_state=42)
```

Figure 5: Results of Jupyter notebook, Topic modeling – LDA method

B) Then I started to find the literature review of the topic – Search in different websites and collecting papers and books as references. In this search, I found some articles which tried to find the optimal place for the desired facility with classical mathematical methods, so I made sure that the topic of my thesis is relatively unique, and I assume that there is no study exactly similar that I have the plan to do. Since among all of the papers that I investigated, I did not find any computational system that can find the solution with help of machine learning algorithms with the priority of considering the sites that have the most renewable energies, the topic became finalized.


Developing a Computational System to Find the Best Location for a New Manufacturing Plant (Visible Lands, Invisible Optimum Sustainable Location for a New Factory)

Spring 2021 – Architecture School, University of North Carolina at Charlotte

Professor Jefferson Ellinger
Sanaz Ahmadzadeh Siyahrood

19- GIS Based Site Suitability Analysis for Location of a Sugar Factory in Trans Mara District

Domestic sugar production in Kenya does not meet consumption demand. Consequently, expanded production and creation of more sugar mills within regions of high agricultural potential is necessary for expansion of the industry. Suitable arable land for such projects in Kenya is scarce, and essentially planners must select the best use of this resource to uphold sustainability. Although locating optimum sites for new industrial investments is important, it involves evaluation of conflicting criteria with large sets of alternatives. The complex nature of finding industrial investment locations requires a technique that can combine geographical data with value judgments. This research aims at finding the most suitable sites for a sugar factory in Trans Mara district in Kenya, using GIS based multi-criteria evaluation. Nine factors including slope, an existing factory location, roads, rivers, electricity sources, land use, soil texture, rainfall, and temperature were selected based on reviewed literature and opinion of experts. Selected factors were then organized into two broad principal classes including physical site conditions and sugarcane crop requirements. Digital data on selected factors was acquired from various governmental institutions, stored, harmonized and geo-processed in Arc GIS 10.1 platform to generate factor maps.

20- Land Suitability Analysis for Agricultural Crops: A Fuzzy Multicriteria Decision Making Approach

Land suitability analysis is a prerequisite for sustainable agricultural production. It involves evaluation of the criteria ranging from soil, terrain to socio-economic, market and infrastructure. Many of these factors are vaguely defined and characterized by their inherent vagueness. Multicriteria decision-making techniques like ranking, rating etc. are employed for suitability analysis. As this process incorporates expert knowledge and judgement by decision makers at various levels, it is very much subjective in nature. Although techniques like Analytic Hierarchy Process (AHP) incorporate experts' knowledge but fails to address the inherent uncertainty in them. Many parameters like soil pH, fertility, etc., which vary continuously over the space, and it is not possible to model as it is. This research focuses on addressing uncertainty in the process of land suitability analysis for agricultural crops. Three approaches, AHP, Ideal Vector Approach (IVA) and Fuzzy AHP are followed. It is found that Fuzzy AHP performs better than rest of the two techniques. Fuzzy AHP is a hybrid approach. The techniques AHP, fuzzy numbers, fuzzy extent analysis, alpha cut and lambda function are involved in it. As stated, the process of decision-making involves a range of criteria and good amount of expert knowledge and judgements. These factors influence the outcomes greatly. The ability of three techniques to model the sensitivity of decision-making process is investigated. Alpha cut and lambda values provide and facilitate good sensitivity analysis. All the three

Figure 6: A literature review of the study

5.2 – WORK OUTLINE

After making sure about the uniqueness of the topic it is time to talk about the methodology for designing this system. In the methodology section, the logic which has been used for all of the calculations is examined in detail. The following overview roadmap summarizes the requirements and logic in each step before diving into the depths of each section.

After searching in existing references as the literature review of this topic, most of the parameters which could be considered in this design were finalized as a checklist since those should be considered both in collecting data and building the database and in designing the plug-in. Accordingly, the required data were collected as much as possible, based on the checklist that was created. This checklist consists of 119 items, but efforts are being made for future extensions of this plugin to find more parameters to be close to the reality for finding the best location. All the data were mapped in QGIS due to two reasons; first in order to exploratory data and getting more familiar with the California state and second since some analysis can be done in the QGIS environment. After this step, a comprehensive dataset was created and the logic of the core part of the plugin for calculations was designed. This part was so important because Python code was written based on the logic of the computing core part of the system. So, after finalizing the concept of calculation its Python code was written. Also, to reach the final result, supervised and unsupervised machine learning algorithms were used which are including clustering and binary classification through the Pycaret library. After completing all of these steps a plugin was created in QGIS and its user interface was designed based on the initial concept of this plugin. In the last step of implementation, all of the written codes were deployed to the main Python code of the plugin to test the entire system. In the end, with knowledge about the details of all parts, a schematic diagram was created to show the mechanism and backend workflow of the entire system. The novelty of this study is using different types of machine learning algorithms for finding the potential solutions, instead of using classical mathematic computing which has been used in previous studies. Besides, in this system, considering the locations which are close to the resources of renewable energies is a priority and is the other novelty of this study. That’s why

California state has been selected as a target site plan to test and analyze the sample data. In the following sections, all parts will be explained in detail.

5.3 – CALIFORNIA STATE – SELECTED SITE FOR ANALYSIS

California is the most populous state in the nation, has the largest economy, and is rich in energy resources, and produces more renewable energies than any other state in the United States. Based on the news of Los Angeles Times on April 29, 2021, California hit nearly 95% renewable energies and by far is the leading solar market in the US thanks to the numerous days of clear, sunny weather and that is why it is known as the Land of the Sun and Golden State. Also, the third-largest state by land area, California stretches two-thirds of the way up the U.S. (1,000 miles long and 500 miles wide) hence it can be one of the best places to use lands for energy purposes. Although California is the second state (Texas is the first) in total energy consumption, on the other hand, the state has one of the lowest per capita energy consumption levels in the United States. The transition from nonrenewable to renewable energy in California has had a significant impact on the US' energy usage. As a result of these policies, it has challenged other states to decrease their reliance on fossil fuels. Furthermore, it has rendered our energy supply safer, healthier, and more sustainable, as well as cost-effective over the long term. In addition to being one of the most beautiful states in the country, California is also a major source of renewable, sustainable energy. This state is not only the second-largest provider of renewable energy in the country (Washington is #1), but it has largely eliminated the use of coal in its electricity production operations. This state is not only the second-largest provider of renewable energy in the country (Washington is #1), but it has largely eliminated the use of coal in its electricity production operations. In order of promoting renewable energy, this state has moved away from energy resources that are less safe and towards more environmentally friendly ones[53]. As a result of

California's growing awareness of clean energy potential, the state's government has finally recognized the downside of fossil fuels.

Does California use renewable energy?

It is for two primary reasons that California is a big proponent of renewable energy. Sunshine and wind are abundant in this state due to its geographical location. First, renewable energy reserves will continue to be abundant in California so long as the wind blows and the sun shines. Despite its diverse climate, California experienced the fourth hottest average temperature in the U.S. in 123 years in 2018 and even though the state is prone to droughts, its rivers and lakes make it ideal for hydropower generation. Secondly, local government initiatives and incentives play an important role in California's success in using renewable energy. Often, Native American tribes in California receive compensation from the Department of Energy (DOE) when installing energy projects on tribal lands[54]. Further, the California Solar Initiative provides residents with grants and rebates as a way to encourage their rooftop solar installations. Hence, California's 2019 building energy efficiency standards mandate that solar photovoltaic systems be installed on all new homes starting in 2020.

What are California’s primitive and mani sources of renewable energies?

Solar energy, wind energy, hydro (hydroelectric/hydrogen) energy, tidal energy, geothermal energy, biomass energy, and ocean energy are currently the most popular renewable energy sources in this state. Solar power is a major source of energy in California. Most of California's solar power comes from its southeastern deserts. The state's biggest solar thermal and solar photovoltaic (PV) plants are located in southeastern, and there are also solar PV plants scattered throughout the rest of the state. Around one-seventh5 of the state's net generation in 2018

was generated by utility-scale PV and solar thermal facilities. Solar energy provided nearly 25% of California's net electricity generation, including small-scale generation. Besides wind power, solar power is one of the most commonly used renewable energy sources in California State. It is easy for professionals to install and maintain solar panels on public buildings. In parts of the US where there is a lot of sunlight each day, solar energy is also a reliable alternative to fossil fuels. Wildfires are common in California because the state is generally very warm. In recent years, acres of natural land have been destroyed by wildfires. There were 4,257,863 acres6 burnt, 33 fatalities, and 10,488 destroyed or damaged structures in 2020 so this year was especially bad. As a state which is susceptible to fires, some might argue that it is even more vital that California continues to utilize renewable energy so diligently.

What are the best renewable energy sources in California?

California is not only a leader in solar energy production, it is also a leader in biomass energy production. During the height of the biofuels industry, California's biomass power plants produced 800 megawatts of electricity from 66 direct combustion biomass plants. Approximately 140,000 tons of wood pellets are produced every year in two notable pellet facilities in California. For electricity generation and heating, these wood pellets are manufactured mostly from recycled wood waste. In addition to solar, geothermal, and biomass, California has a diverse energy portfolio. Despite the state's sunny climate, it also has large-scale wind farms. According to the 2019 Wind Energy Capacity report, California ranks fourth among all US states7. Oklahoma, Texas, and Iowa were the states with the most wind power. California is the world's sixth-largest economy, with a GDP (gross domestic product) of $2.6 trillion, ranking higher than the entire country of India. California State has approximately the same number of residents as Poland, with about 39 million residents. Due to its large population, the state’s energy consumption is very high.

The state of California is leading the way in sustainability and clean energy use in the United States. A growing number of wind and solar farms are being constructed throughout the state and country, competing effectively with nonrenewable power companies. Most wind turbines are located in six areas: 1- Altamont area, 2- East San Diego County, 3- Pacheco, 4- Solano, 5- San Gorgonio, and 6- Tehachapi. The size and capacity of modern wind turbines have increased by 30 times compared to older models, and technological advancements have made wind power an economically viable and grid-compatible energy source. Among the major sources of electricity in the US, wind energy is the fourth largest after natural gas, coal, and nuclear power. Electric cars are selling quickly, and households are switching to renewable energy providers, resulting in lower battery prices. There is a growing trend towards a renewable lifestyle in California, and if the rest of the nation follows in its footsteps, we'll no doubt see an improvement in the state of the environment as a whole. In terms of using sustainable energy, California is arguably the top US state, setting the bar exceptionally high for the rest of the nation.

According to the above information, it appears that California state has a good potential for developing manufacturing plants that can use renewable energy when they are producing their products. California should build more solar panels than it can regularly consume, according to new research published in the peer-reviewed journal Solar Energy and by this way, it keeps electric power prices low on a power grid made up of renewable energy so this issue would be a problem that can be solved. By 2030, California plans to have 60% of its power grid powered by renewable energy, as well as a long-term goal of 100% environmentally friendly energy, which this study also aims for.

5.4 – DATA COLLECTION AND PREPARATION

After searching in existing references as the literature review of this topic, I tried to finalize the important factors and parameters which I should consider for designing this system. Based on the information that was found during the previous step, a list of variables was provided. This list as a checklist includes the most important required parameter that should be considered in collecting data and building the database of this system. This checklist, which includes 119 items, contains the basic requirements of this system, and more parameters can be added to it in the future. Also, as mentioned in the previous section California State was selected for collecting open-sources data and testing the initial results since the priority of this computational system is to find the best location among the sites which have the most renewable energies. Hence, the most important parts of this checklist are the data that can show the distribution of different types of renewable energies around California State.

Table 1: Checklist of required parameters and variables. [65]

NO.	Parameter	Status
1.	Access to Your Customers and Supply Chain	✓
2.	Labor Supply	✓
3.	Financial Incentives	✓
4.	Local Geography and Climate	✓
5.	Environmental and Ecological Problems	✓
6.	Distance	✓
7.	Expansion Potential	✓
8.	Accessibility	✓
9.	Security	✓
10.	Competition	✓
11.	Business Rates	✓
12.	Skill base in the area	✓

13.	Potential for growth	✓
14.	Proximity to the Market	✓
15.	Rent Price	✓
16.	Analyze the Demographics.	✓
17.	Infrastructure and Accessibility	✓
18.	Distribution Network	✓
19.	Competition, to Be Closer or Not to Be?	✓
20.	Remote Business Location	✓
21.	Style of Operation	✓
22.	Demographics	✓
23.	Foot Traffic	✓
24.	Parking and Accessibility	✓
25.	Competition	✓
26.	Site's Image and History	✓
27.	Site Characteristics	✓
28.	Size, shape, topography, and room for future expansion; site buffer should provide protection from residential and commercial neighbors. Industrial parks often provide good locations for multiple users with similar buffer and utility requirements.	✓
29.	Elevation and risk of flooding.	✓
30.	Geotechnical status including bearing capacity and verification of seismic risk and water table elevation.	✓
31.	Environmental status typically verifying wetlands, if present, will not inhibit construction footprints and that ground contamination from previous users or other sources does not exist; the absence of endangered species and archeological remnants should be verified.	✓
32.	Site access confirming acceptable ingress and egress for employees, inbound raw materials, and outbound finished products.	✓
33.	Proximity to OEMs, Suppliers	✓
34.	Availability of Utilities	✓
35.	Labor draw	✓
36.	Disaster risk	✓
37.	Business climate	✓
38.	Support of manufacturers by local citizens and the political and business community	✓
39.	A business and political community that does not support labor union activity.	✓

40.	Acceptance of foreign firms and local support for cultural and educational activities	✓
41.	The proximity of educational institutions that support manufacturers that compete on a global basis.	✓
42.	Start-up and Operational Costs	✓
43.	Value of Incentives	✓
44.	The goods the plant will produce	✓
45.	The number of goods the plant will produce	✓
46.	Five years of production planning	✓
47.	Future growth expectations	✓
48.	Local geography	✓
49.	Daily operations	✓
50.	Utility and water costs	✓
51.	Distance	✓
52.	Environmental issues	✓
53.	Proximity to raw materials and suppliers	✓
54.	Access to top technical talent	✓
55.	Access to low-cost labor	✓
56.	Experienced workforce	✓
57.	Lower tax and regulatory burdens	✓
58.	State-of-the-art production facilities, equipment, and processes	✓
59.	Easy accessibility to major airports and transportation hubs	✓
60.	Comfort with cultural norms, business practices, and language	✓
61.	More reliable lead times	✓
62.	Greater ability to easily scale production	✓
63.	Decreased shipping time	✓
64.	Fewer import and customs hurdles	✓
65.	Favorable fiscal policies	✓
66.	Political stability	✓
67.	Smaller environmental impacts	✓
68.	Local Geography and Climate Play a Large Role in Your Manufacturing Plant Site Selection	✓

69.	Environmental and Ecological Issues	✓
70.	Governmental Policy and Political Climate	✓
71.	Distance	✓
72.	Business Costs	✓
73.	Potential for Expansion	✓
74.	The alternate geographic trading areas.	✓
75.	Determine the type of location.	✓
76.	Literature of the population	✓
77.	Trading factors	✓
78.	Accessibility	✓
79.	Amenities	✓
80.	Geographic Challenges	✓
81.	Environmental Considerations	✓
82.	Economic Benefits and Challenges	✓
83.	Supply Chain Infrastructure / Logistics and Access to Customer Markets	✓
84.	Effective Corporate Tax Rates and Incentives	✓
85.	Tax Domiciles, Exchange Rates and Economic Conditions	✓
86.	Business Regulatory Regimes and Customs/Trade Agreements	✓
87.	Business Operating Costs	✓
88.	Facility / Real Estate Costs	✓
89.	Utility Costs	✓
90.	Labor Unions and Wage Costs	✓
91.	Employee Benefits such as Healthcare, Pensions, Unemployment, Insurance	✓
92.	Network Effect / Industry Clusters / Talent and Knowledge Base	✓
93.	2. Business Transparency and Criminal Activity	✓
94.	Cost of Living for Employees	✓
95.	Quality of Life Consideration	✓
96.	Health and Safety	✓
97.	Educational Institutions	✓
98.	Cultural Institutions, Language, Religious Worship	✓

99.	Diversity and Inclusion	✓
100.	Recreation and Leisure	✓
101.	States with Lower Tax Rates	✓
102.	States with Low Cost of Living	✓
103.	Open Shop States	✓
104.	Raw material: Availability of natural resources that can be used as raw material.	✓
105.	Technology: To turn the resource into an asset with value.	✓
106.	Power: To utilize the technology.	✓
107.	Labor: Human resources in the area who can function as labor to run the processes.	✓
108.	Transport : Road/rail connectivity.	✓
109.	Storage and warehousing.	✓
110.	Marketing feasibility.	✓
111.	Characteristics of land and soil.	✓
112.	Climate.	✓
113.	Precipitation and water resources.	✓
114.	Vulnerability to natural resources.	✓
115.	Capital investment.	✓
116.	Availability of loans.	✓
117.	Investment climate.	✓
118.	Government policies/regulations.	✓
119.	Influence of pressure groups.	✓

Therefore, the required data was collected based on a checklist of the previous step. Based on the search 180 different variables and parameters have been found from different open sources of data for California state which include 4 main categories as following:

Layouts: 1- Movement and transportation, 2- Power and energy resources, 3- Climate and land, 4- All other information. Also, the format of all data and layouts are SHP which is a shape file, and also CSV.

Figure 7: List of all collected data

5.5 – EXPLORATORY DATA ANALYSIS THROUGH GIS

To develop a successful machine learning model, adequate, high-quality, and clean data are needed and required. Data collected from different open sources were processed and analyzed so they could be used for training the model. Also, cleaning the data and processing were done to handle null values in all of the CSV files. The dataset consists of various layouts with different categorical and numerical features. To get familiarized with the data and the features, we performed exploratory data analysis through QIS software. It is important to mention that all of the significant data were imported to the QGIS software to check and visualize the specific layout. Also, different analysis was done for some of the layouts with the help of algorithms in QGIS. All of the exported layouts from QGIS are as following:

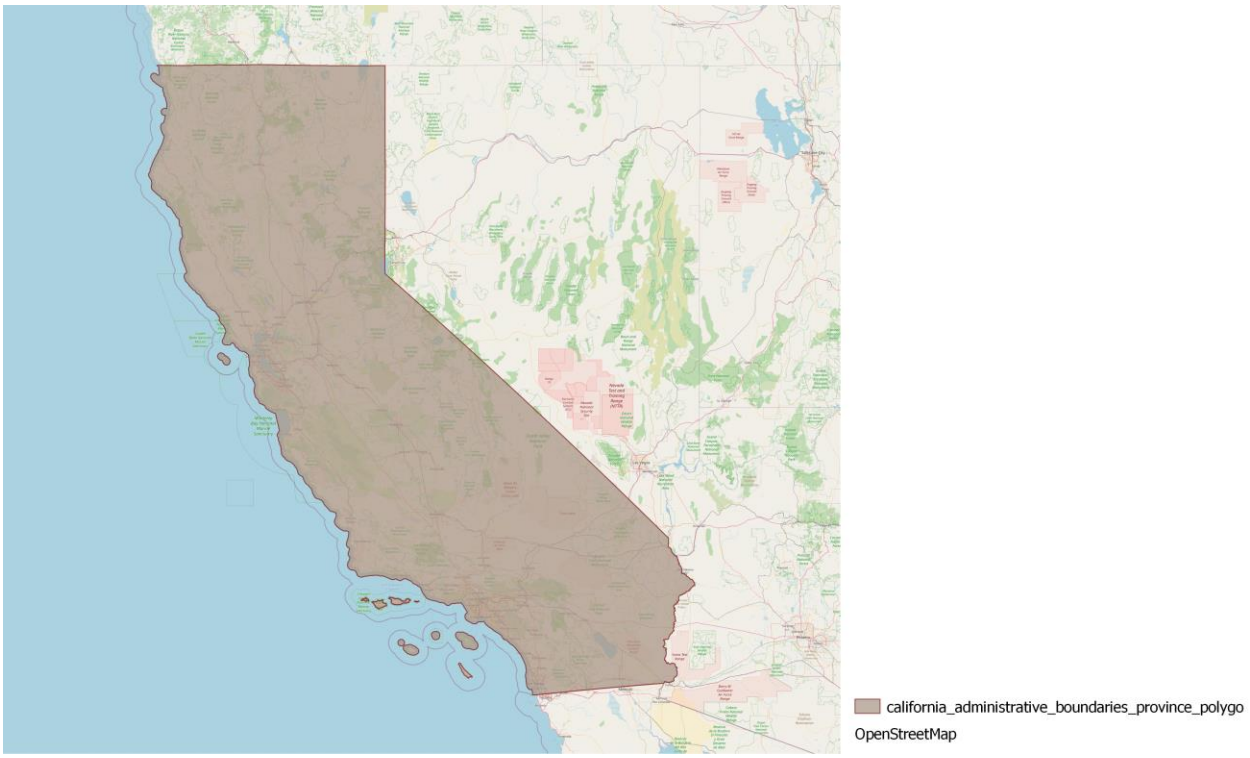


Figure 8: Administrative boundary of California [65]

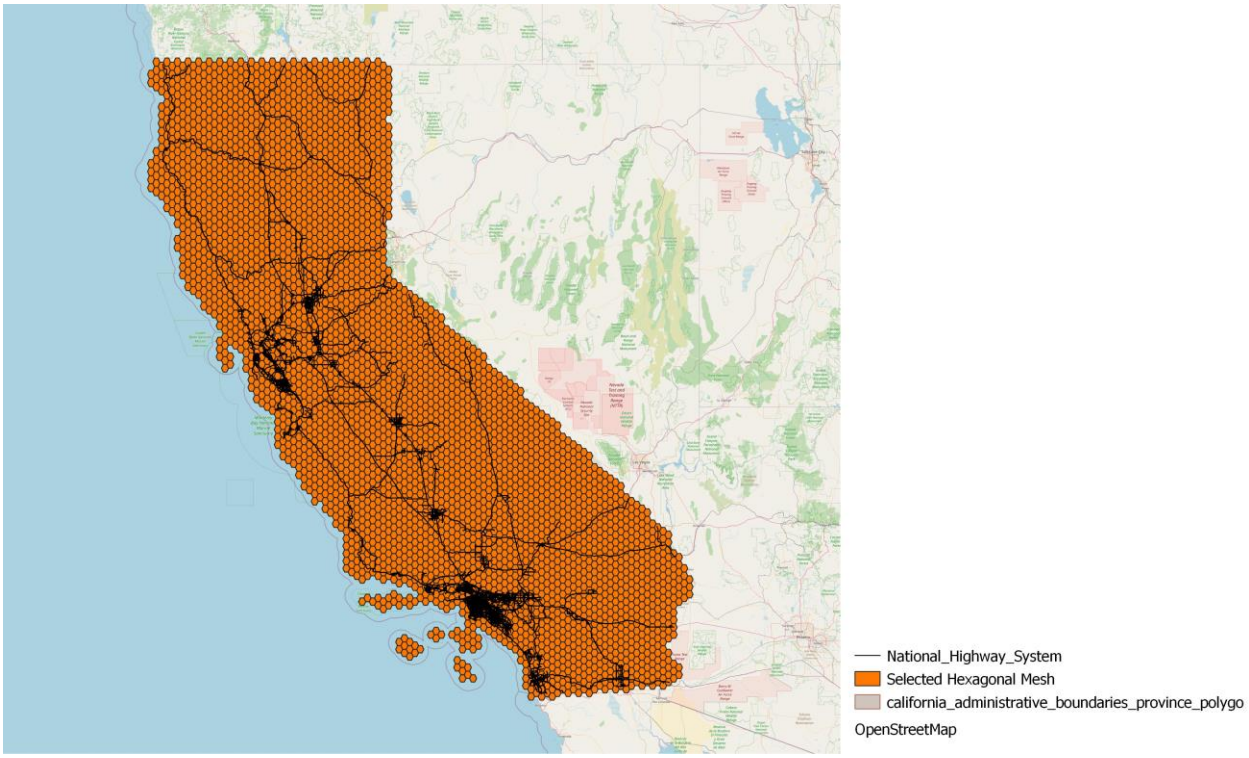


Figure 9: National highway system and hexagon mesh (1.5 mile) of California [65]

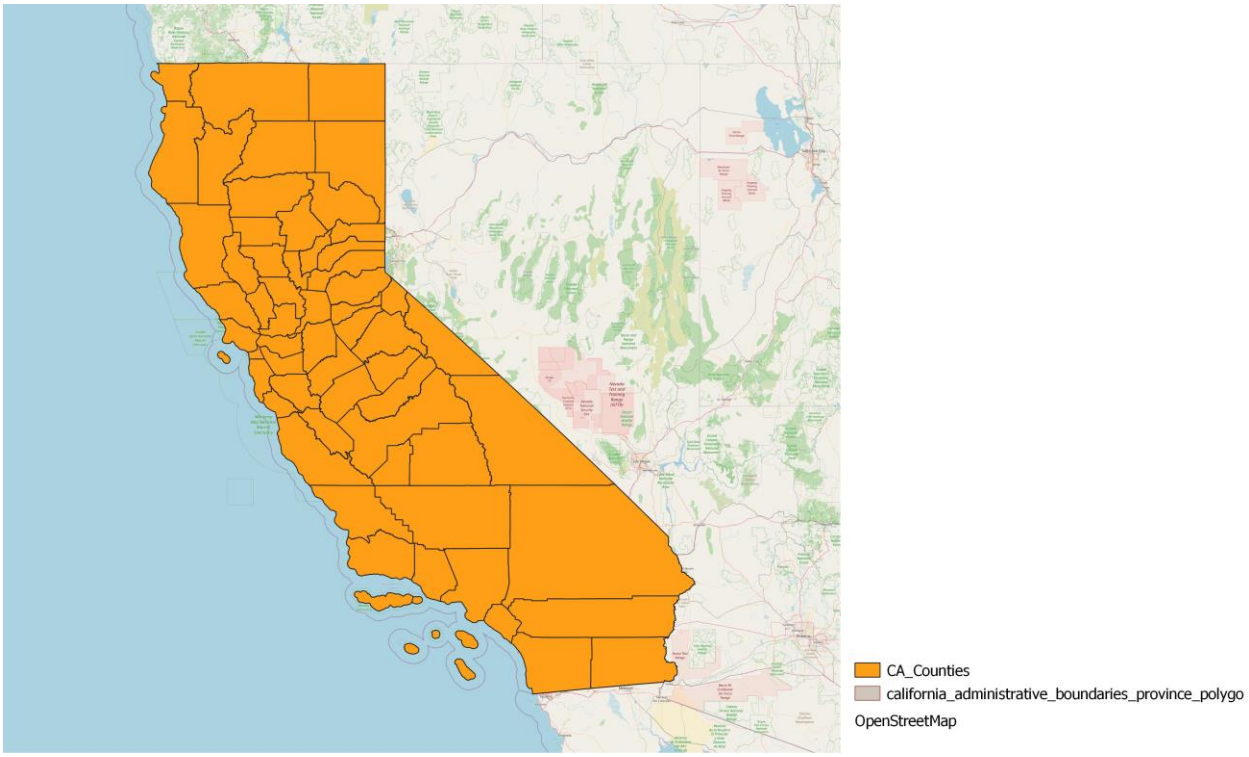


Figure 10: boundaries of California's Counties [65]

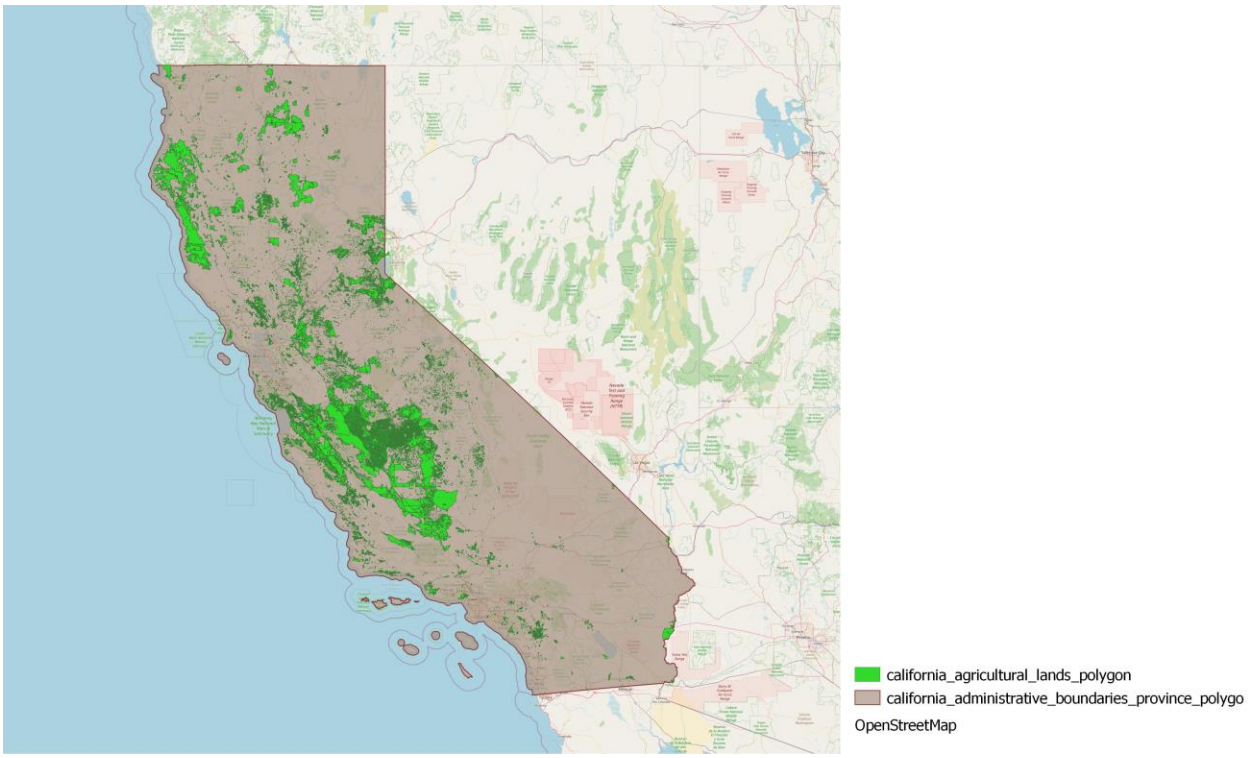


Figure 11: California agricultural lands polygons[65]

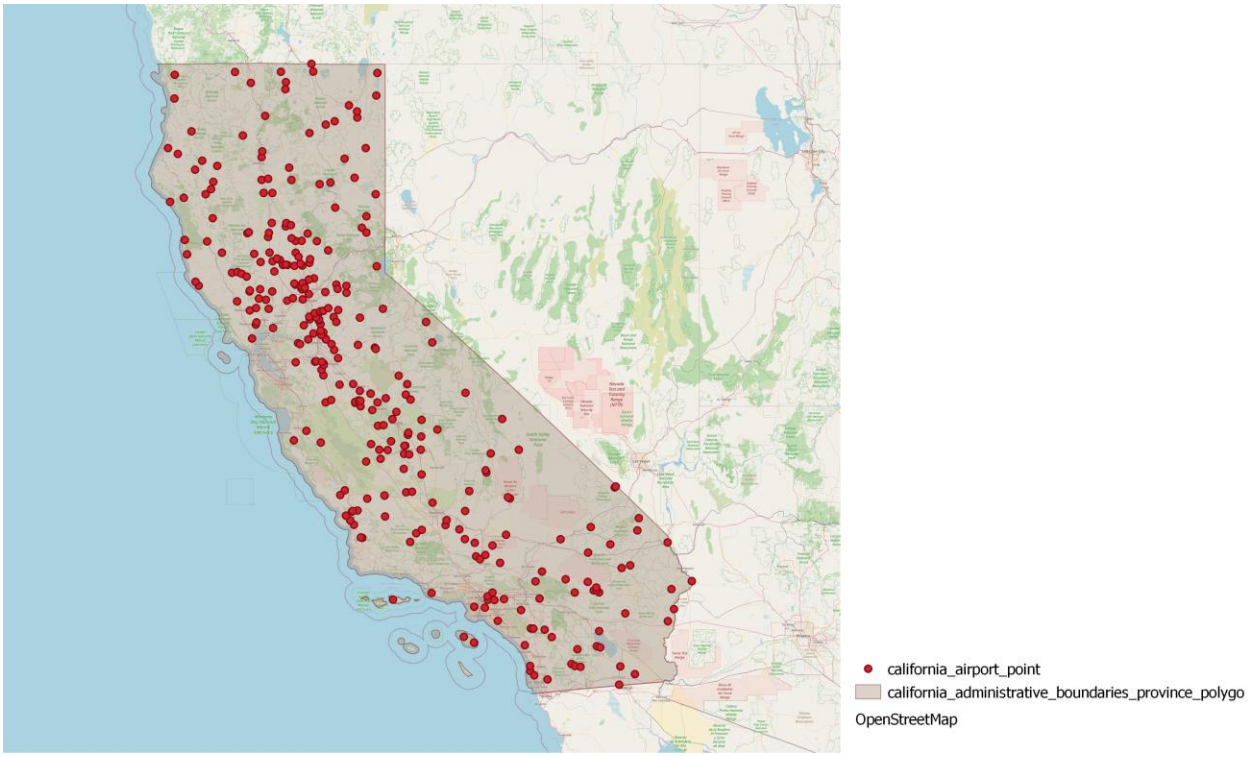


Figure 12: California's airports' points [65]

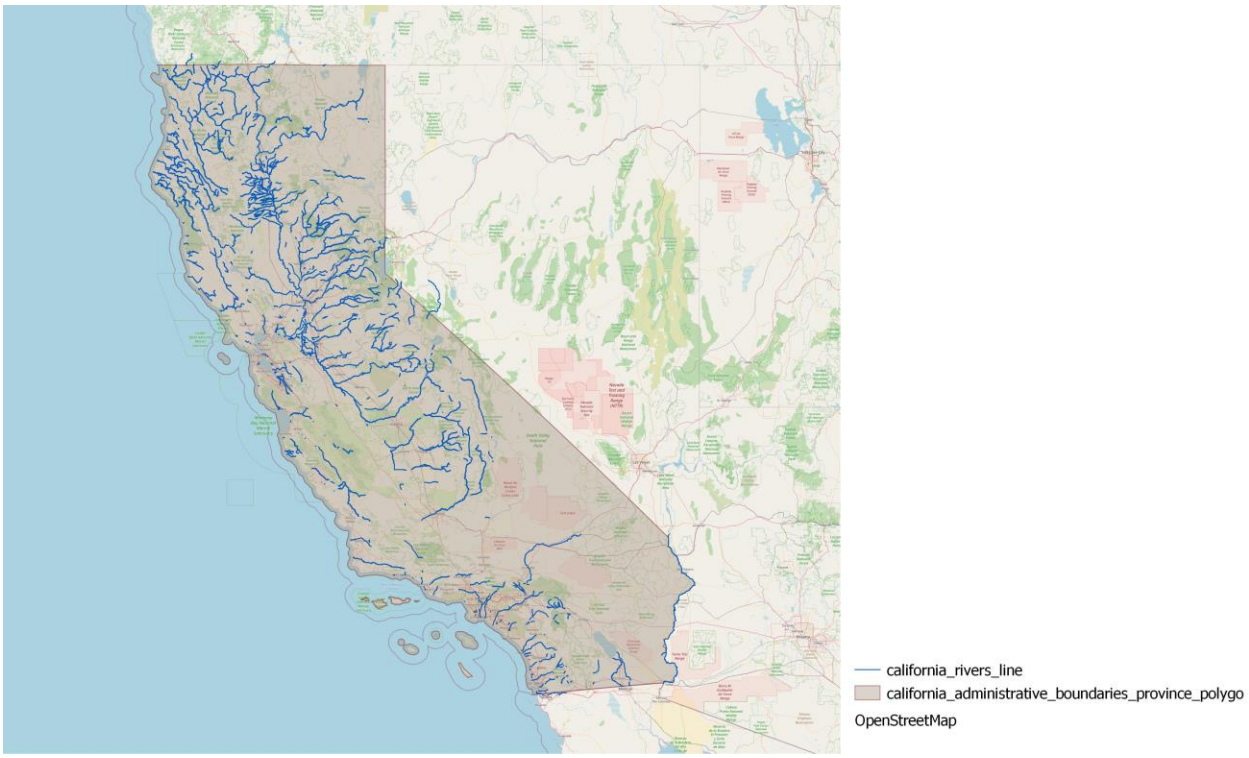


Figure 13: California's rivers line[65]

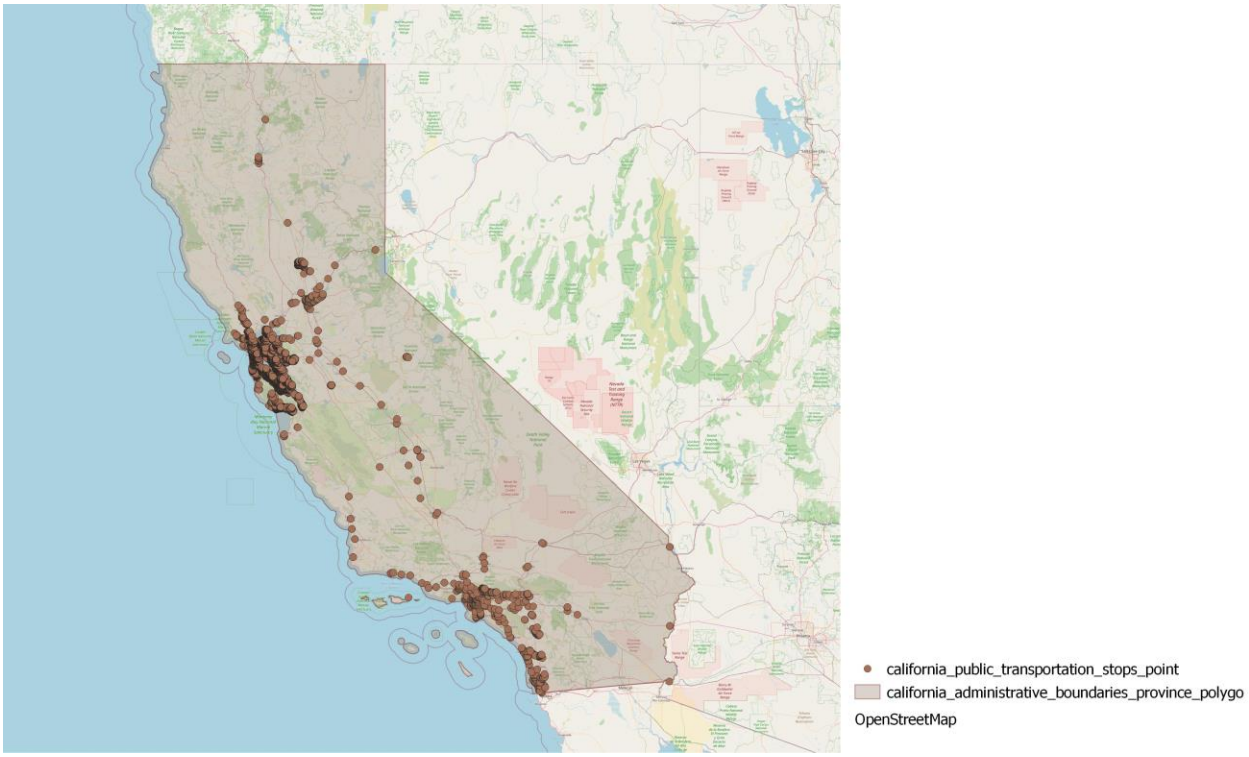


Figure 14: California's public transportation stop points[65]

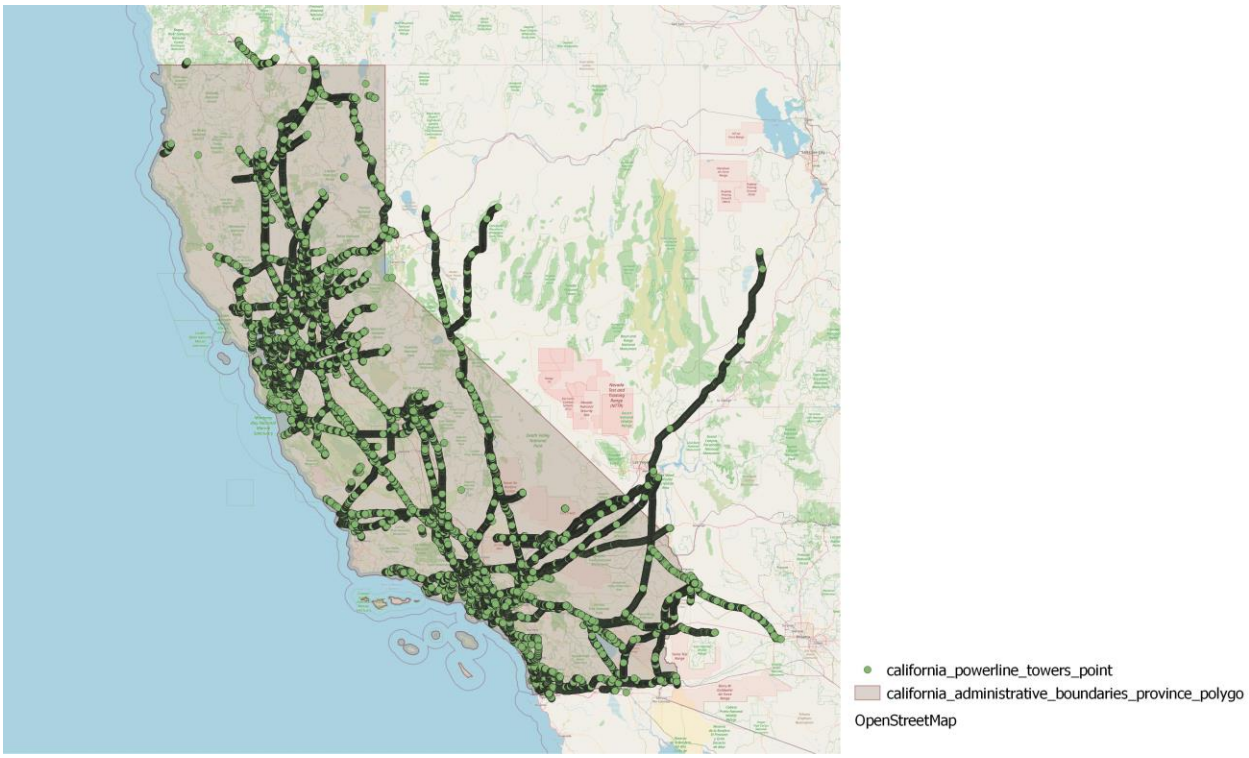


Figure 15: California's powerline towers points[65]

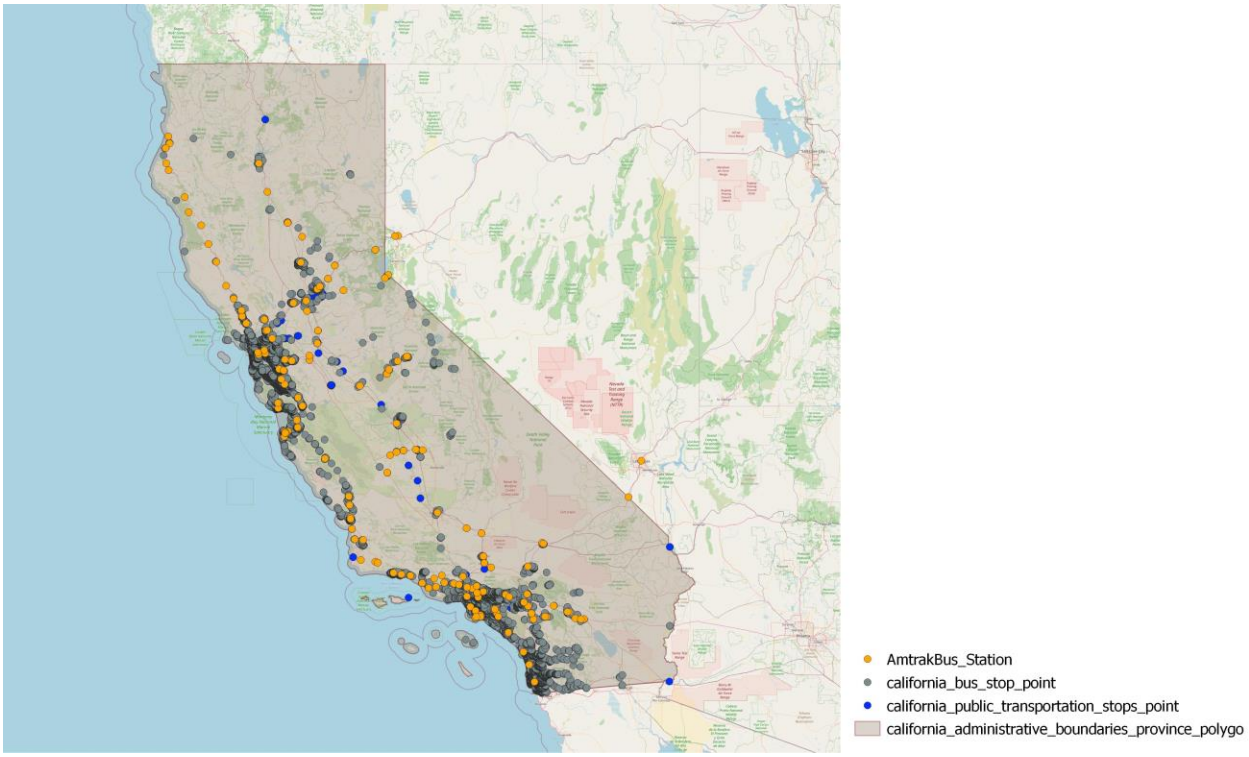


Figure 16: California's bus stop points & AmtrakBus stations [65]

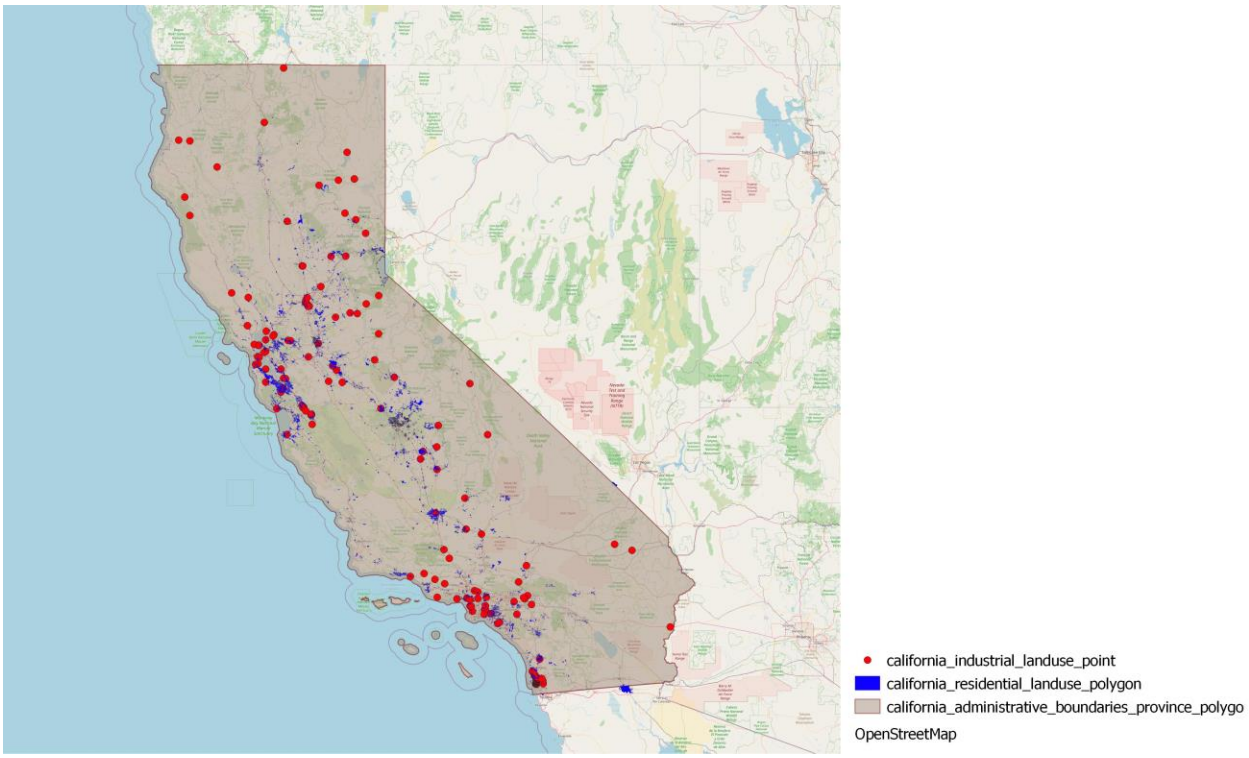


Figure 17: California's industrials land-use points & residential land use polygons[65]

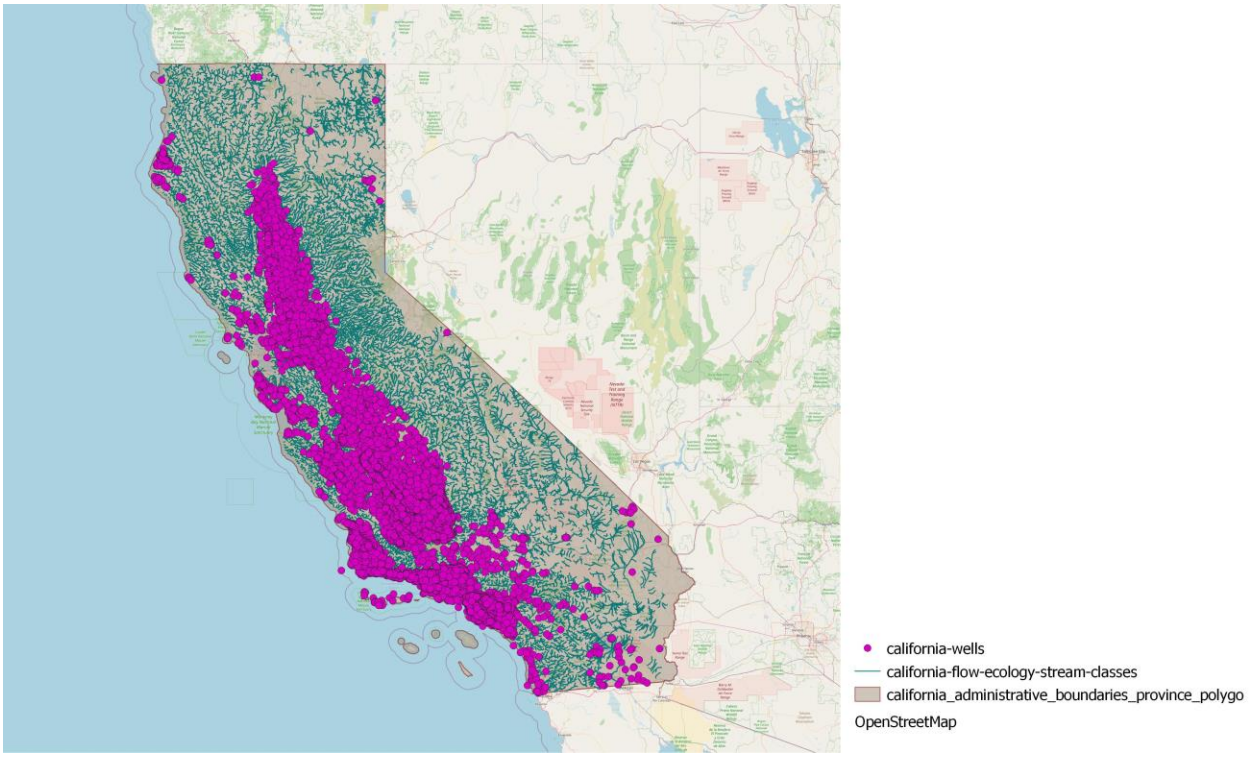


Figure 18: California's wells & flow ecology stream classes[65]

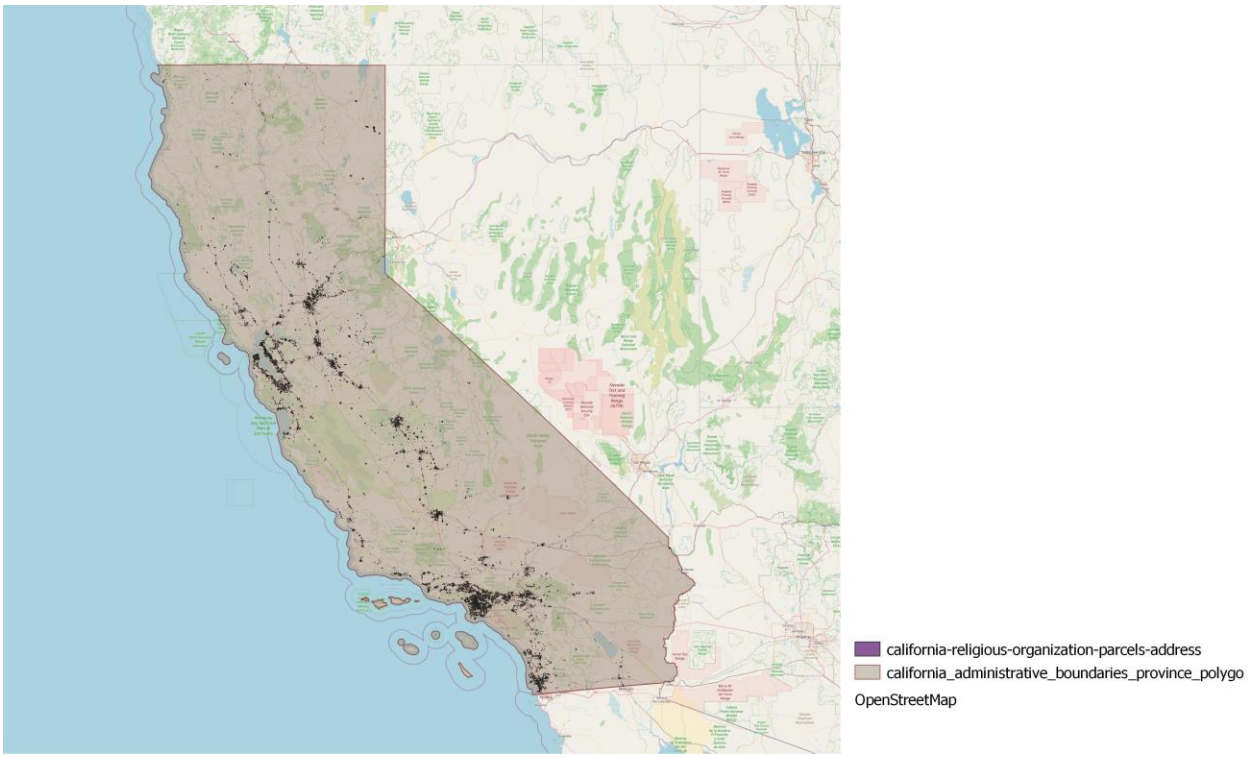


Figure 19: California's religious parcels address[65]

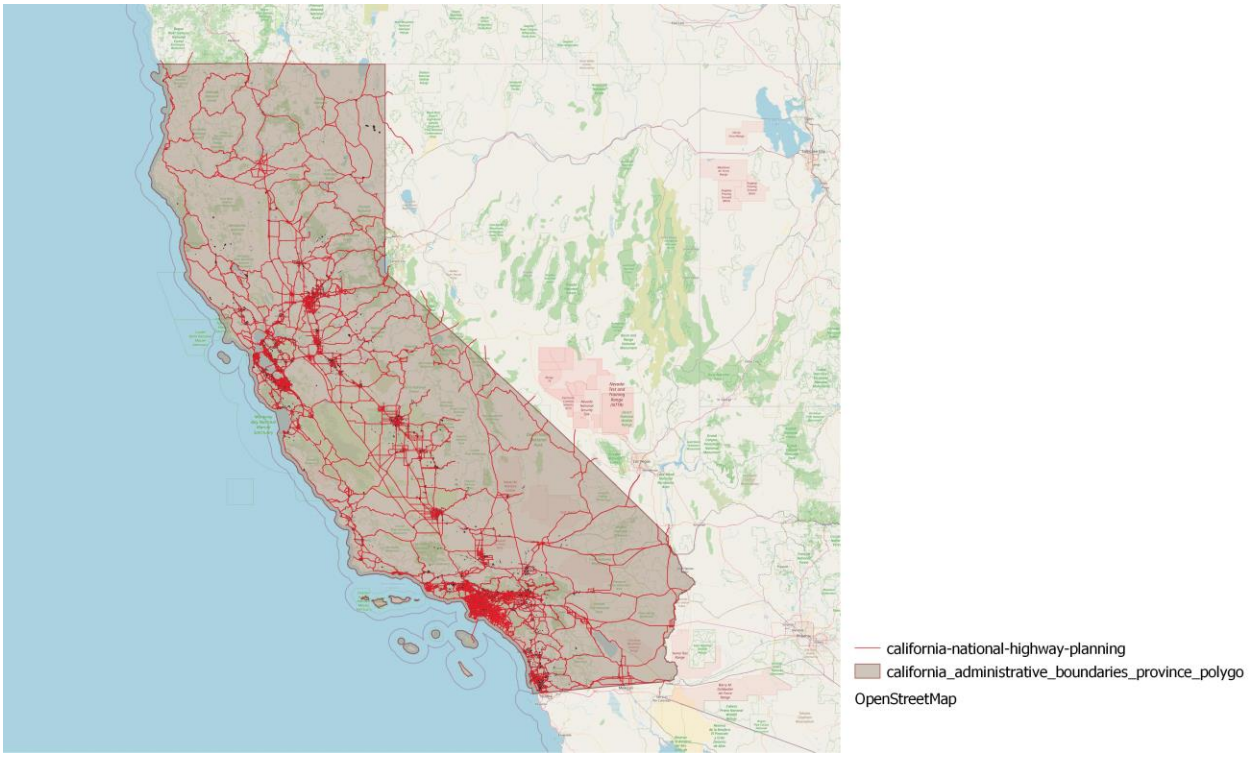


Figure 20: California's national highway planning [65]

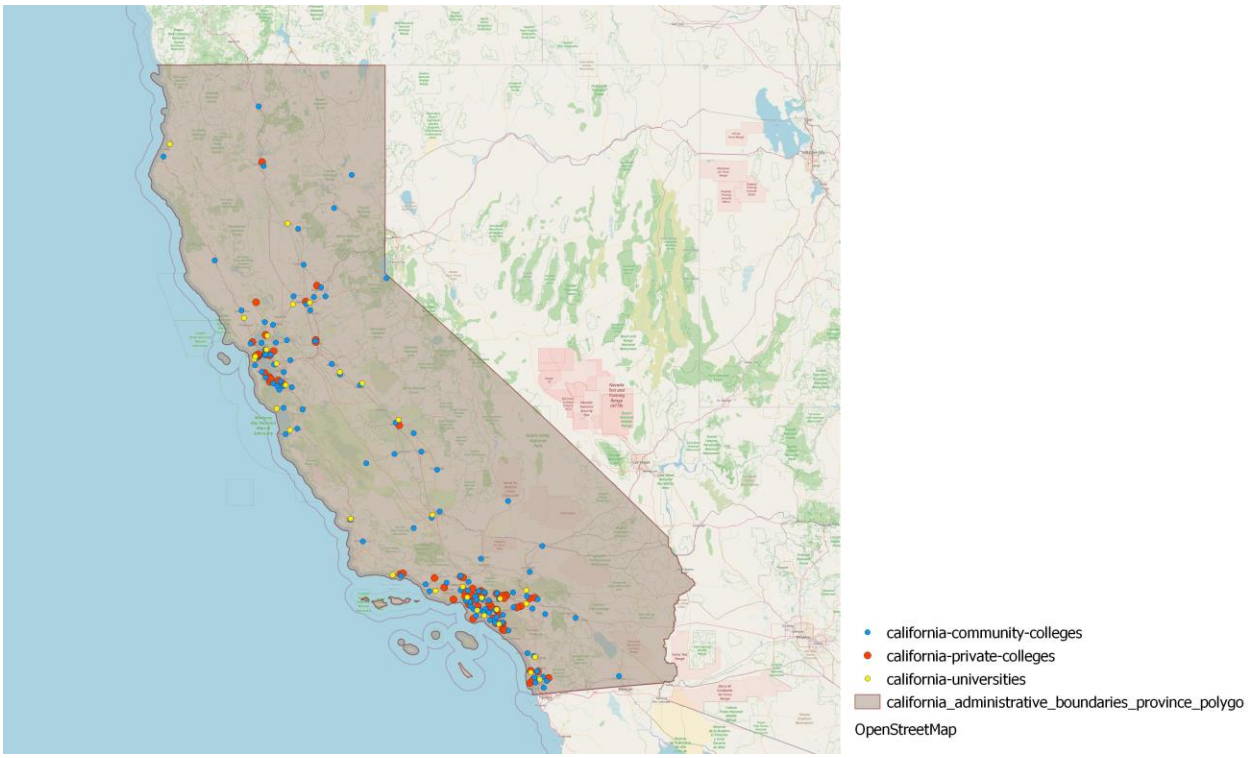


Figure 21: California's colleges & universities [65]

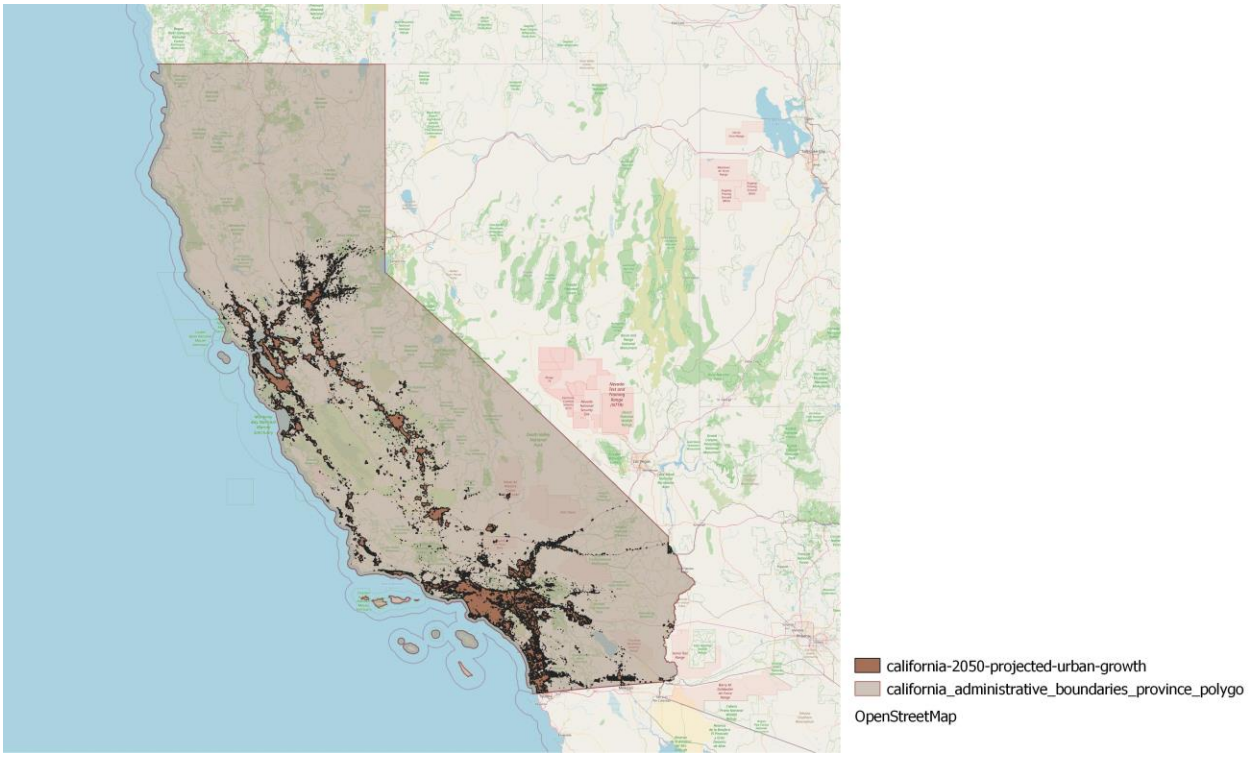


Figure 22: California's 2050 projected urban growth[65]

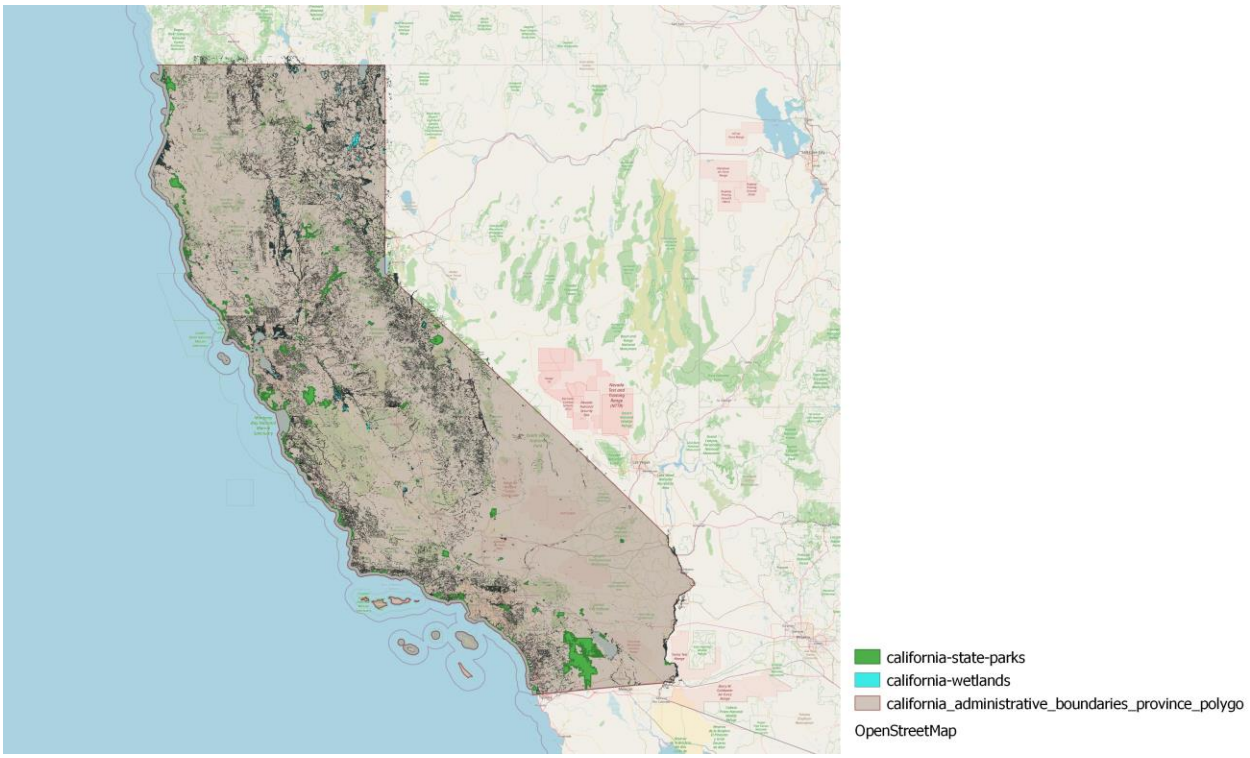


Figure 23: California's state parks & wetlands[65]

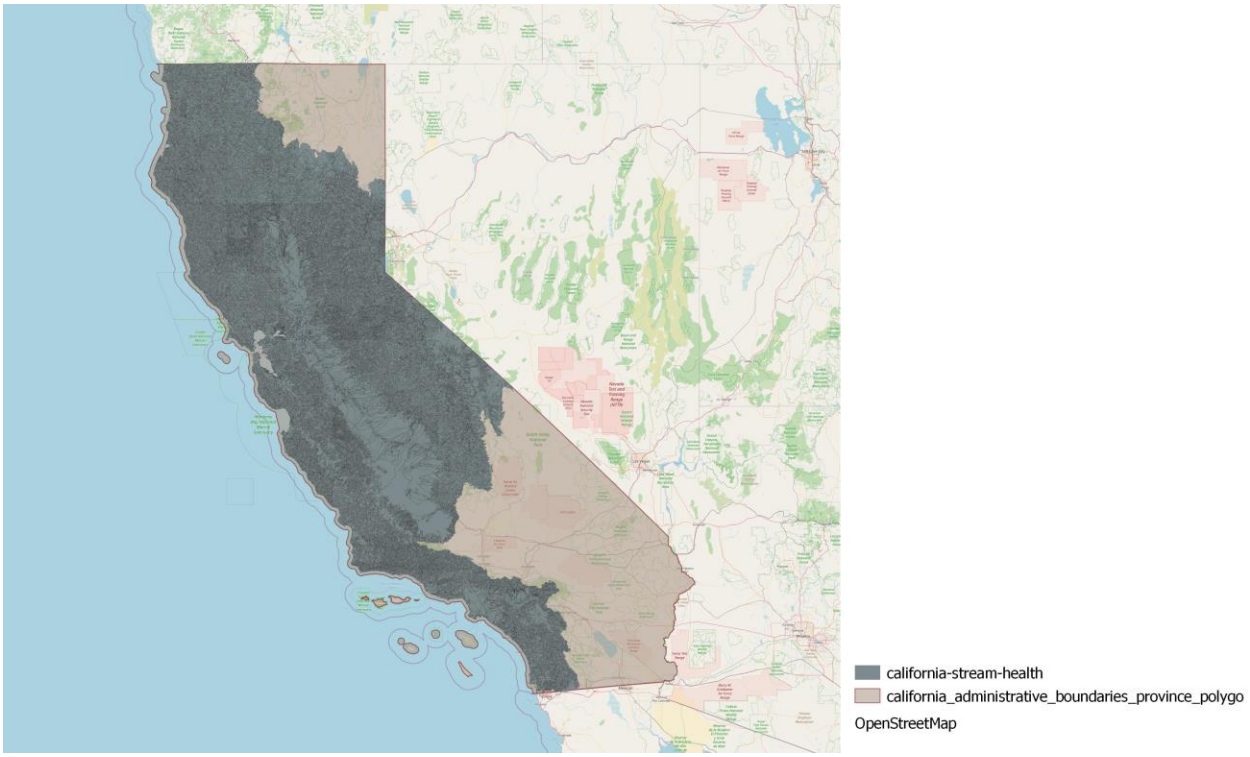


Figure 24: California's stream health [65]

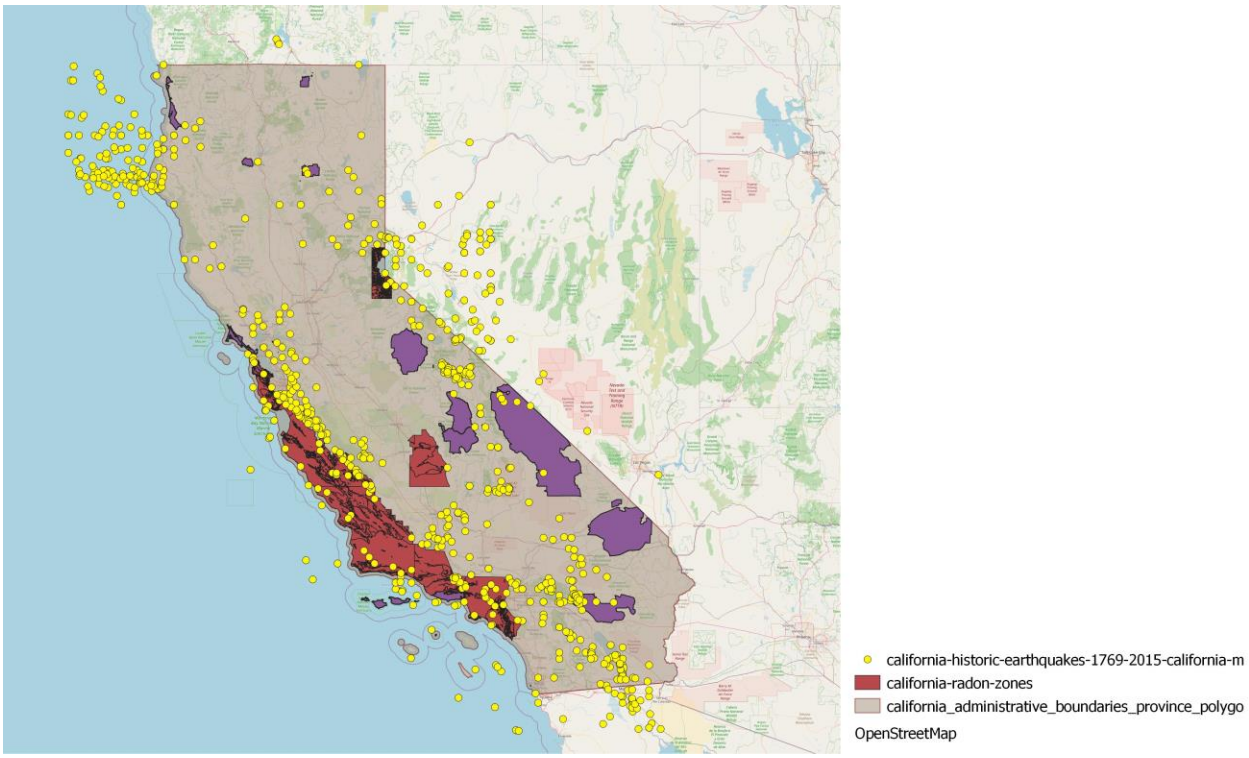


Figure 25: Historic earthquakes & radon zones[65]

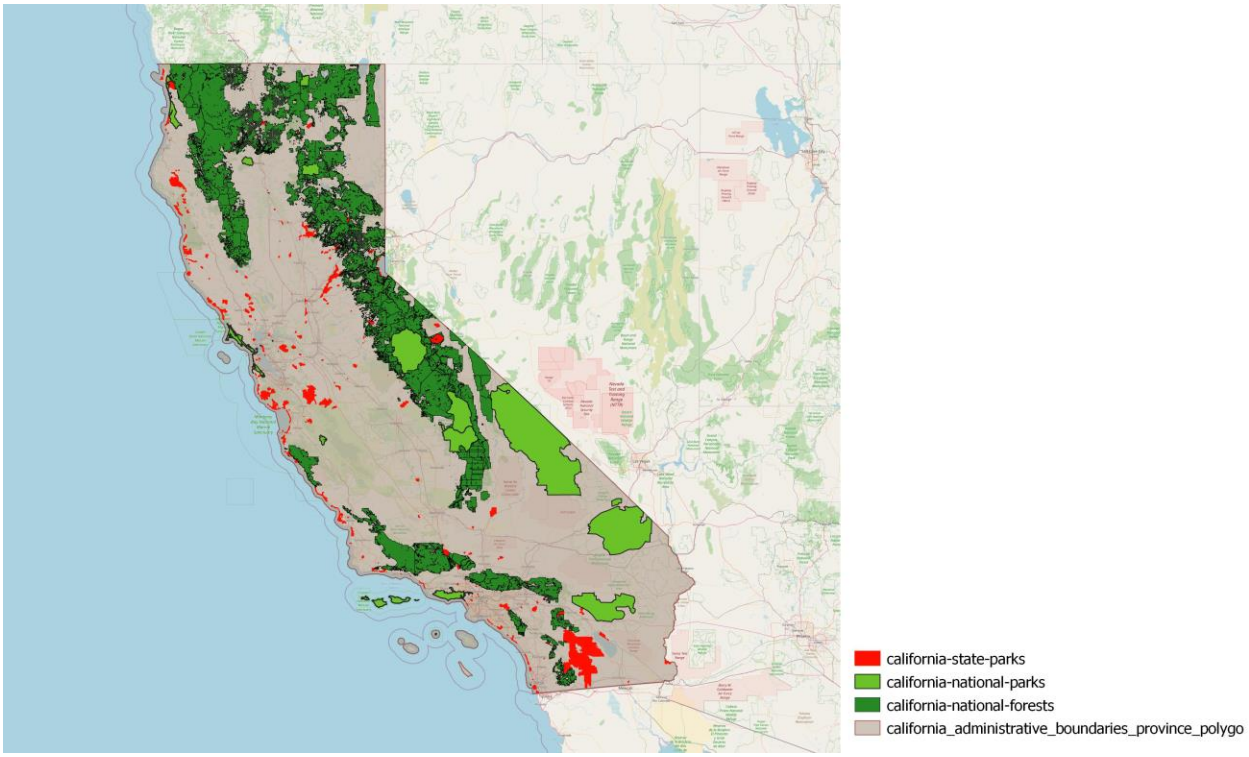


Figure 26: California's national parks, state parks & National forests [65]

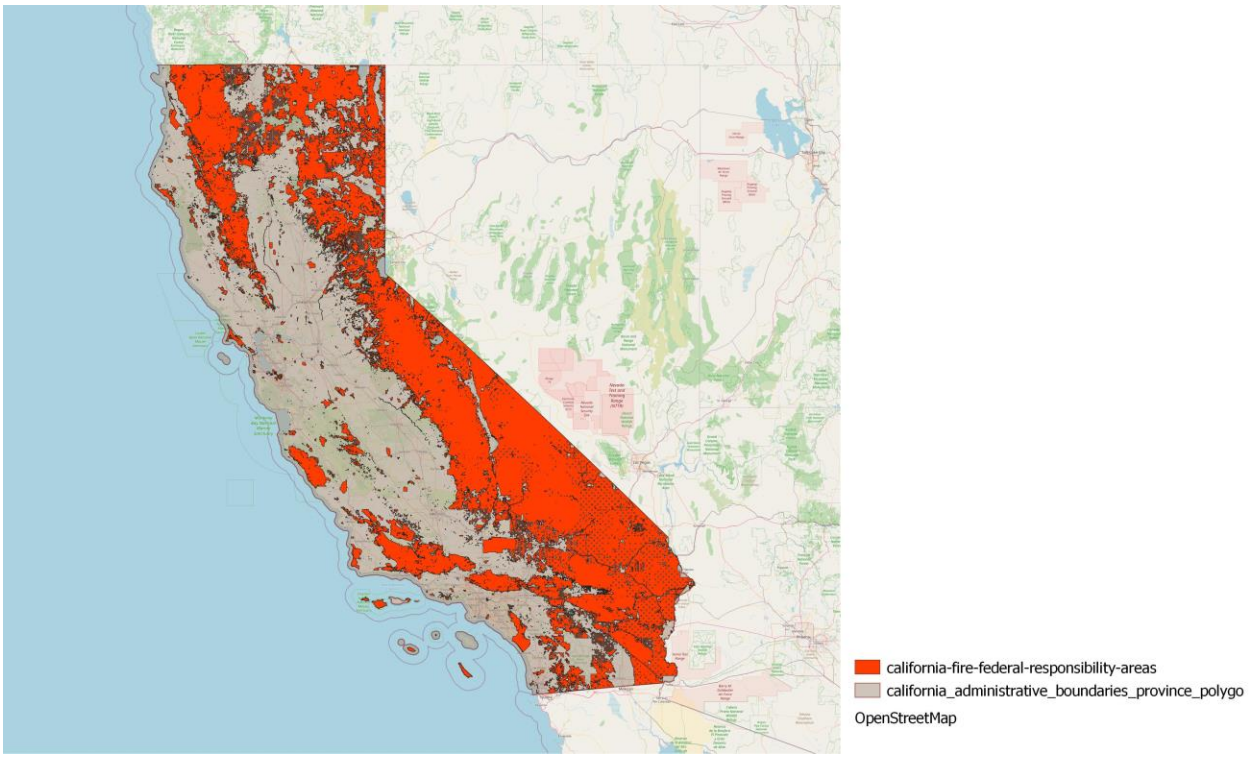


Figure 27: California's fire federal responsibility areas[65]

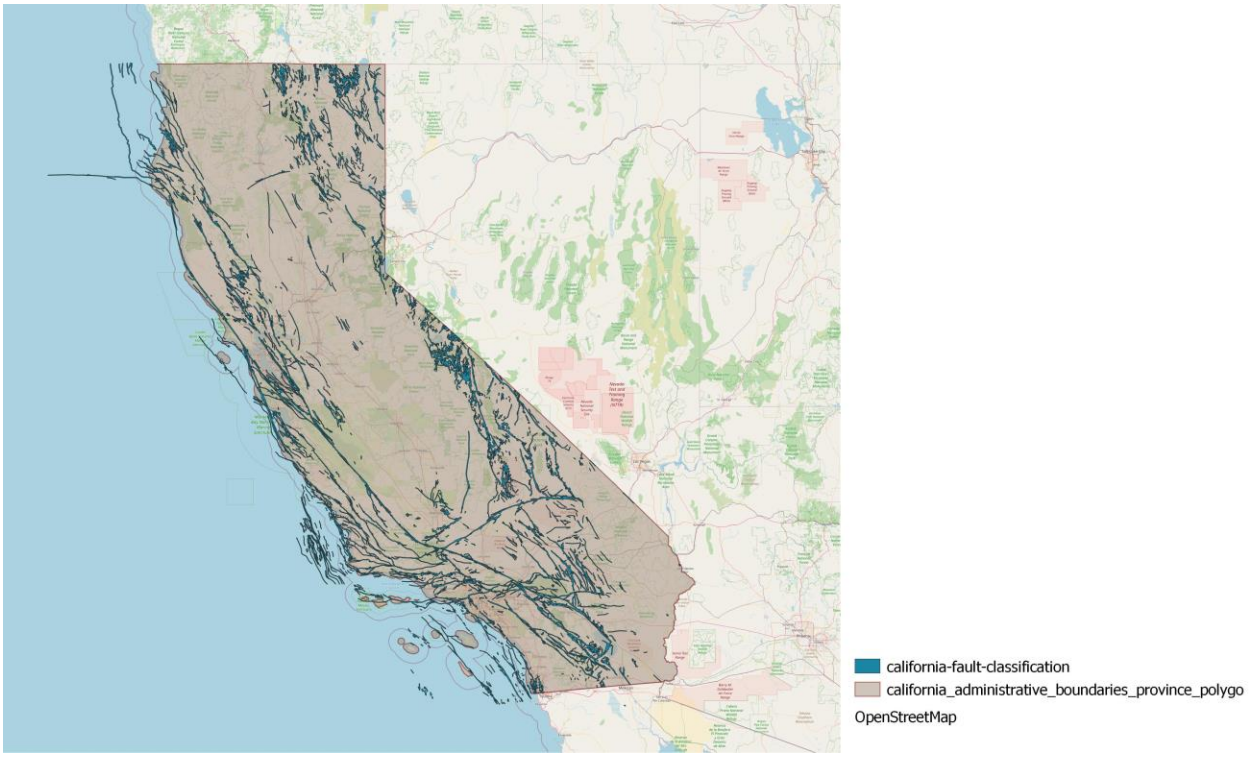


Figure 28: California's fault classification[65]

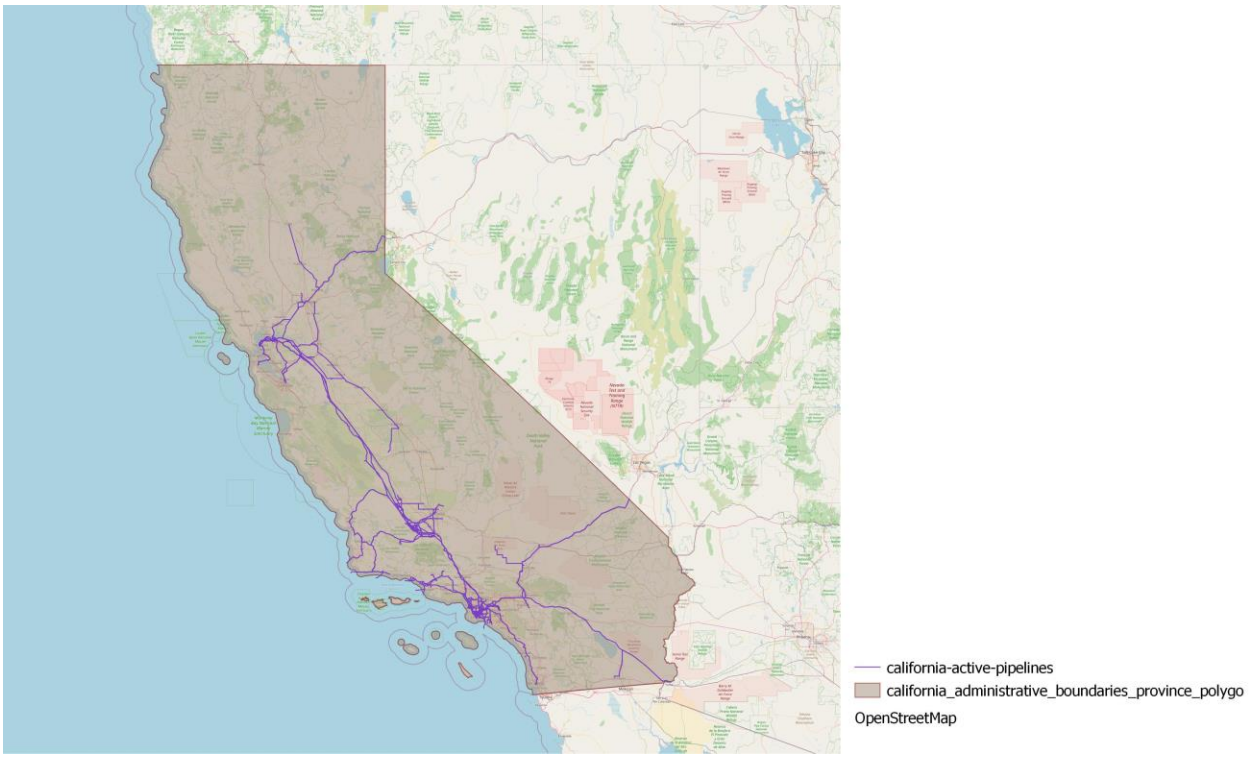


Figure 29: California's active pipelines[65]

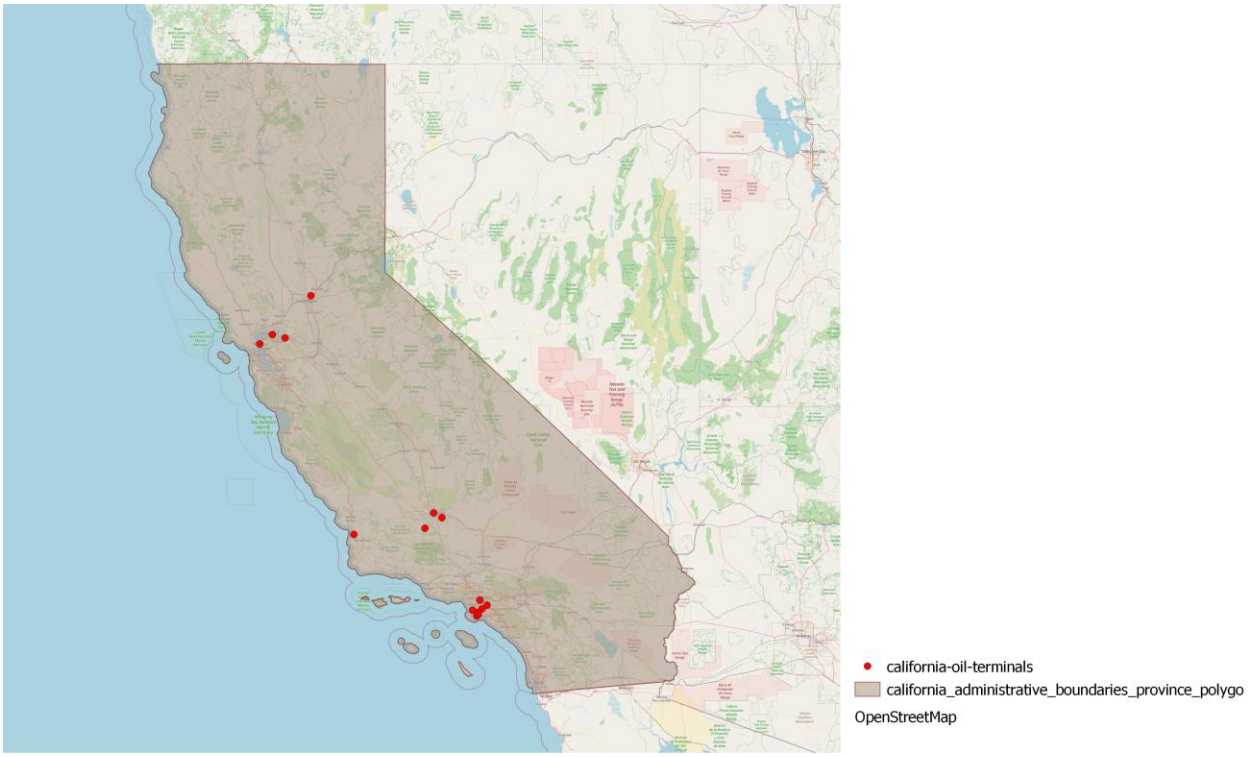


Figure 30: California's oil terminals[65]

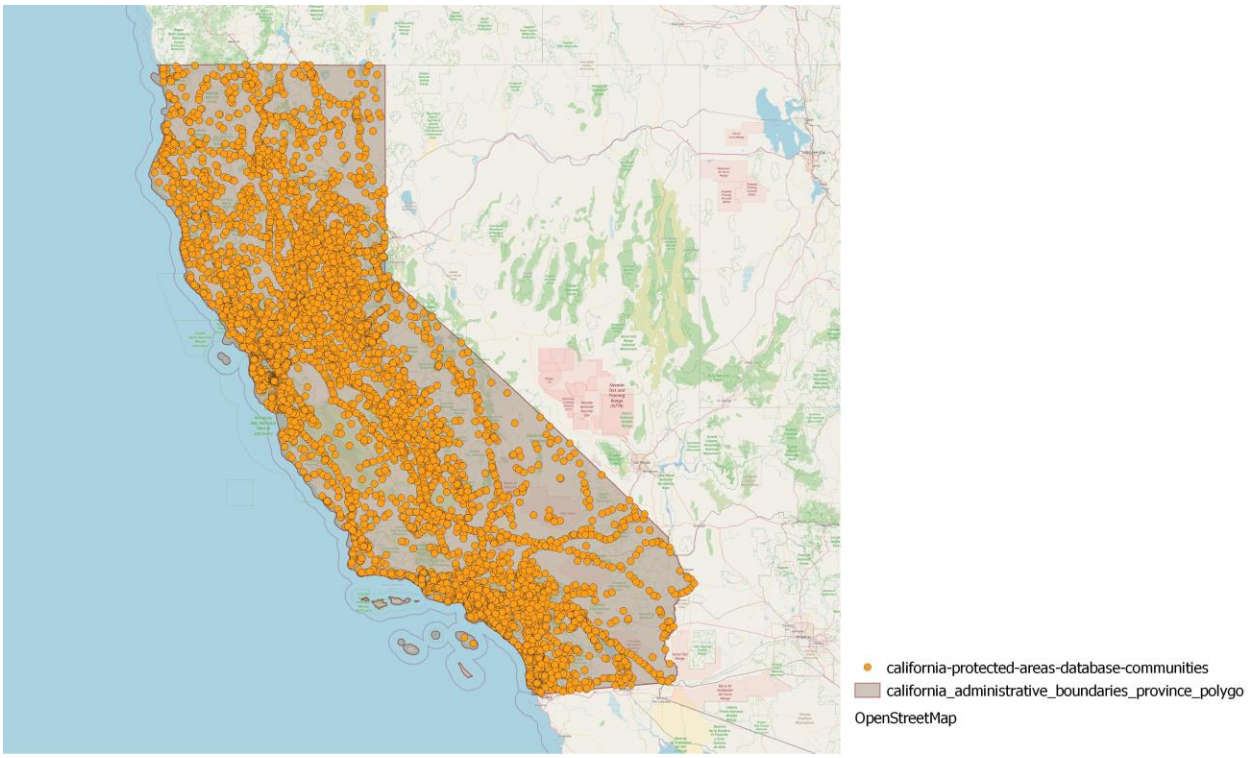


Figure 31: California's protected areas communities[65]

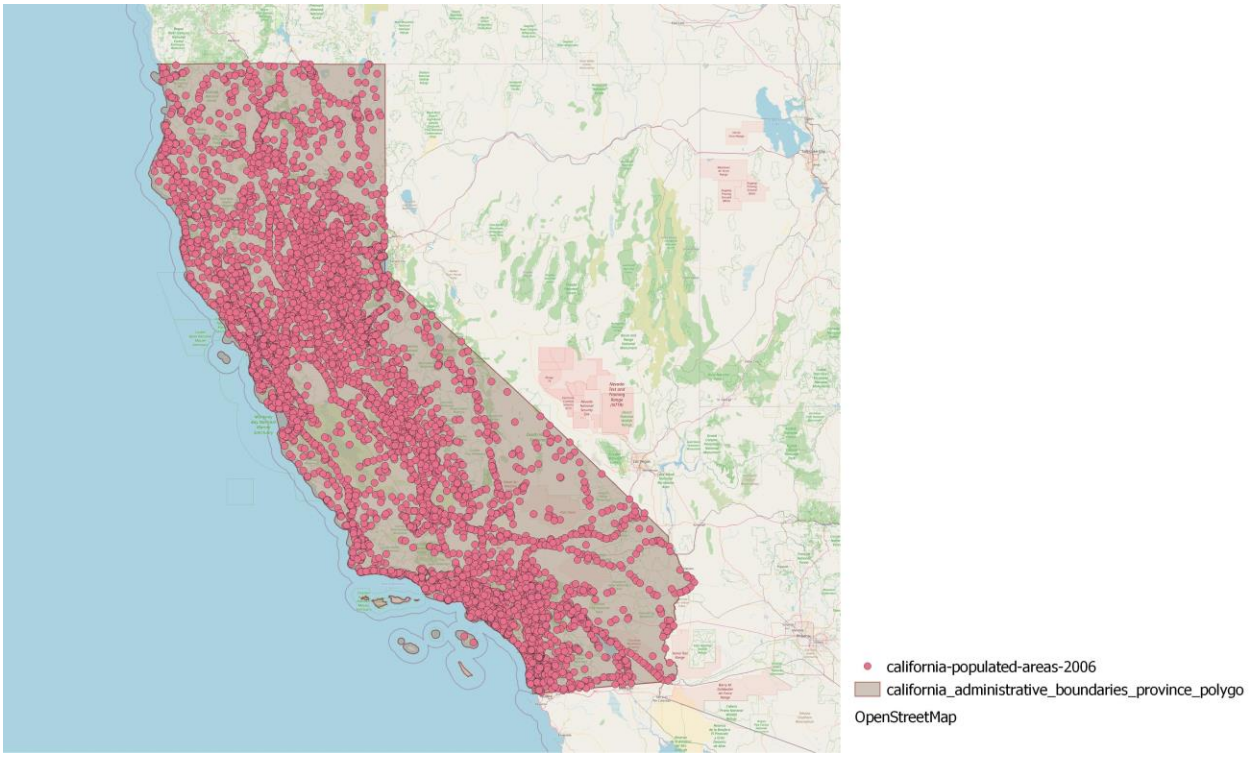


Figure 32: California's populated areas [65]

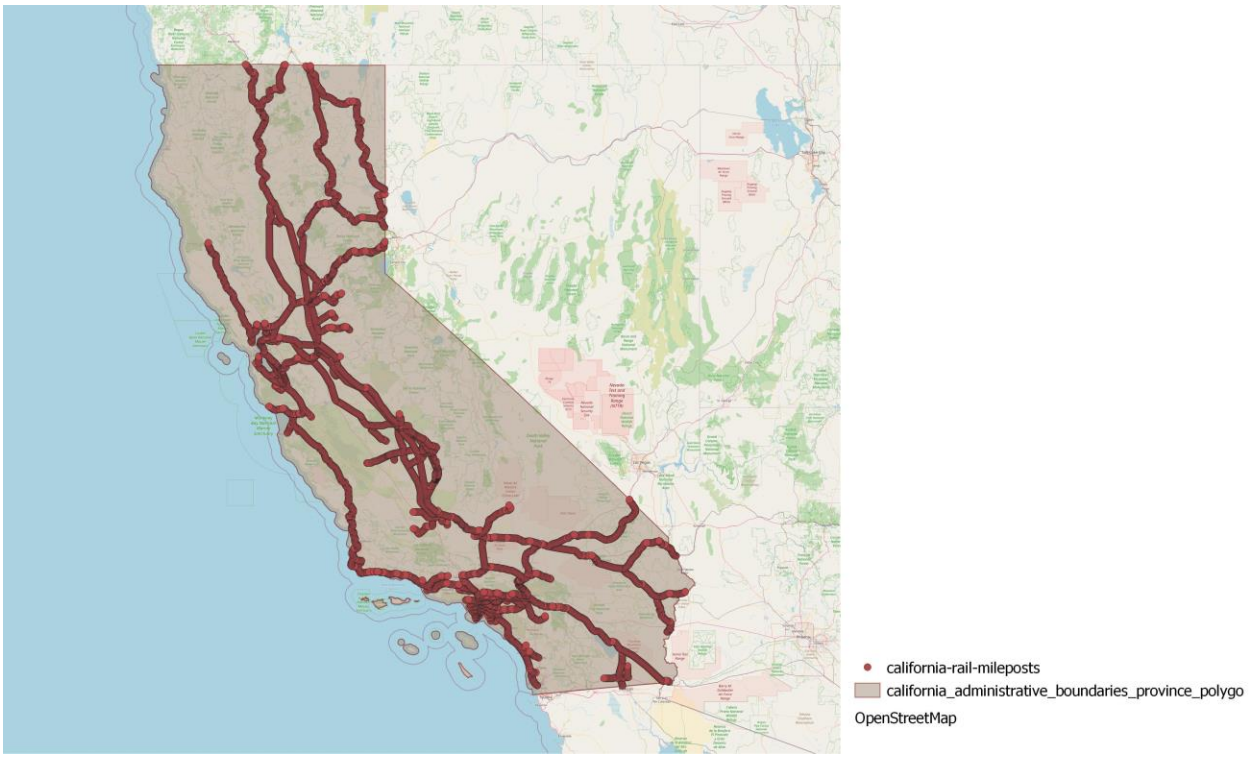


Figure 33: California's rail mileposts[65]

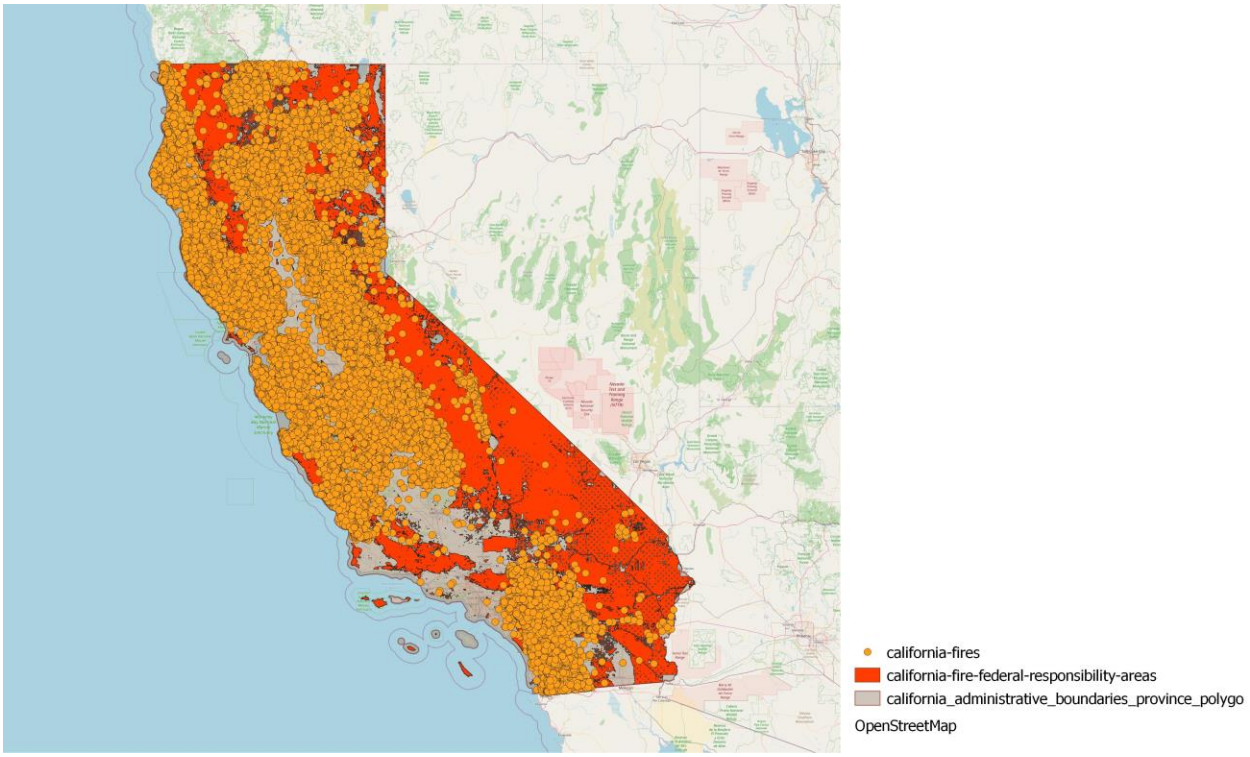


Figure 34: California's fires & fire federal responsibility areas[65]

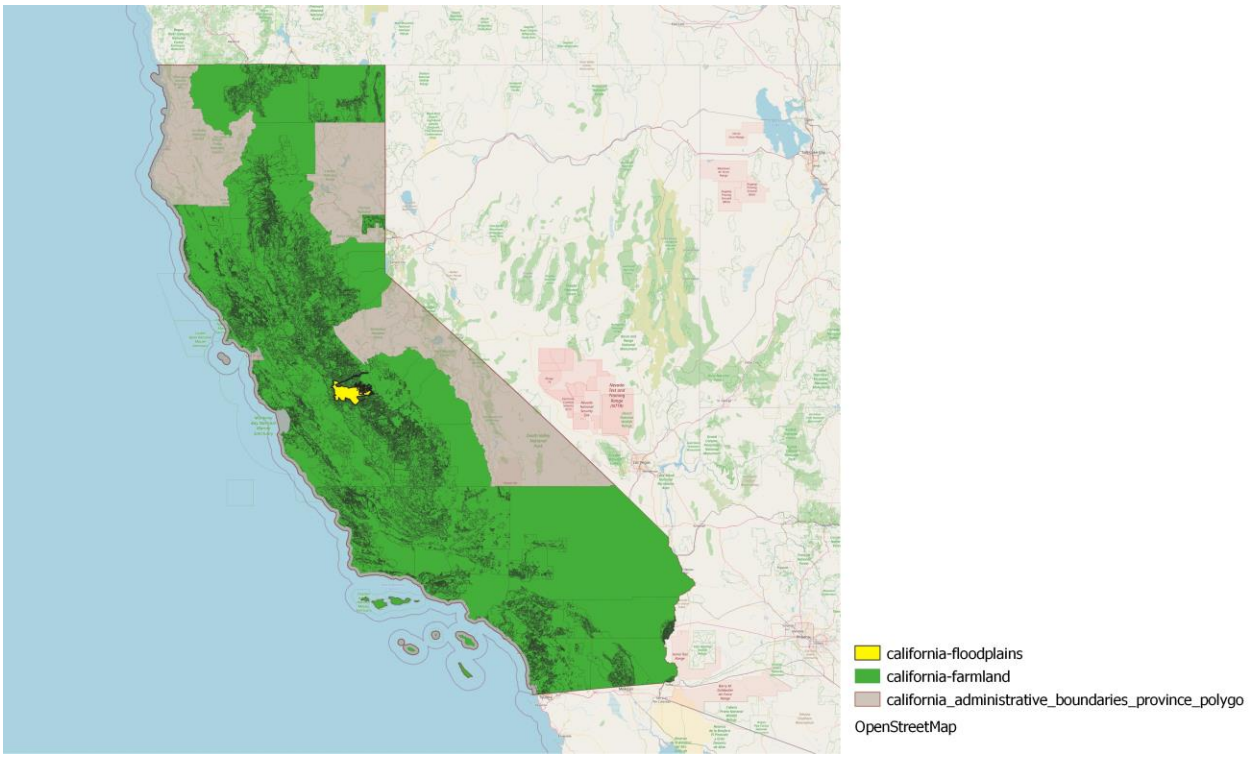


Figure 35: California's floodplains & farmlands [65]

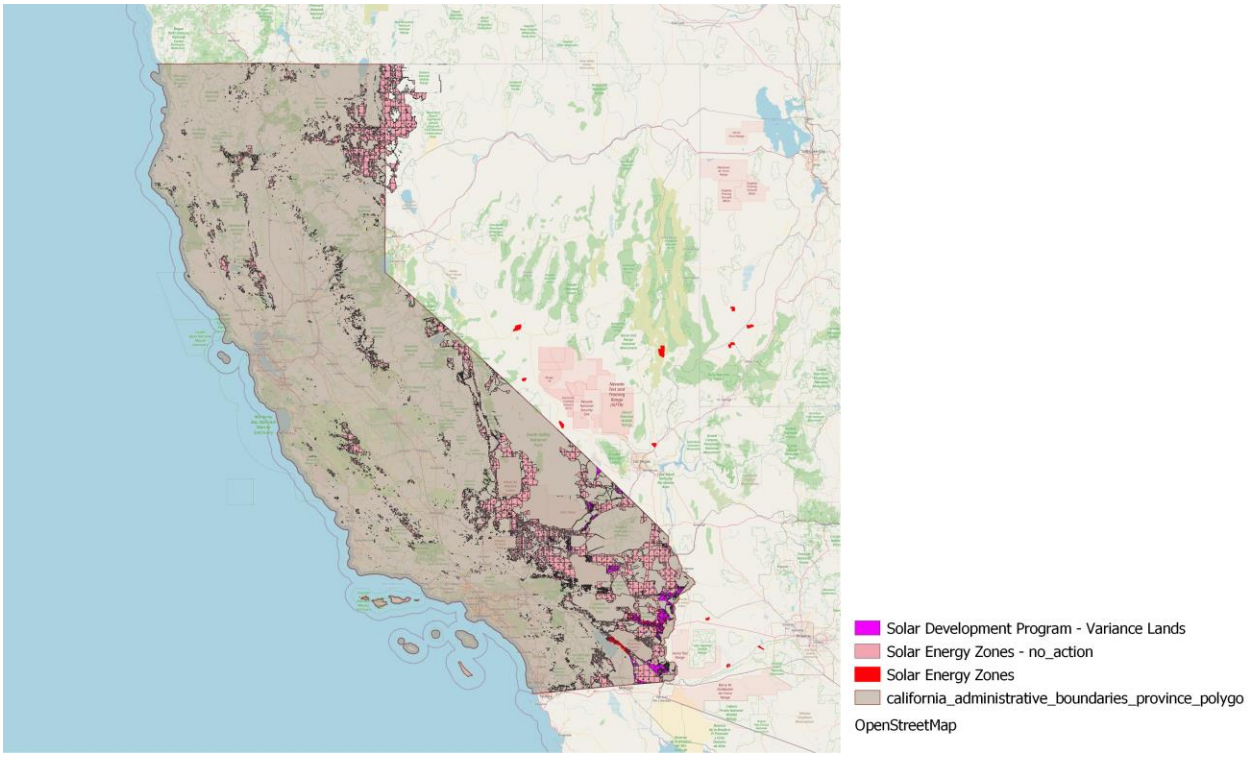


Figure 36: California's Solar development program, solar energy zones[65]

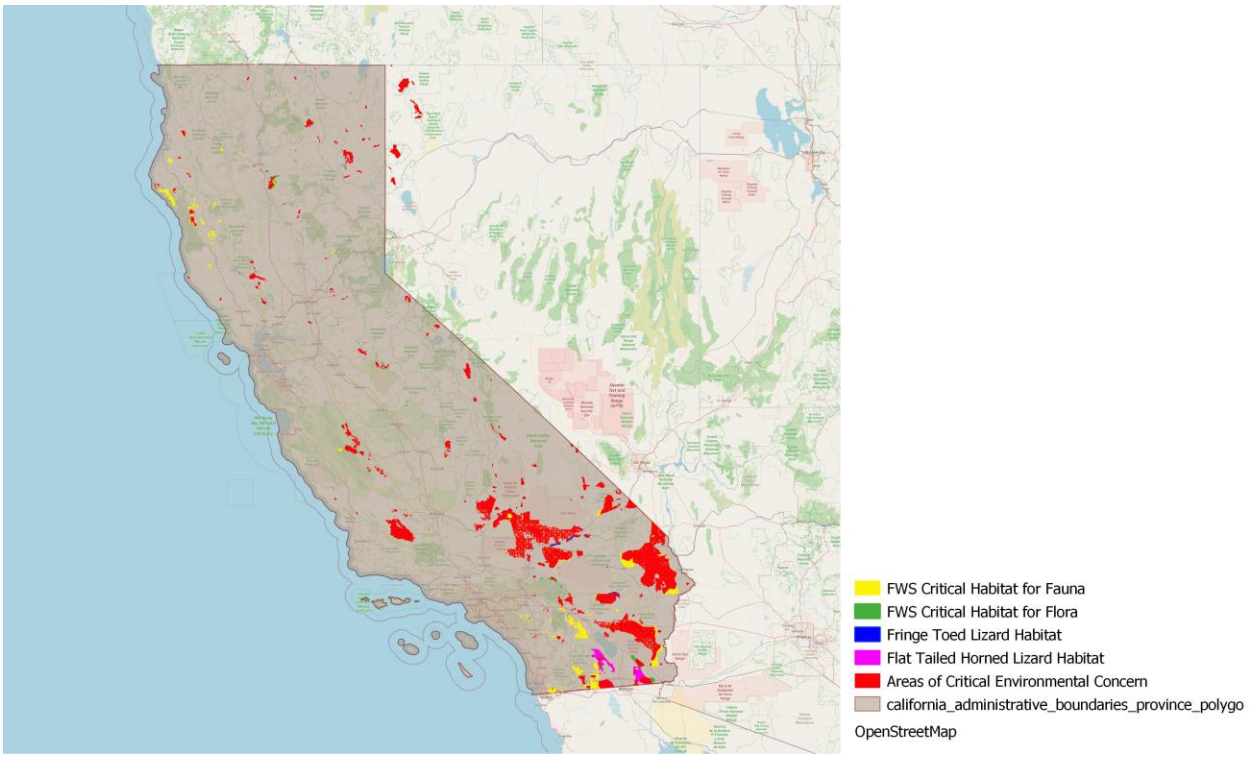


Figure 37: California's critical habitat for flora & fauna, critical environmental concern[65]

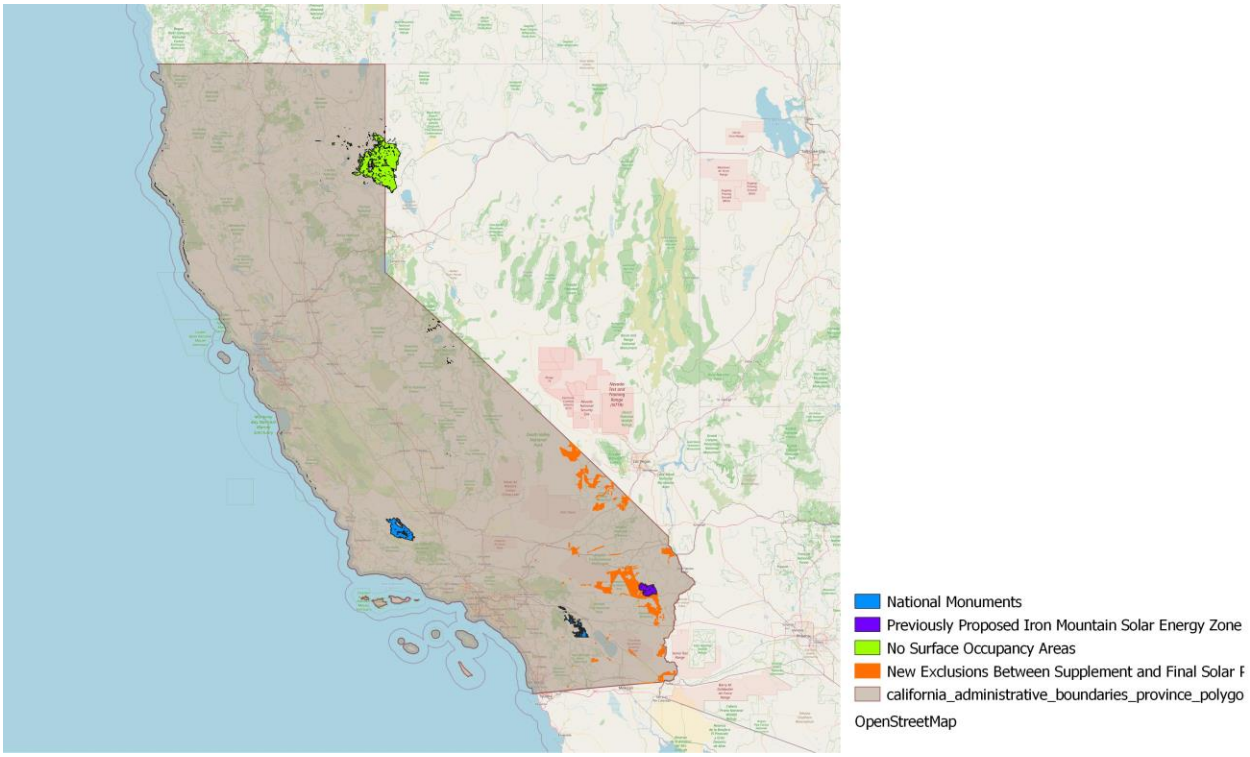


Figure 38: California's Iron mountain solar energy zone & no surface occupancy are[65]

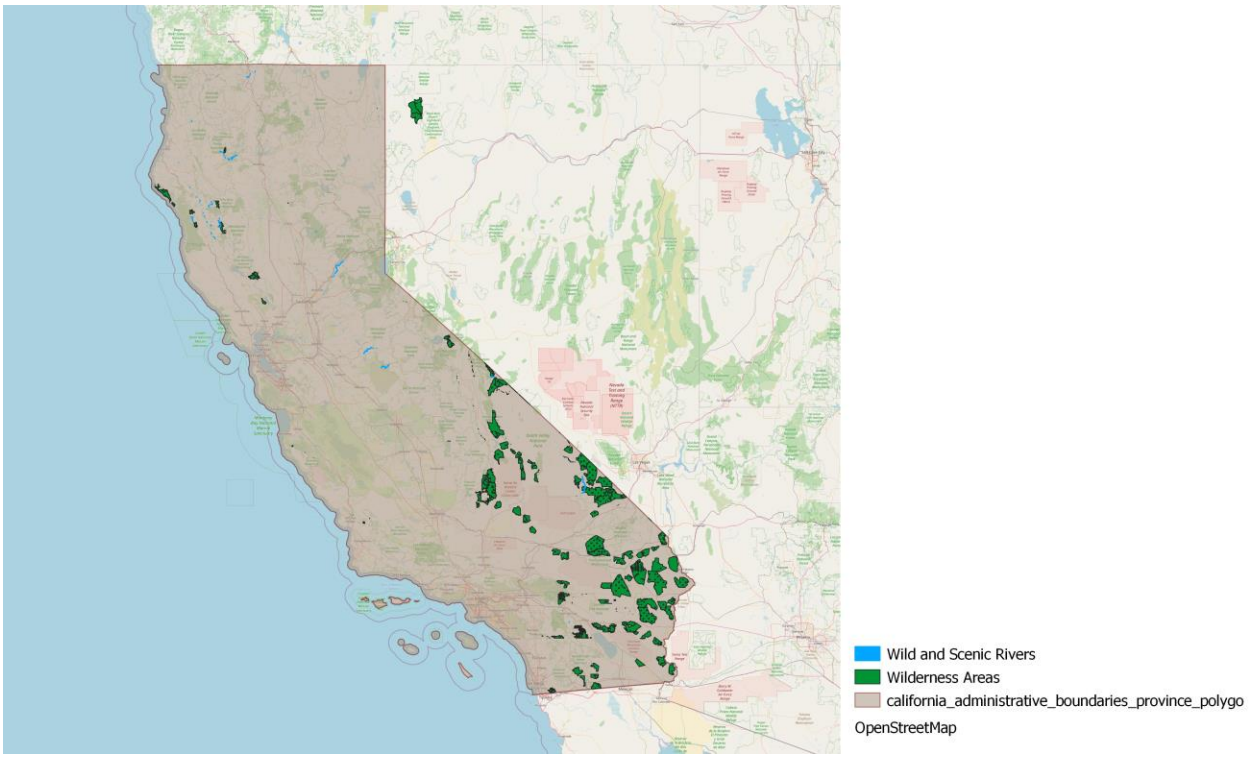


Figure 39: California's wild and scenic rivers & wilderness areas[65]

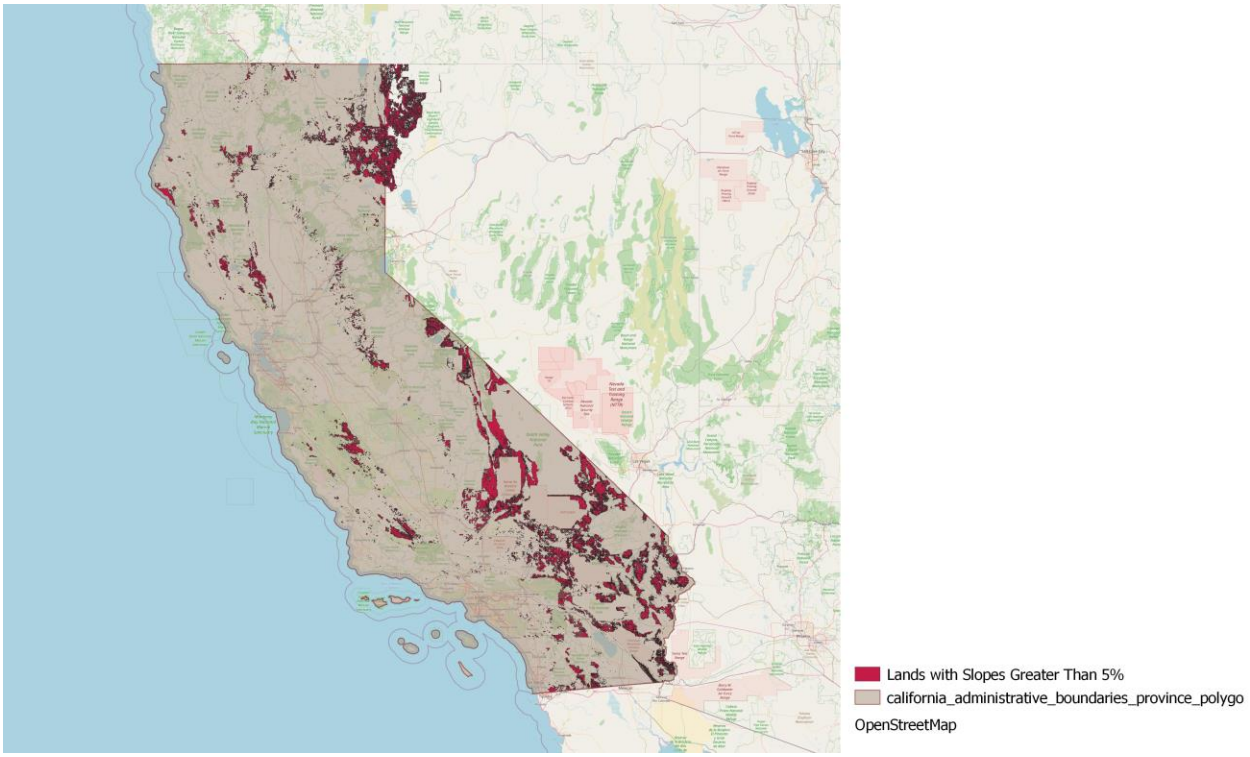


Figure 40: California's lands with slopes greater than 5%[65]

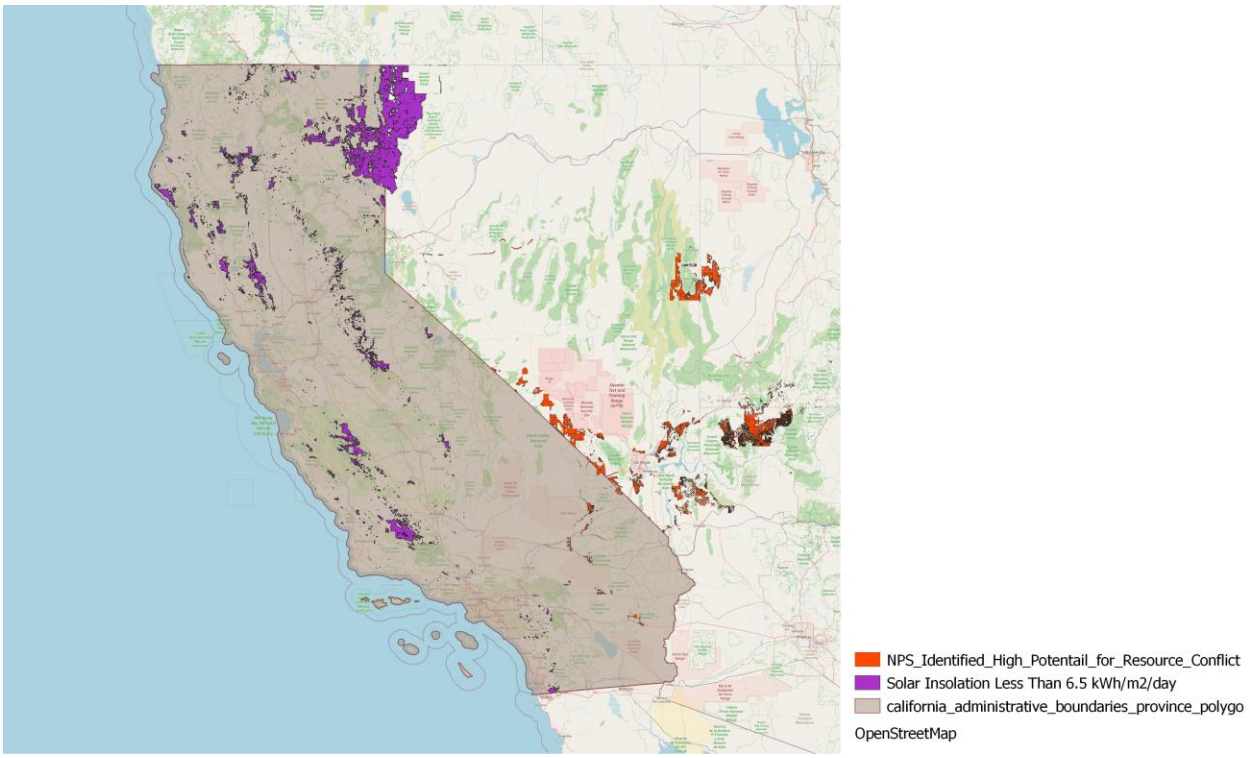


Figure 41: California's NPS identified high potential for resources conflict & solar insolation less than 6.5 KWh/m2/day[65]

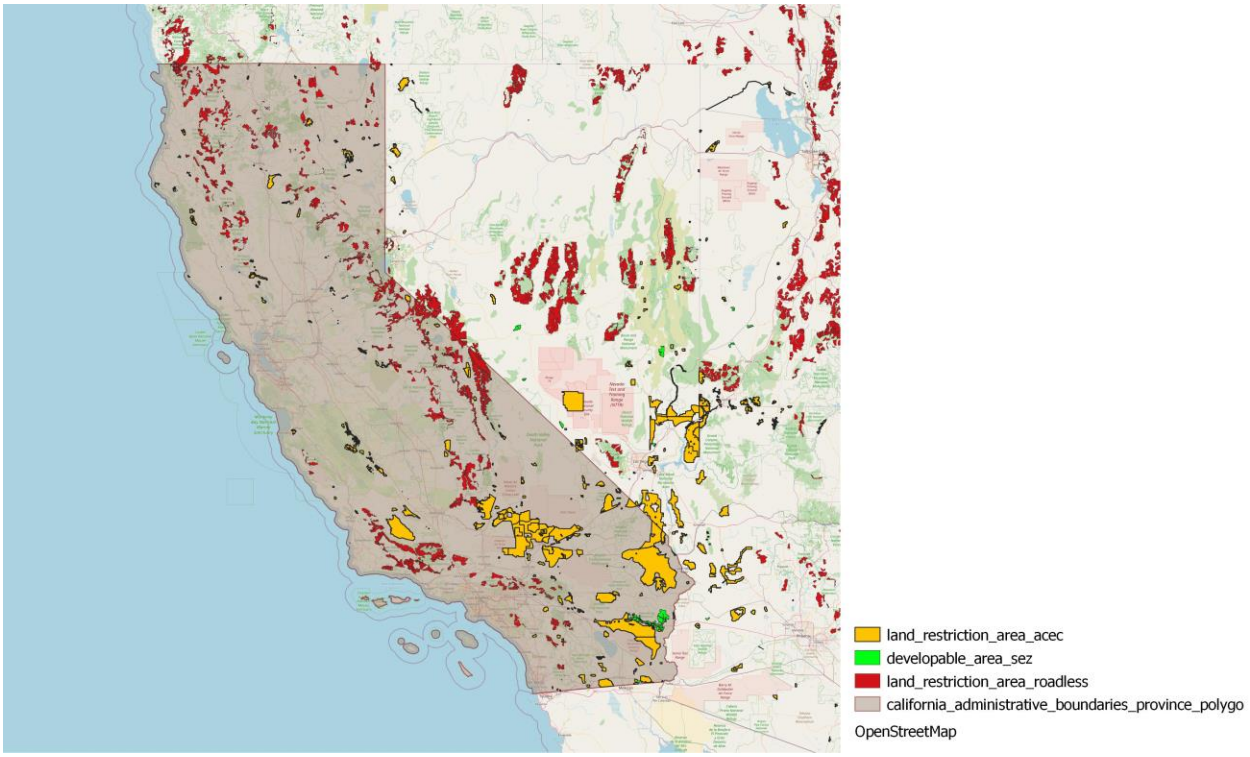


Figure 42: California's land restriction area & developed are & land restriction area roadless[65]

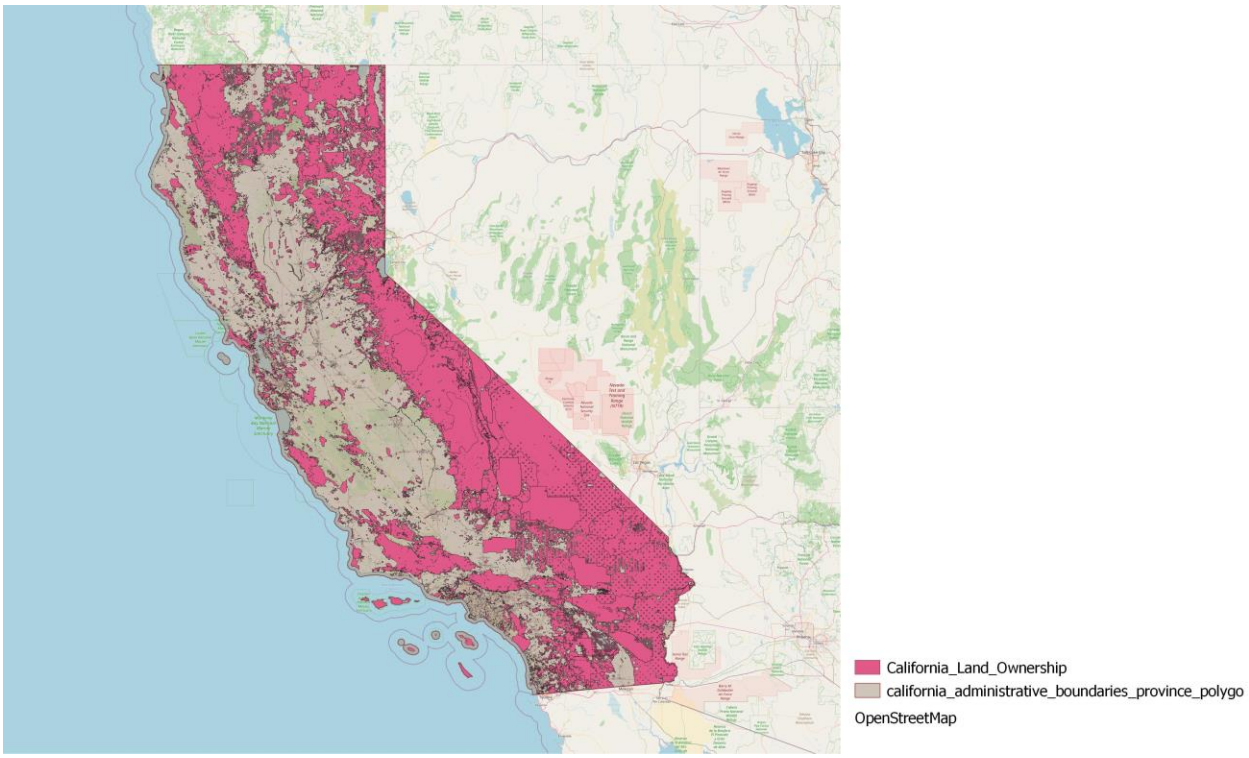


Figure 43: California's land ownership[65]

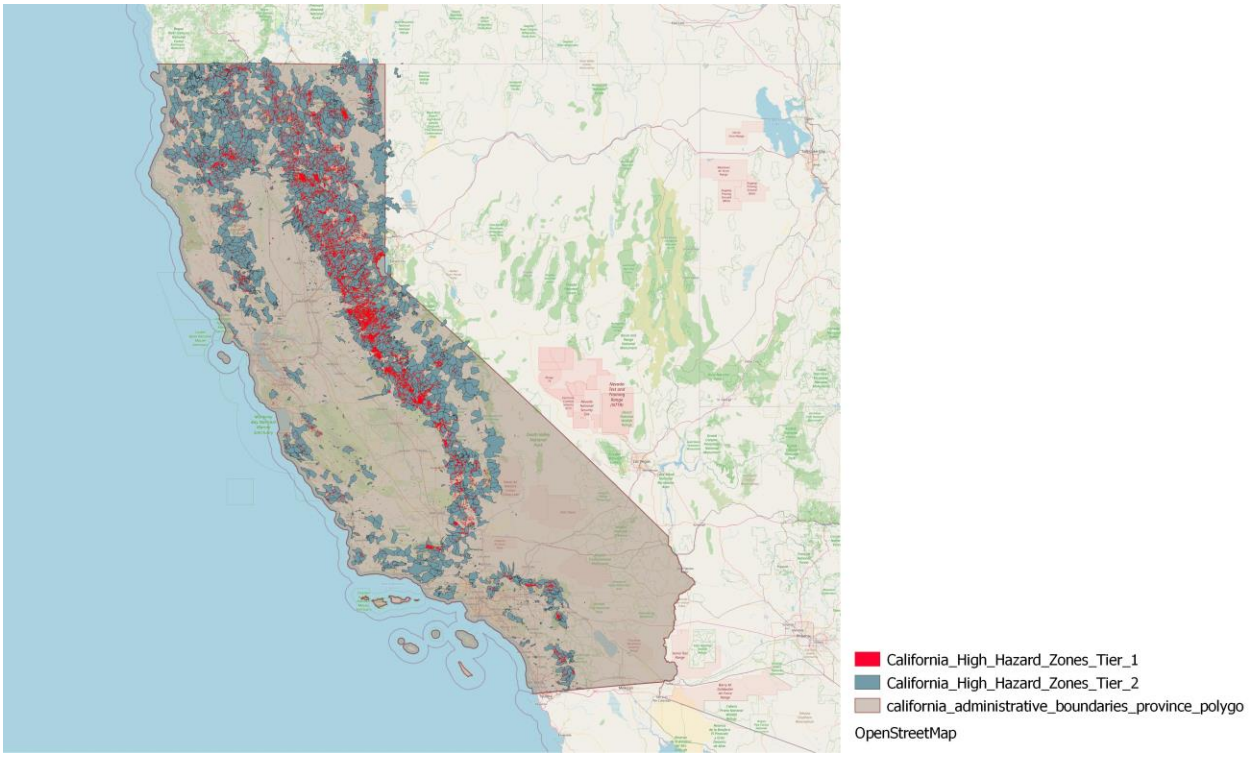


Figure 44: California's high hazard zone[65]

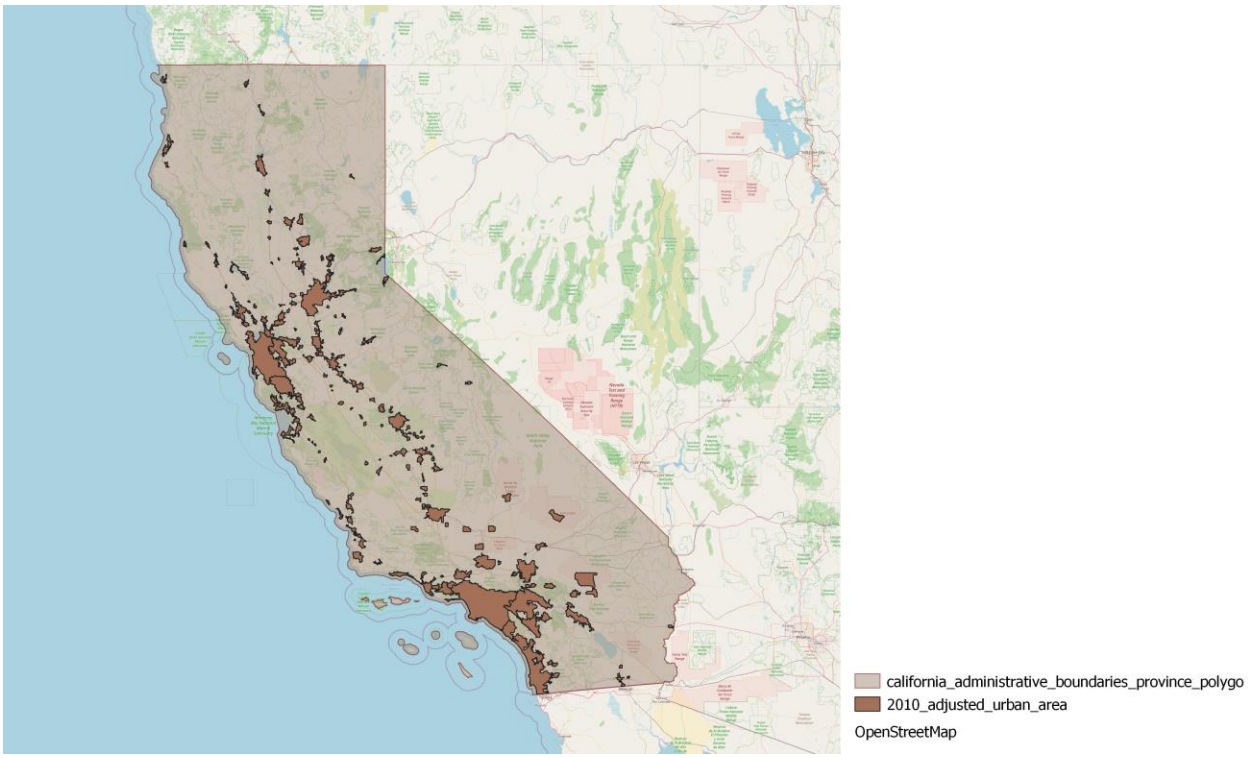


Figure 45: California's adjusted urban area[65]

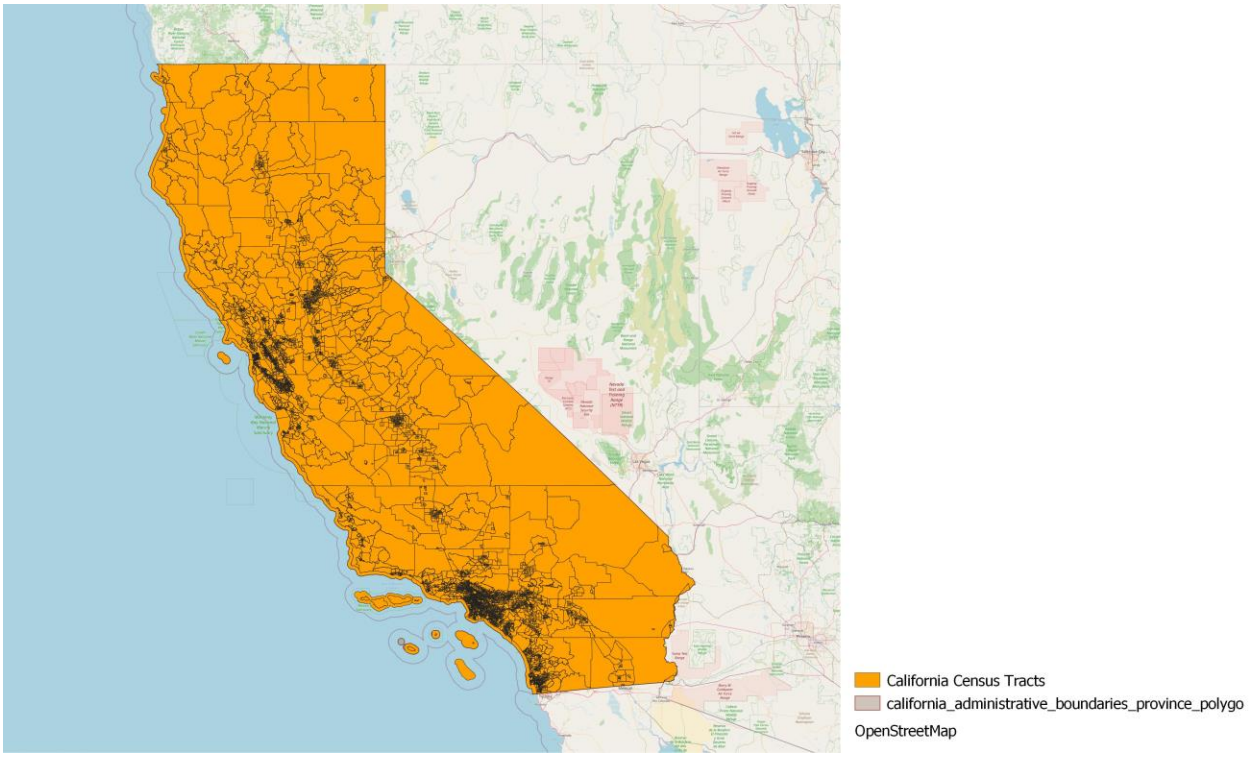


Figure 46: California's census tracts[65]

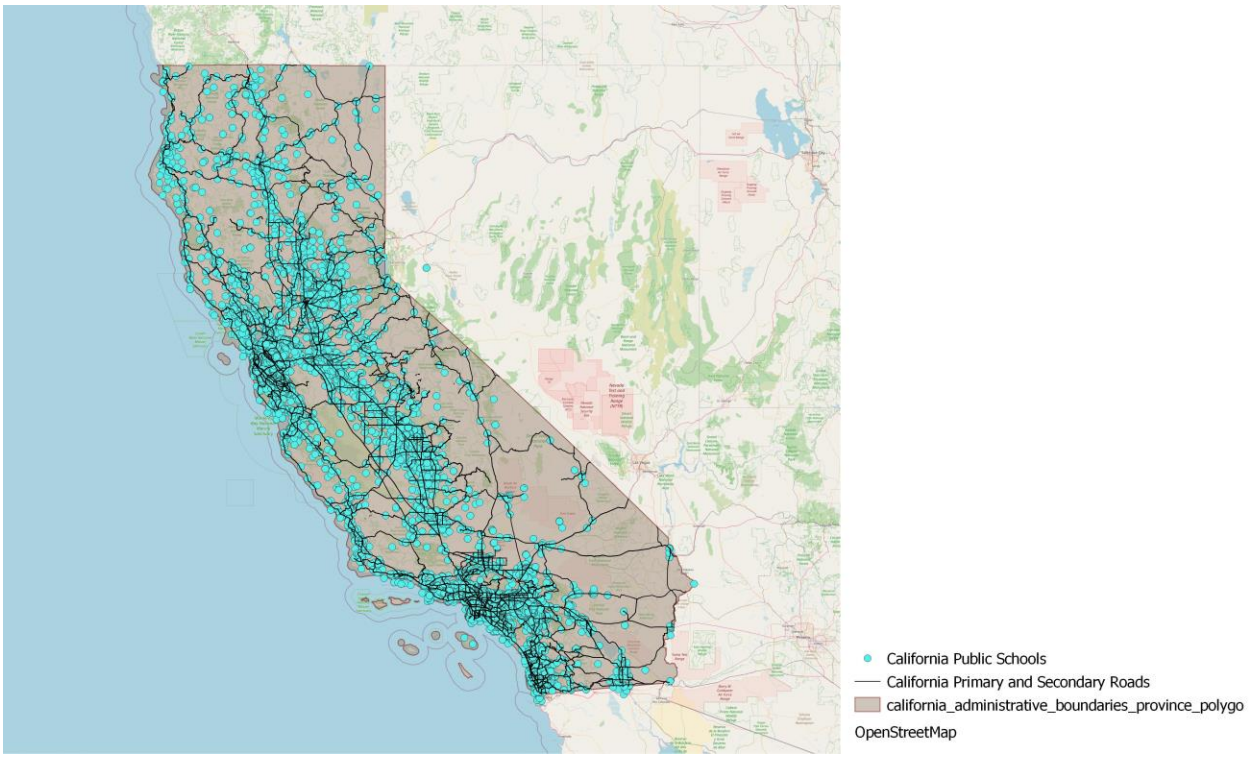


Figure 47: California's public schools & primary and secondary roads[65]

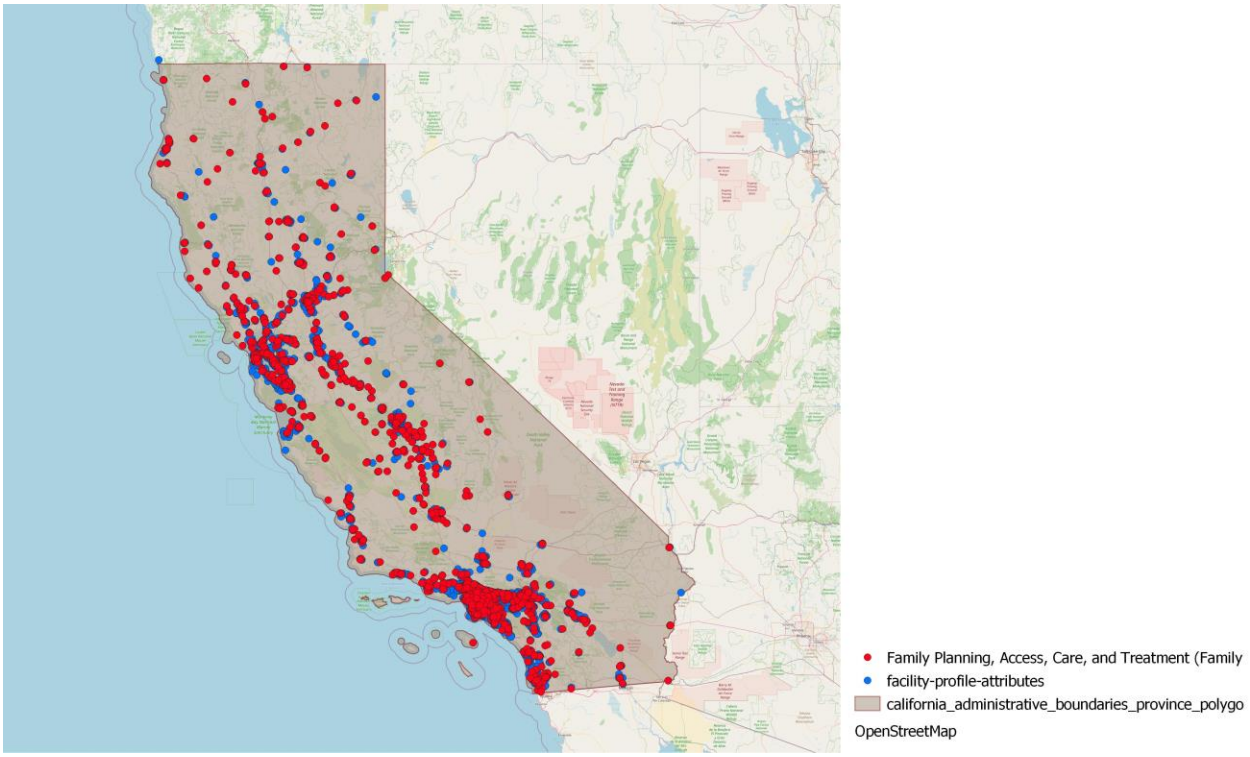


Figure 48: California's family planning, access, care and treatment & facility profile attributes[65]

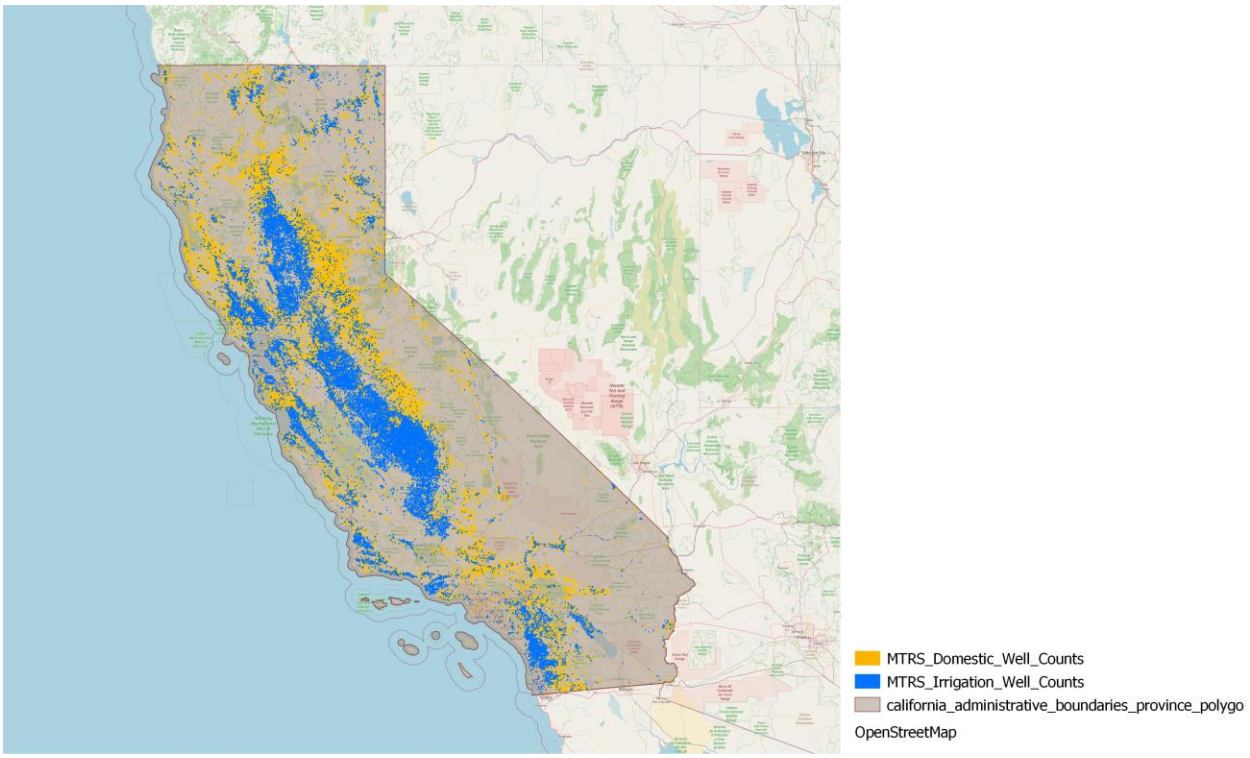


Figure 49: California's domestic and irrigation wells[65]

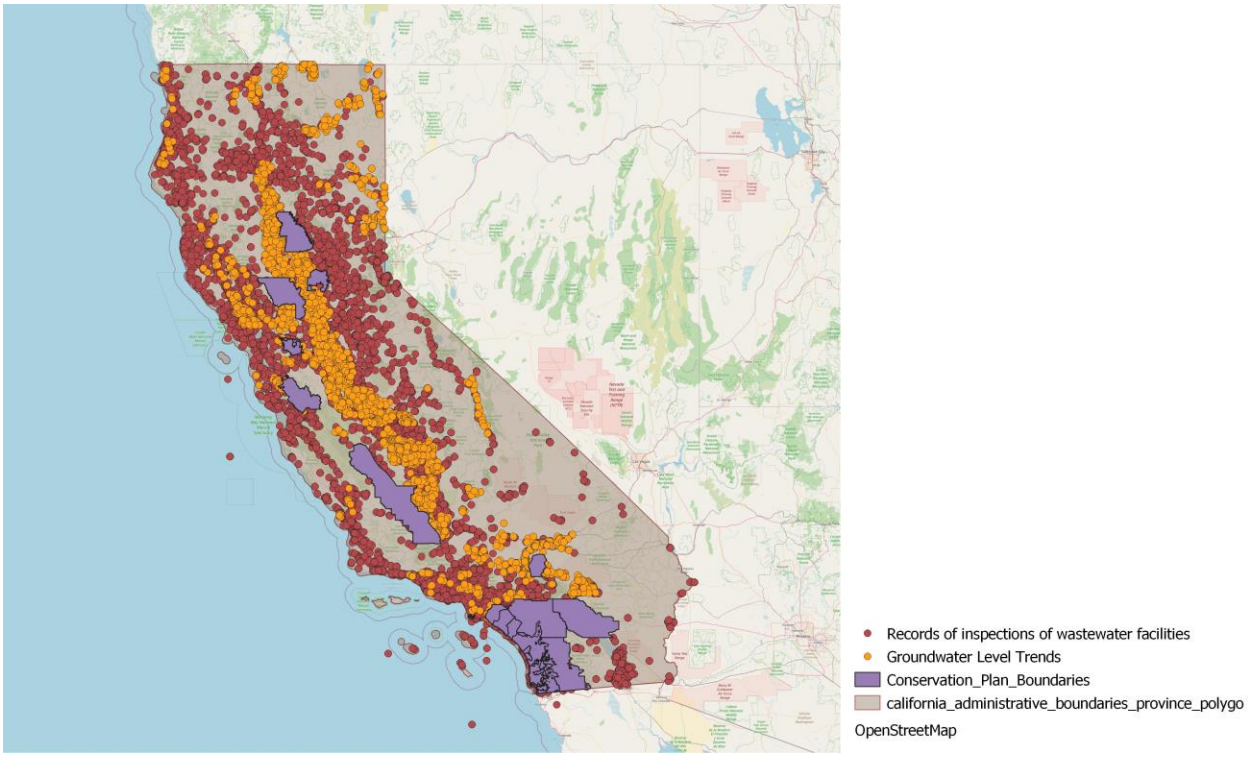


Figure 50: California's inspections of wastewater facilities & groundwater level trends & conservation plan boundaries[65]

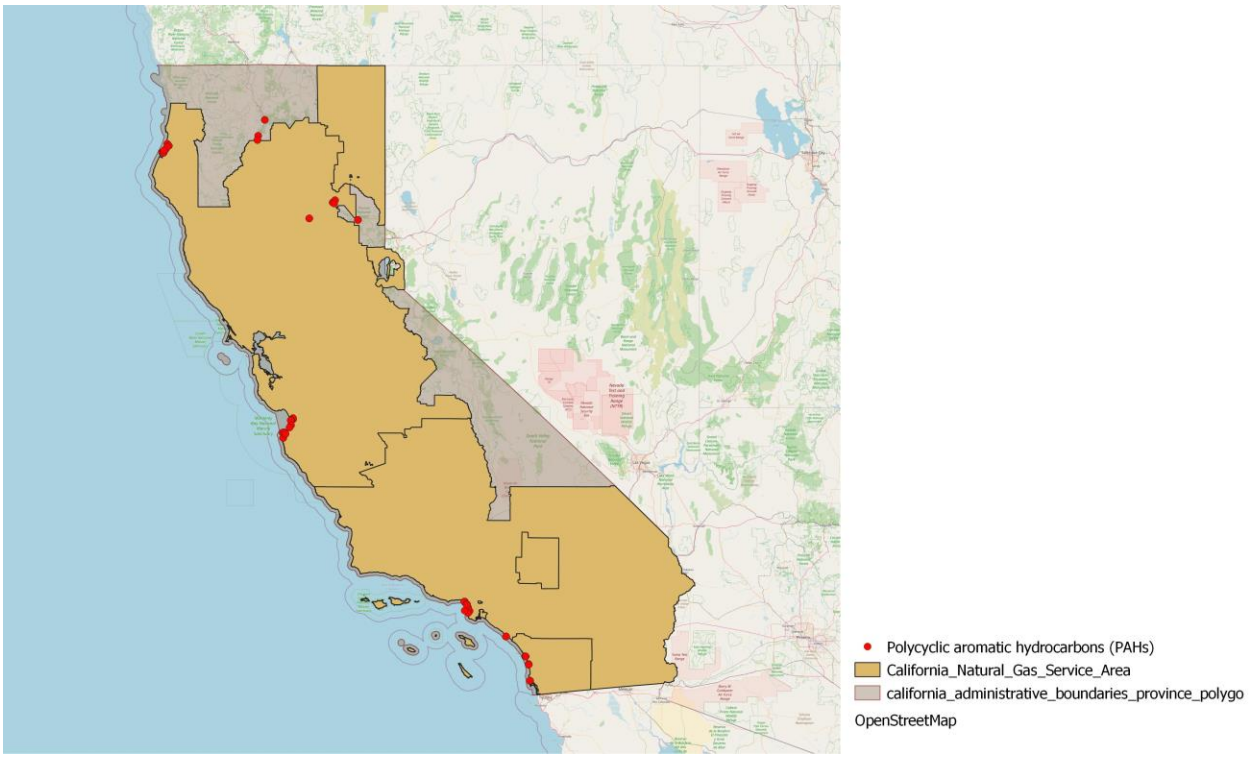


Figure 51: California's natural gas service area[65]

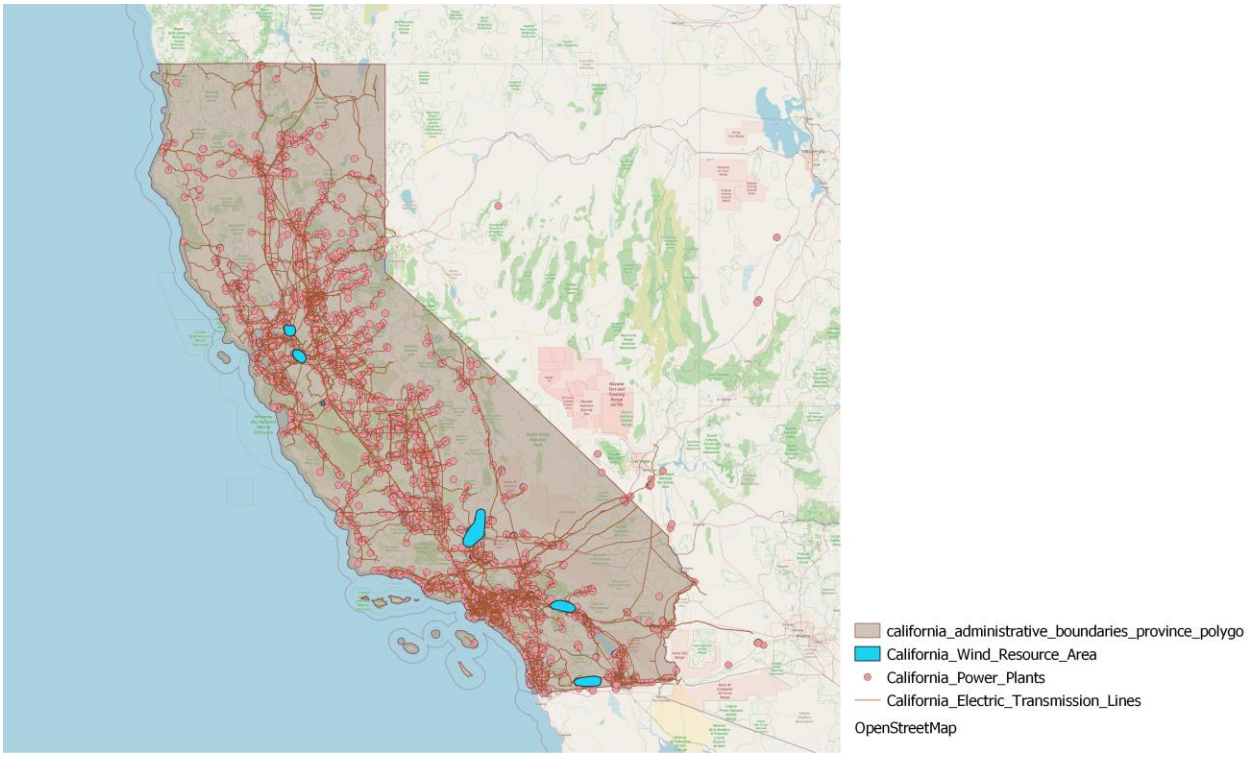


Figure 52: California's wind resource area & power plants & electric transmission lines[65]

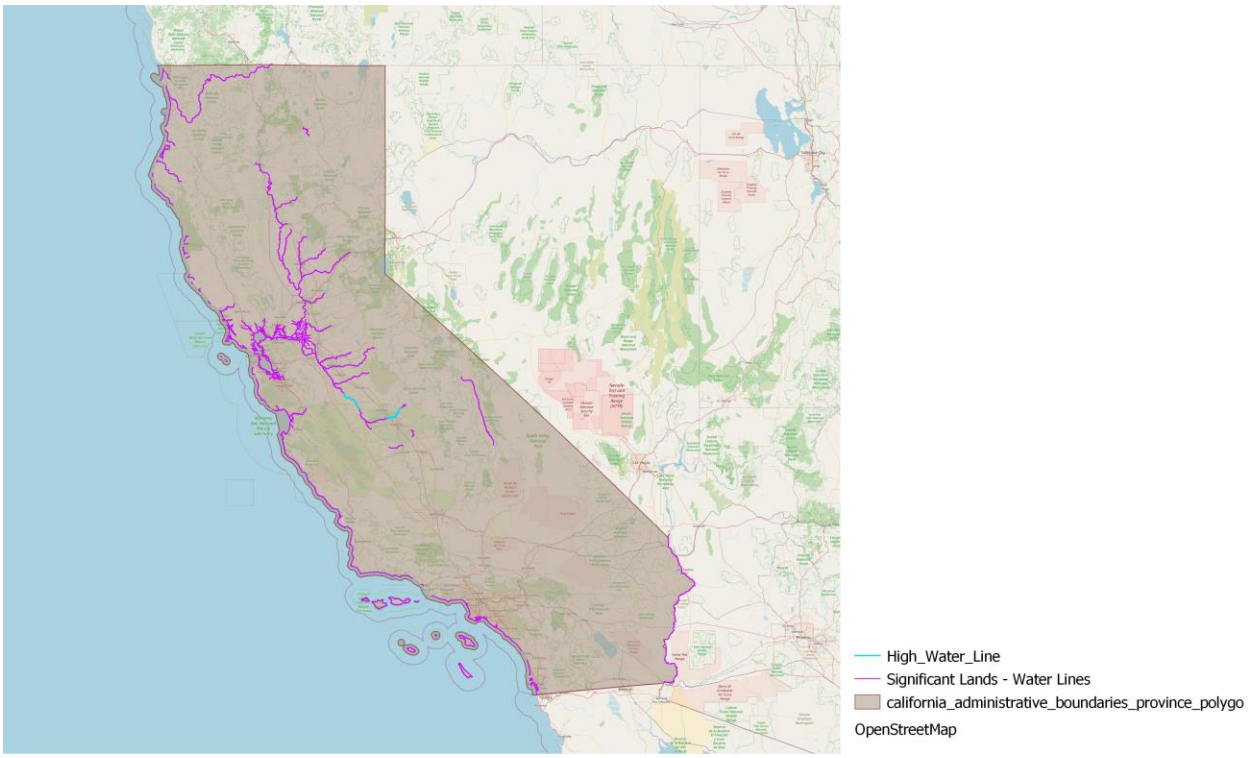


Figure 53: California's high water line & significant lands (water line) [65]

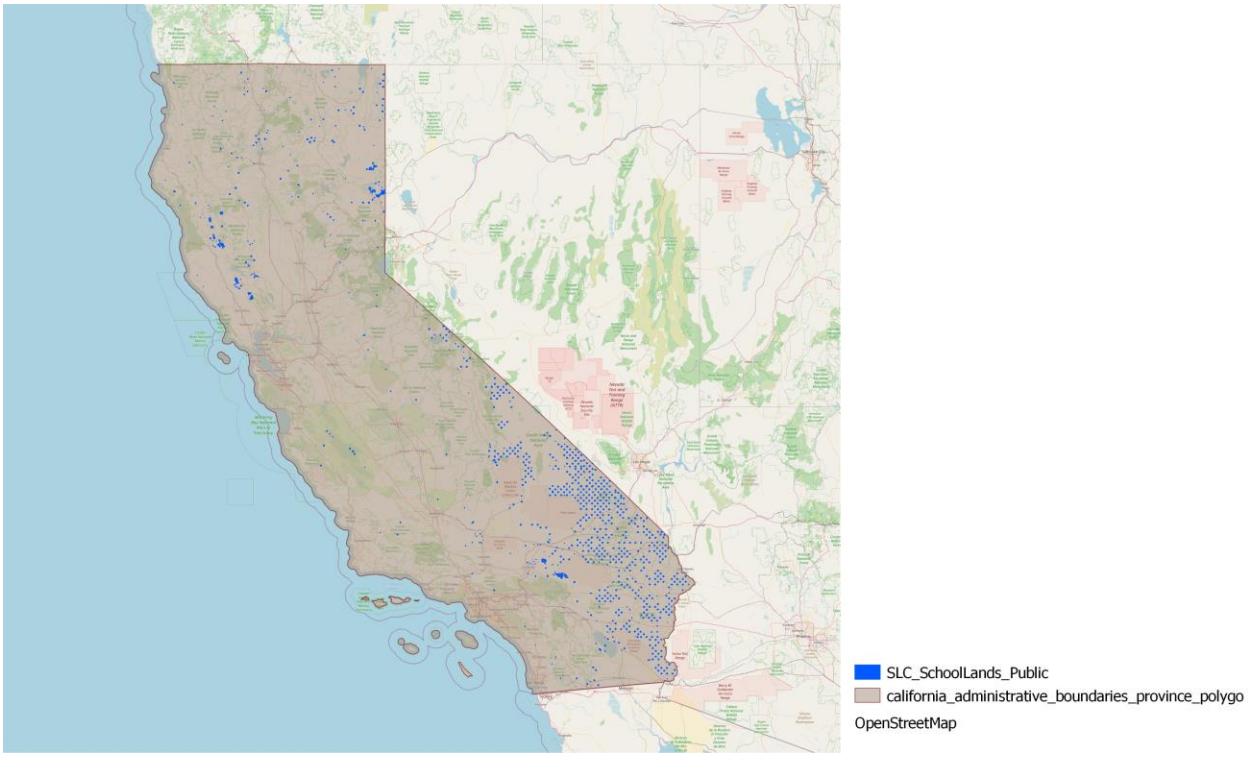


Figure 54: California's schools' lands[65]

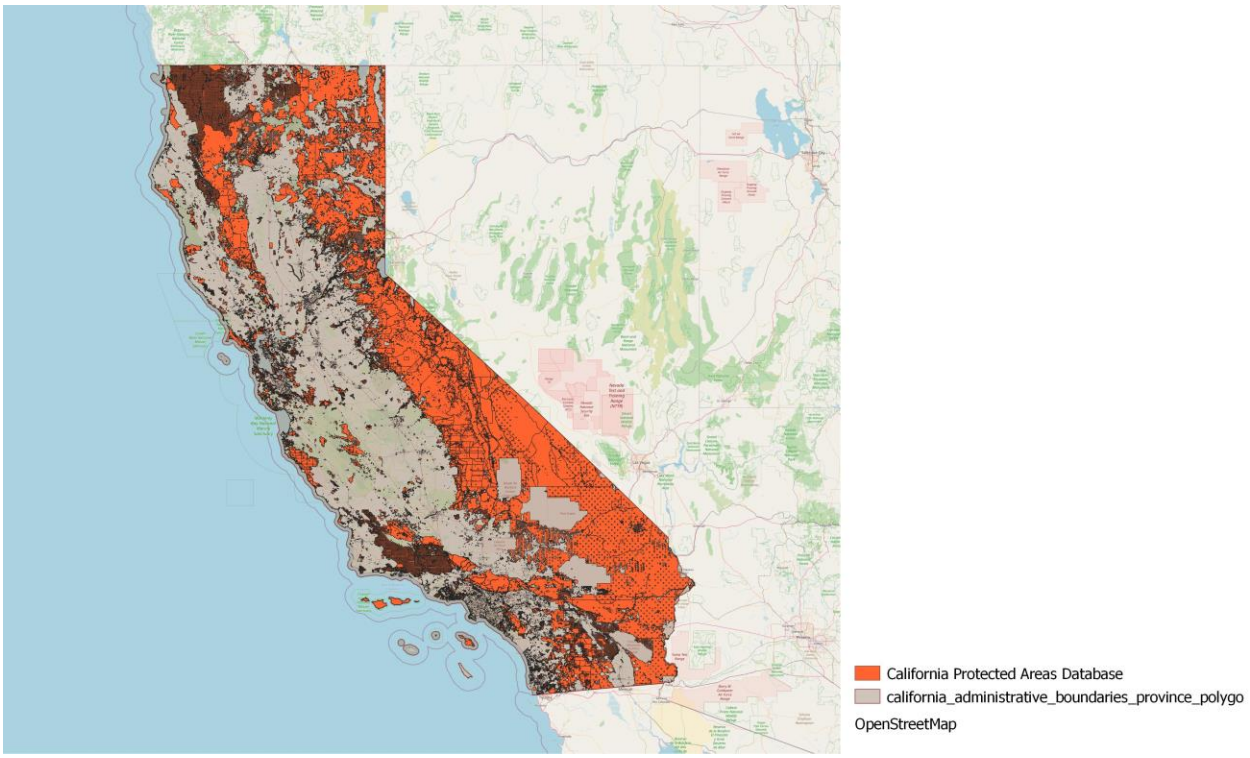


Figure 55: California's protected areas[65]

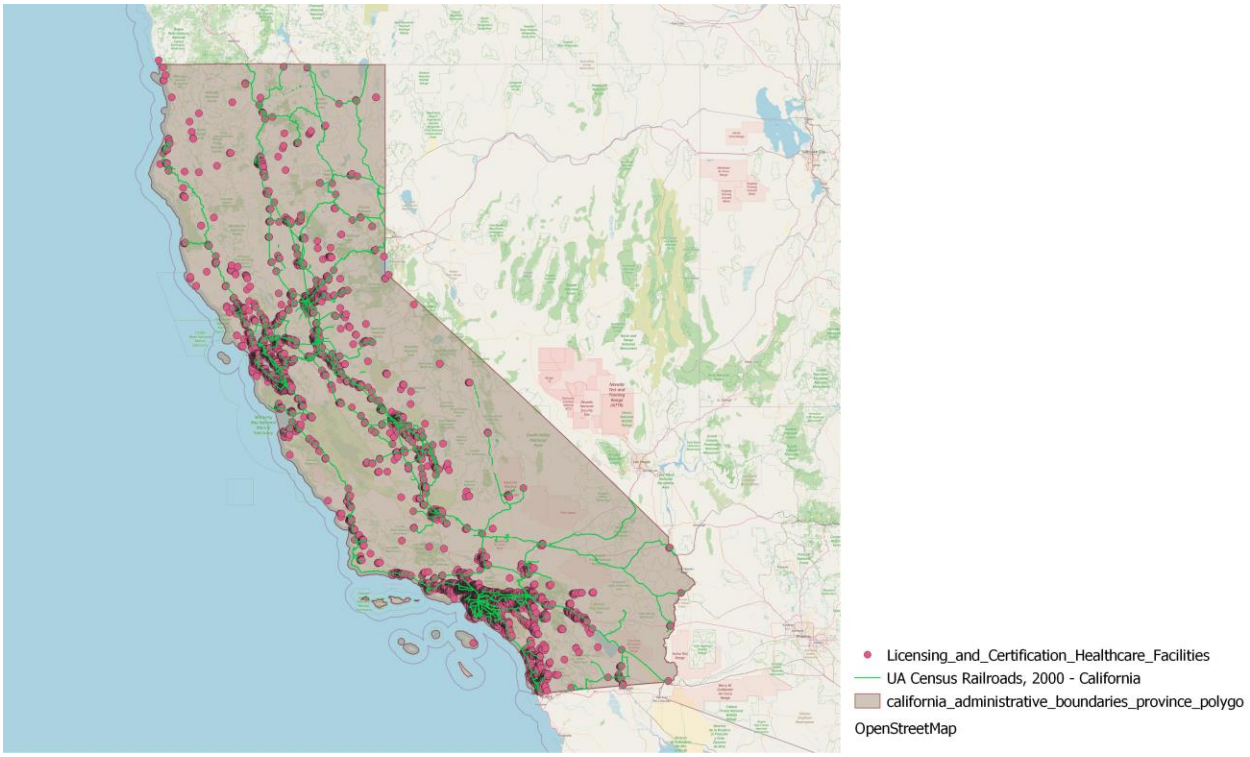


Figure 56: California's healthcare facilities & census railroads[65]

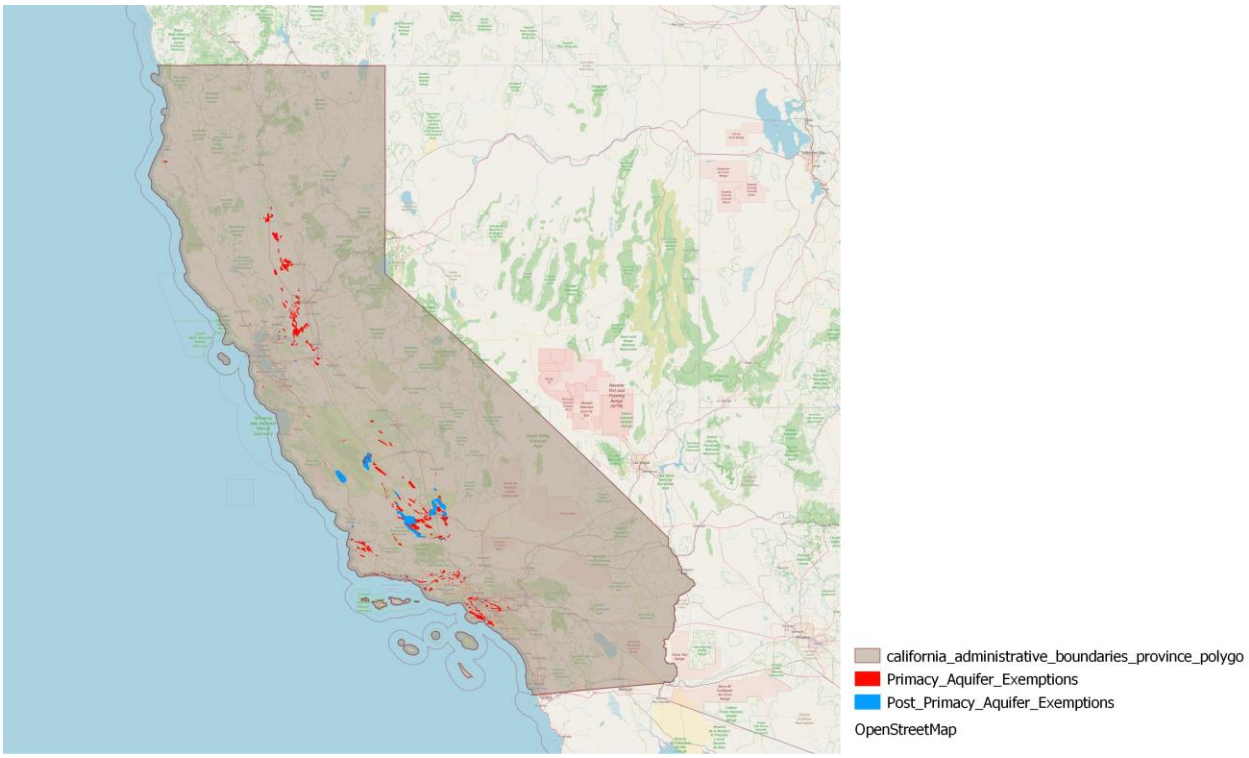


Figure 57: California's primary and post-primary aquifer exemptions[65]

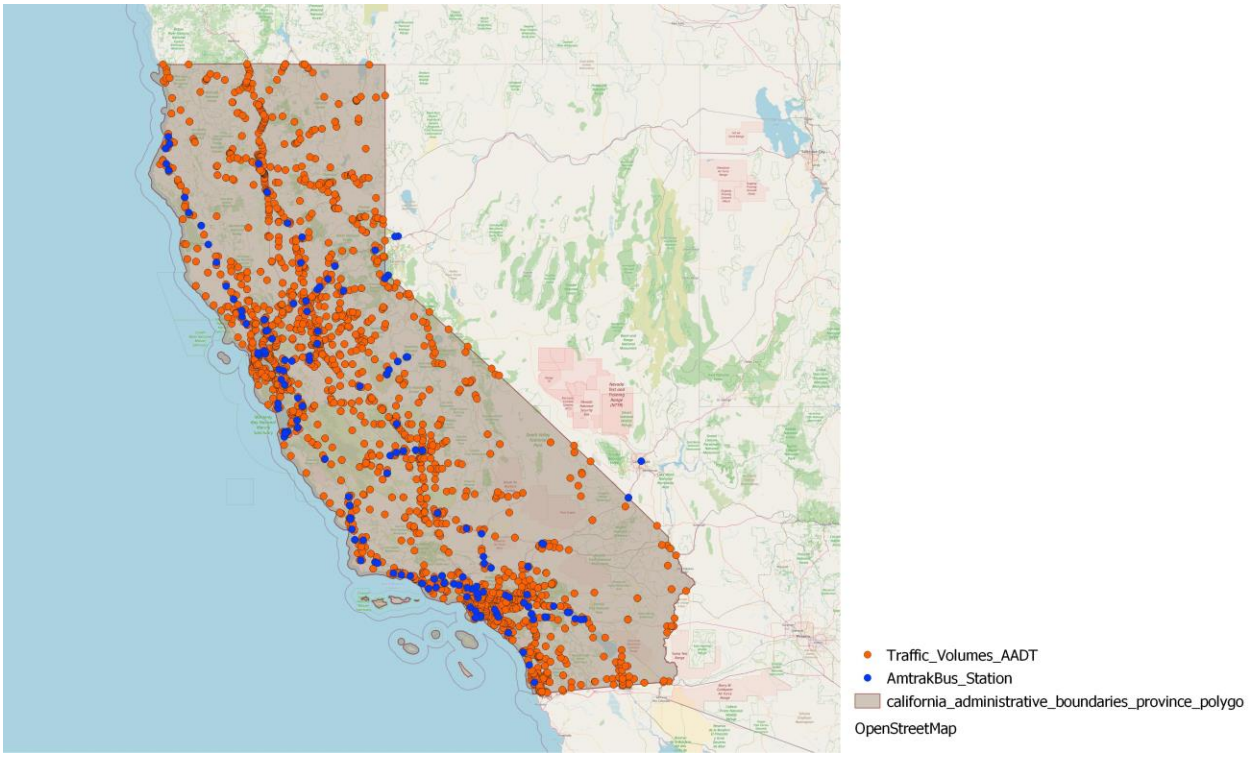


Figure 58: California's traffic volume[65]

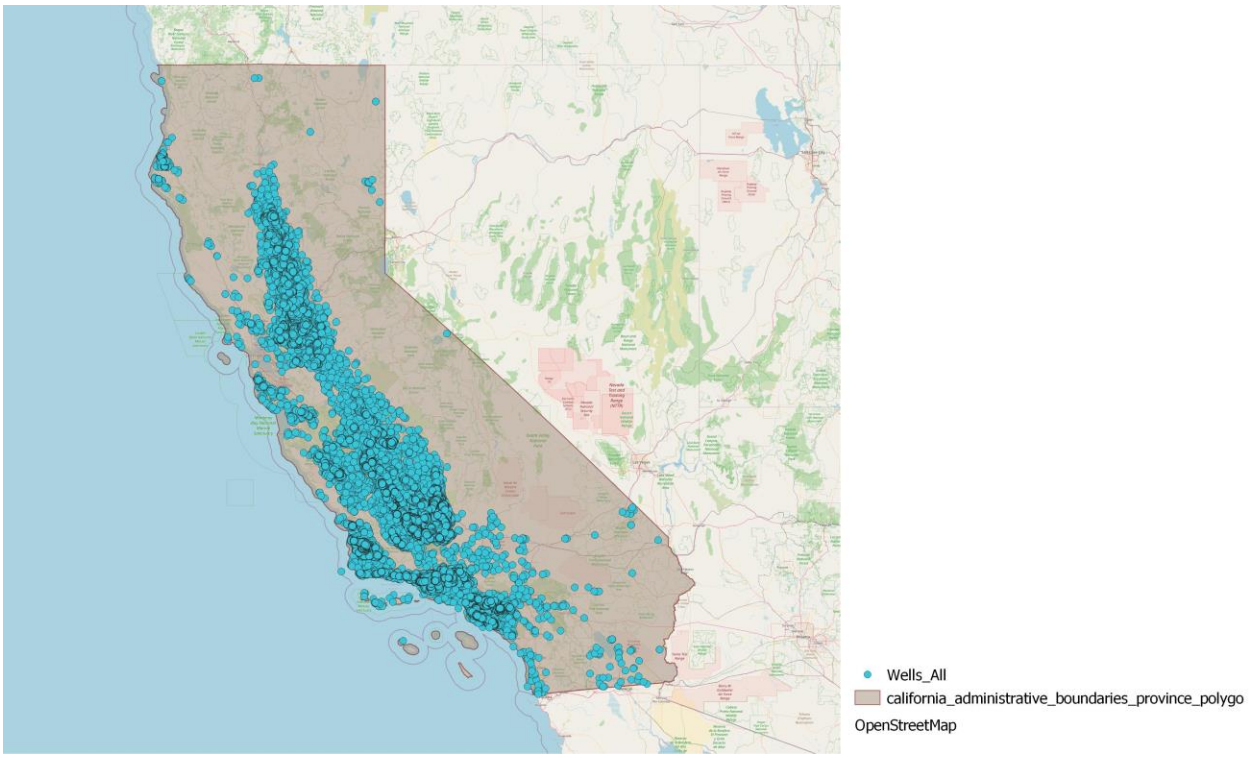


Figure 59: California's all wells[65]

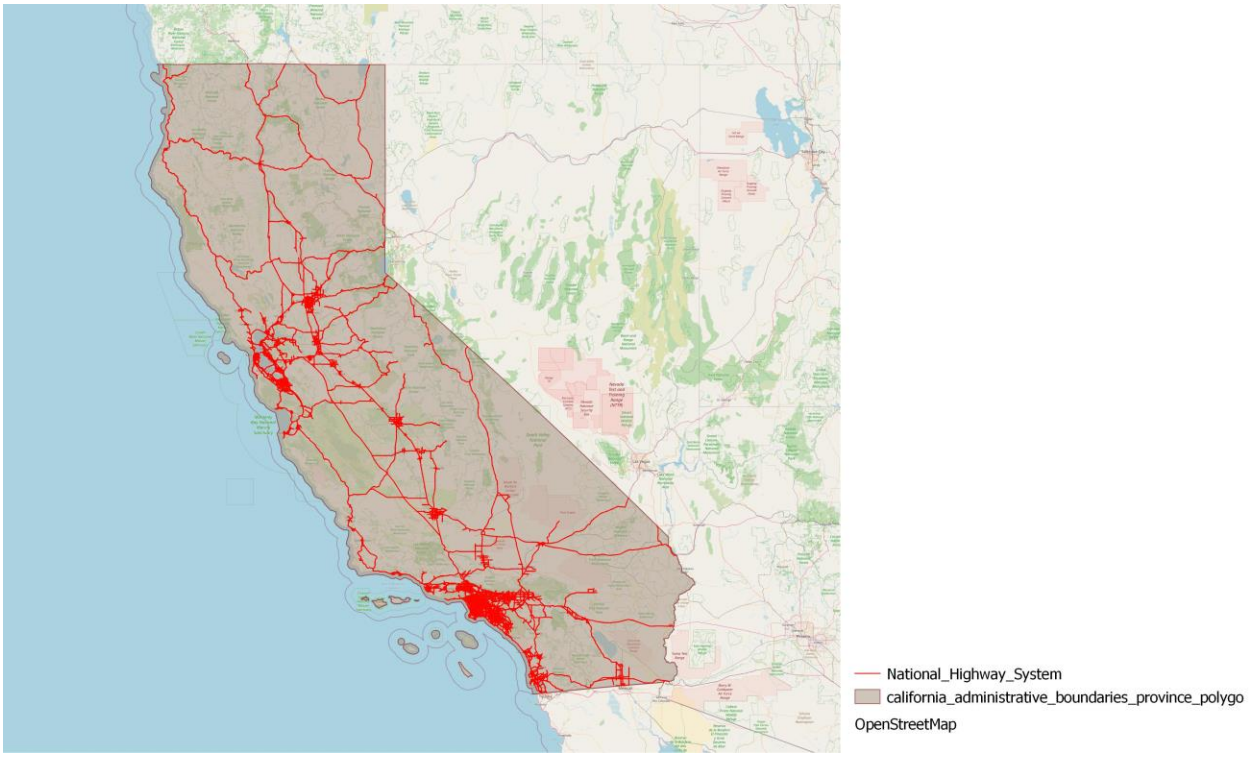


Figure 60: California's national highway system[65]

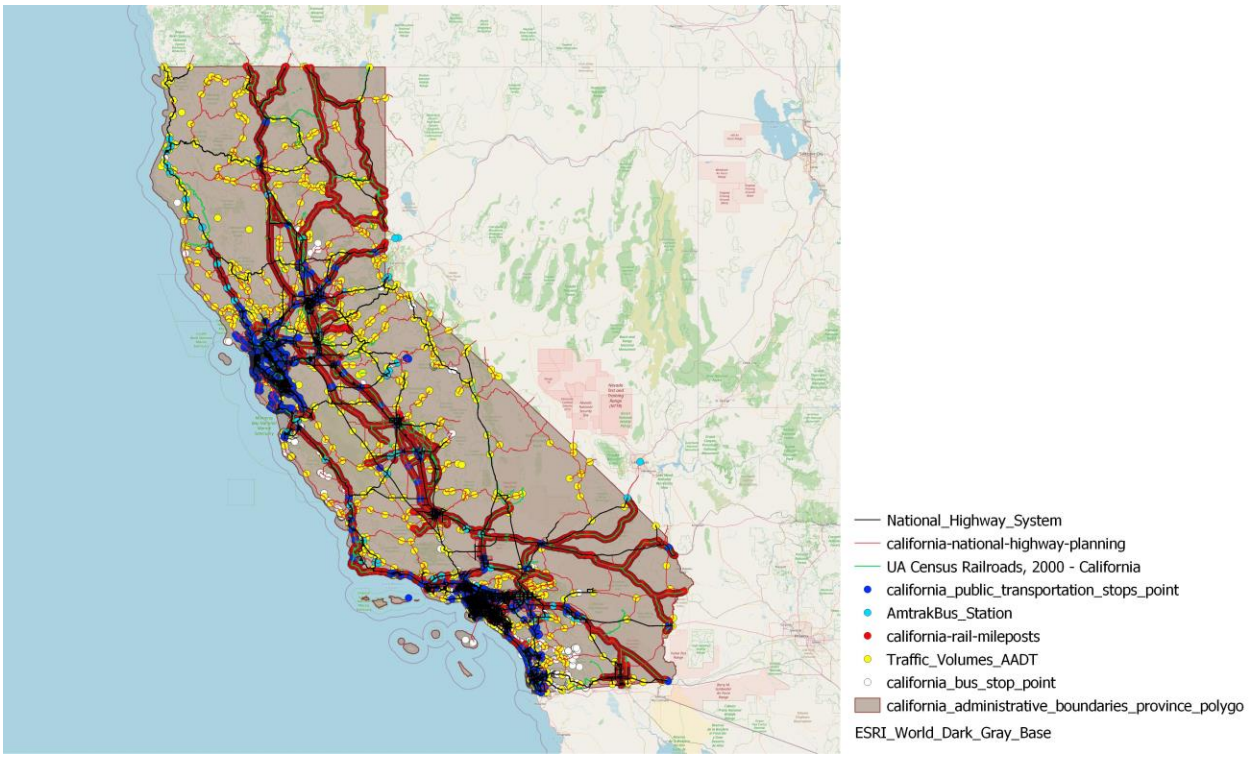


Figure 61: California's national highway & bus station & bus stops & traffic volume & railroad[65]

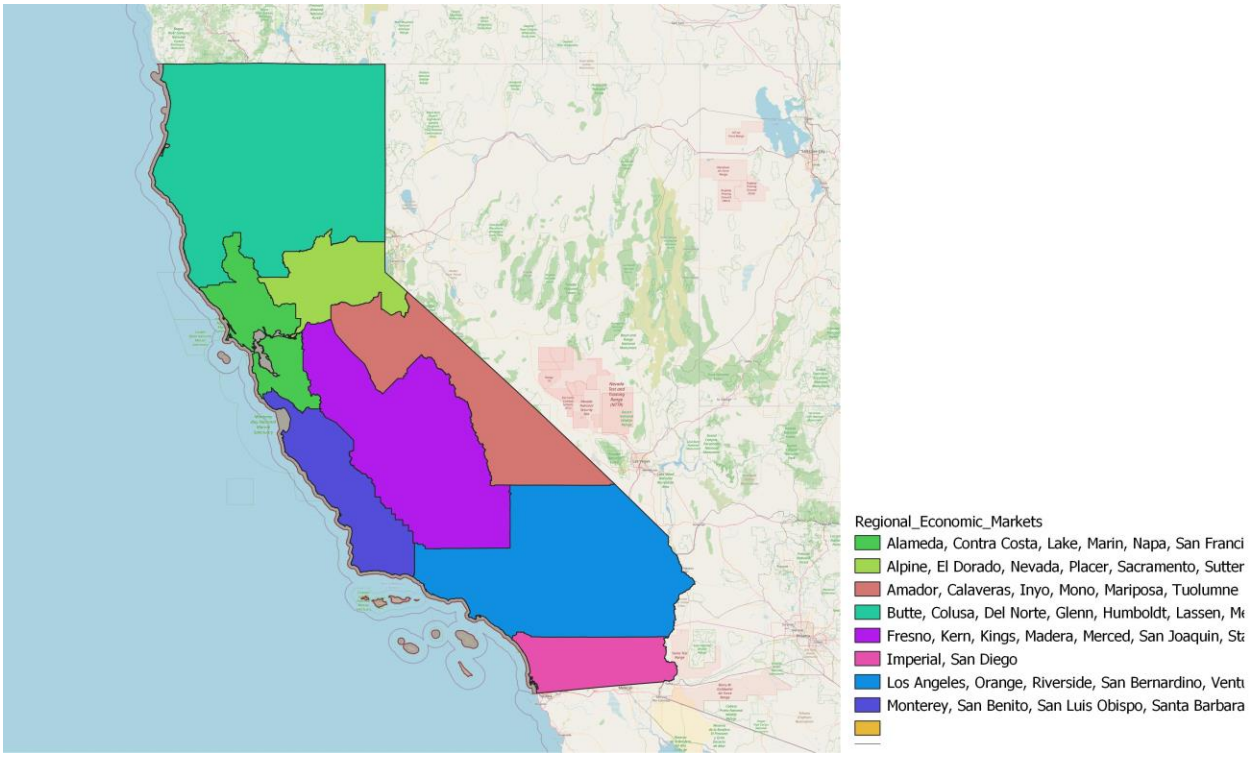


Figure 62: California's regional economic markets[65]

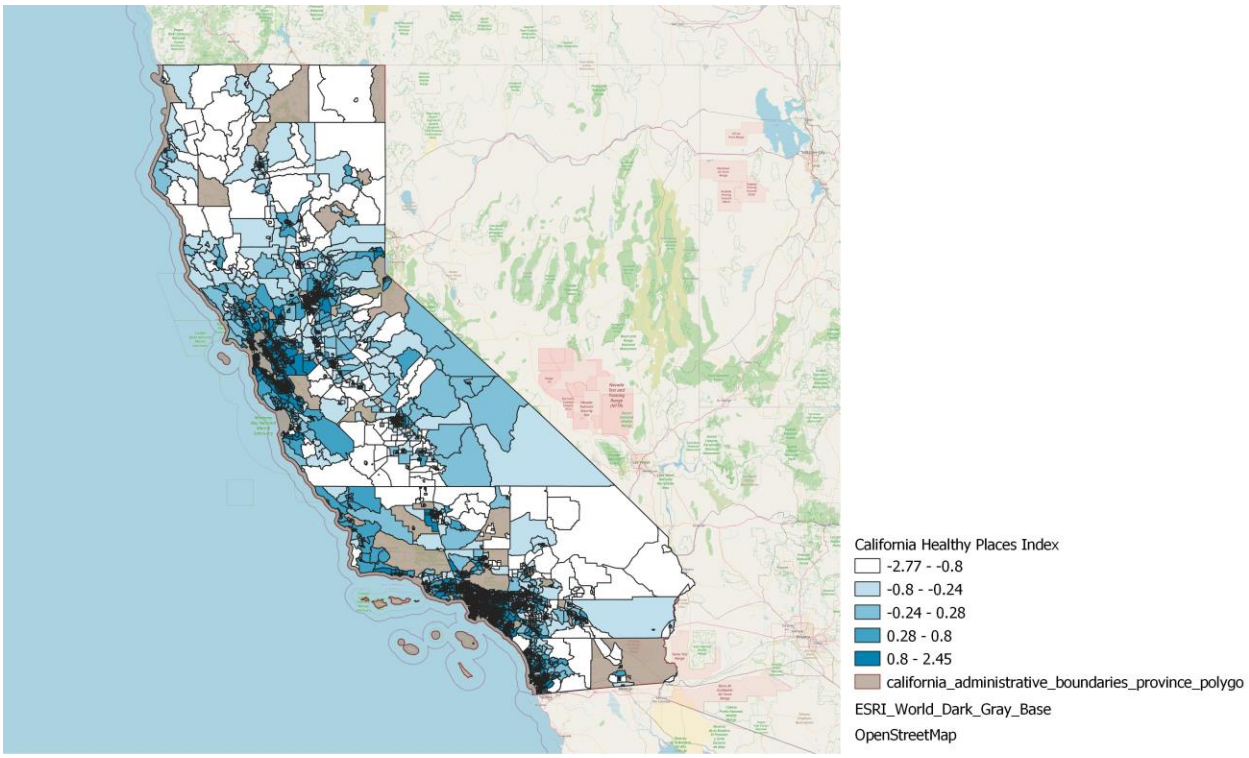


Figure 63: California's healthy places index[65]

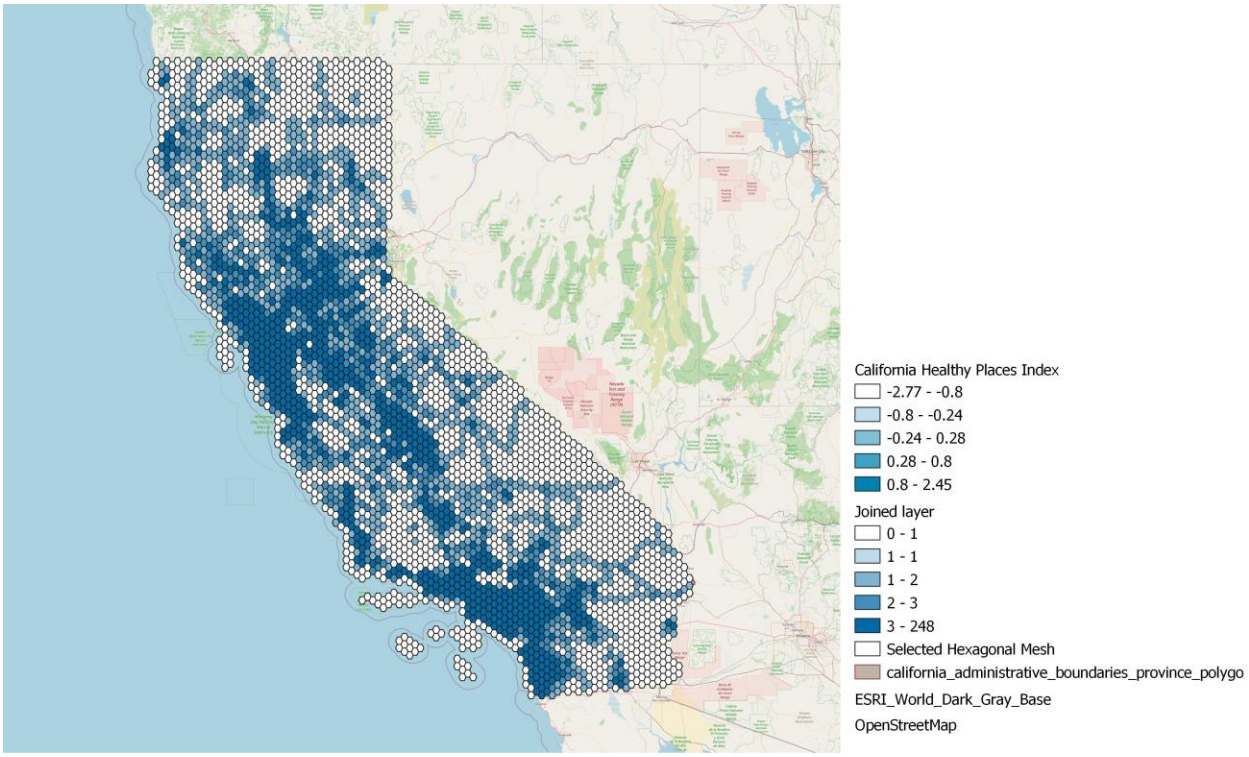


Figure 64: California's healthy places index & hexagonal mesh[65]

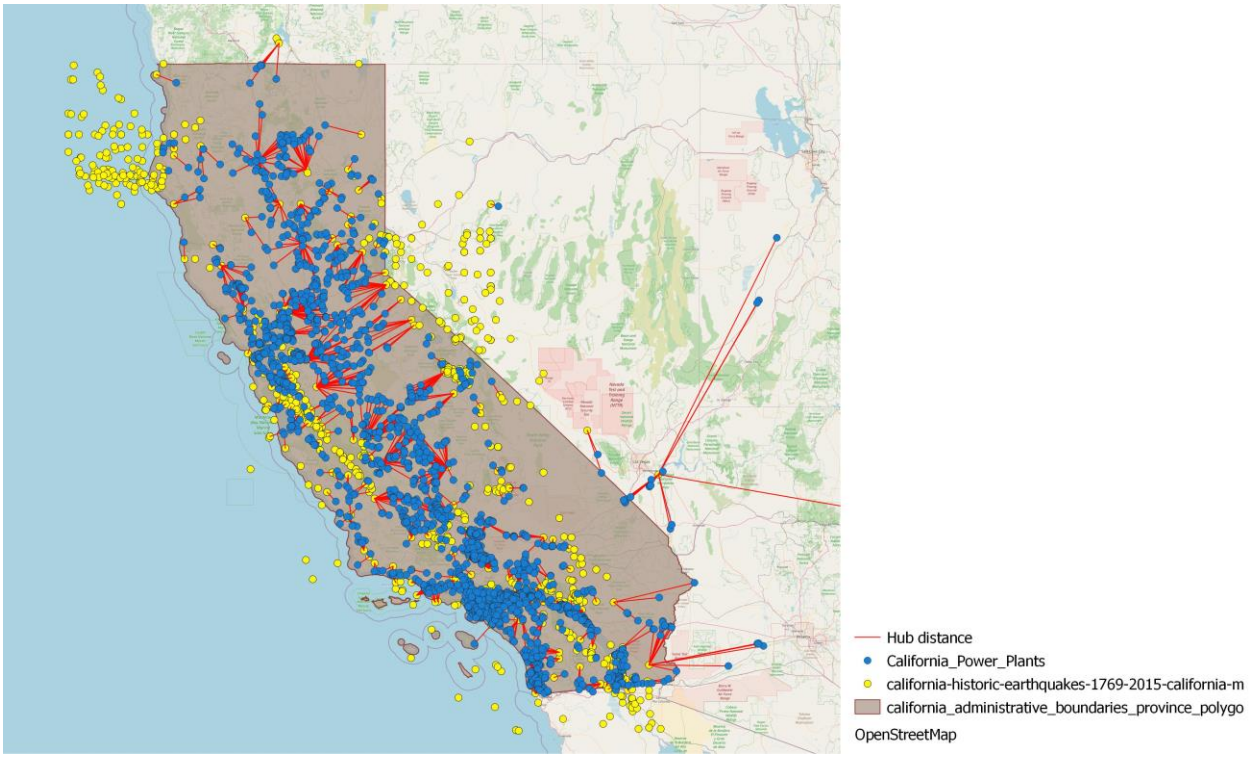


Figure 65: California's power plants, finding hub distance. [65]

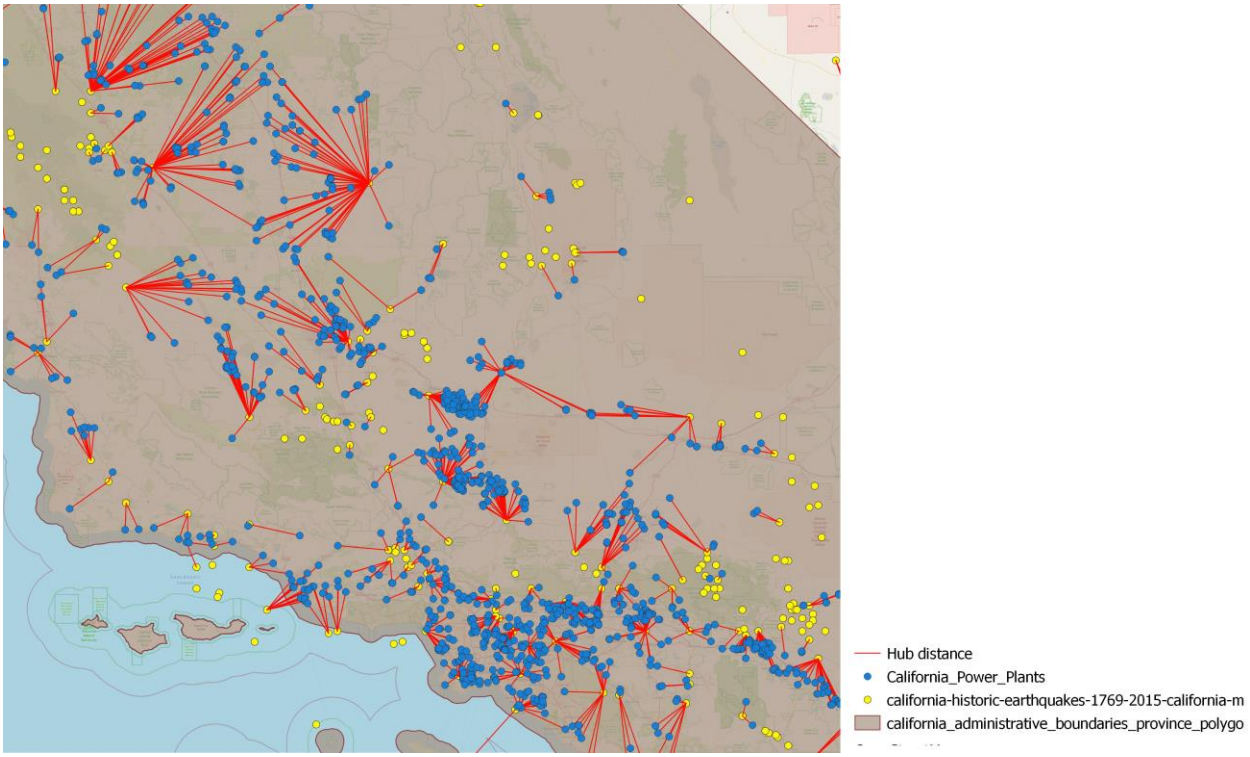


Figure 66: California's power plants, finding hub distance[65]

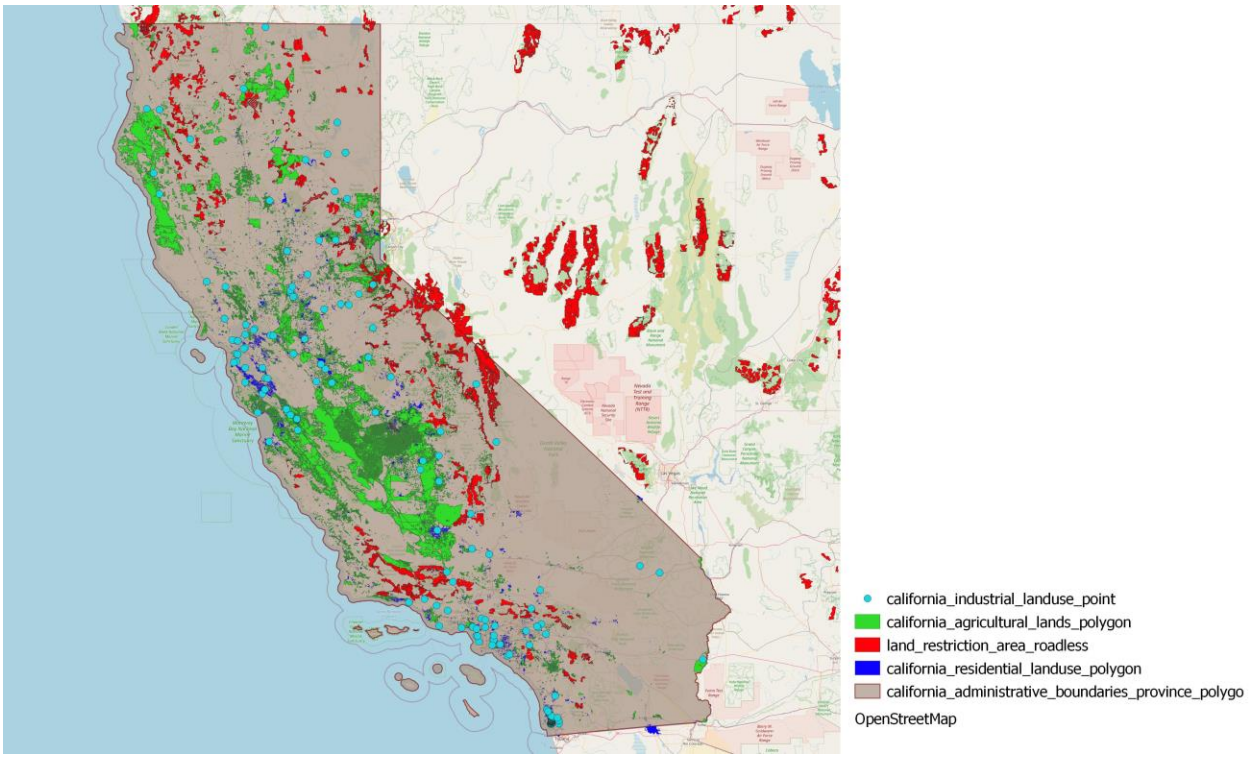


Figure 67: California's industrial land use & agricultural lands polygon & land restriction area roadless & residential land use[65]

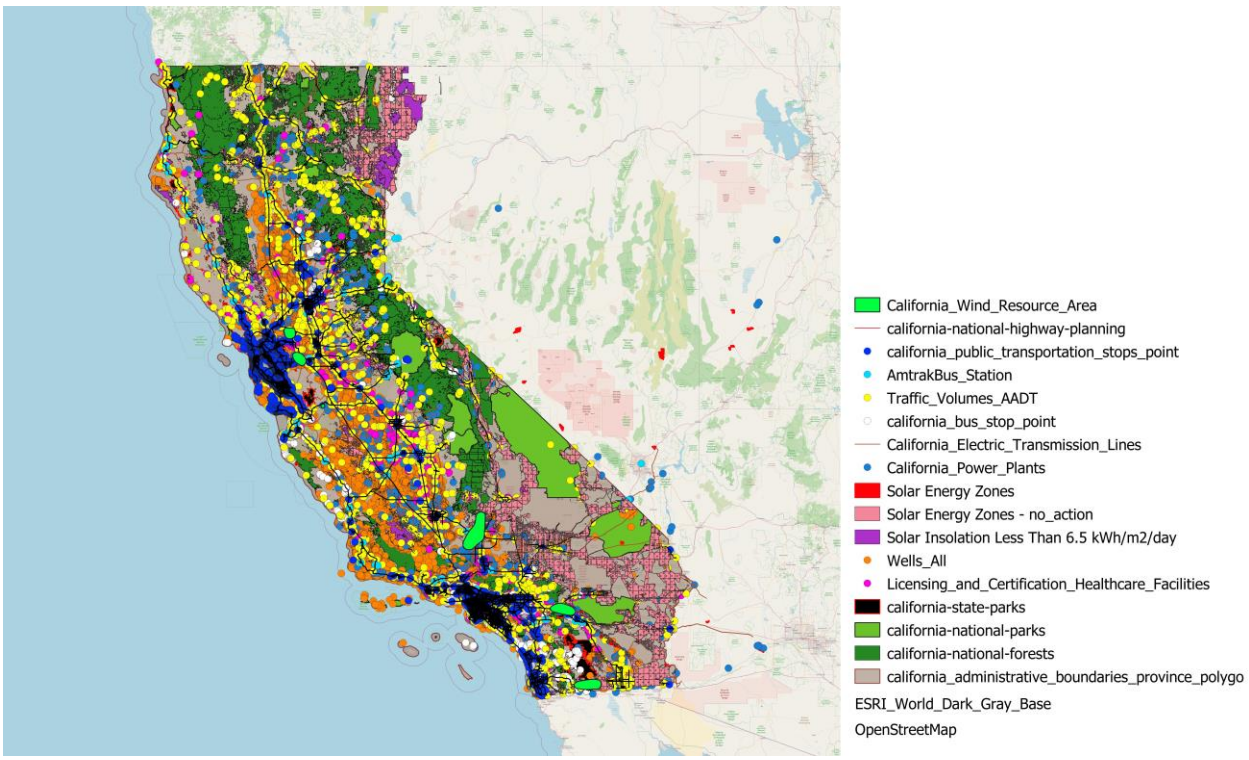


Figure 68: Cross-referencing of some important layouts[65]

Before more explanations about further sections, first, we should know about different types of data in GIS and QGIS software to get acquainted with the ultimate list of GIS formats and geospatial file extensions[65].

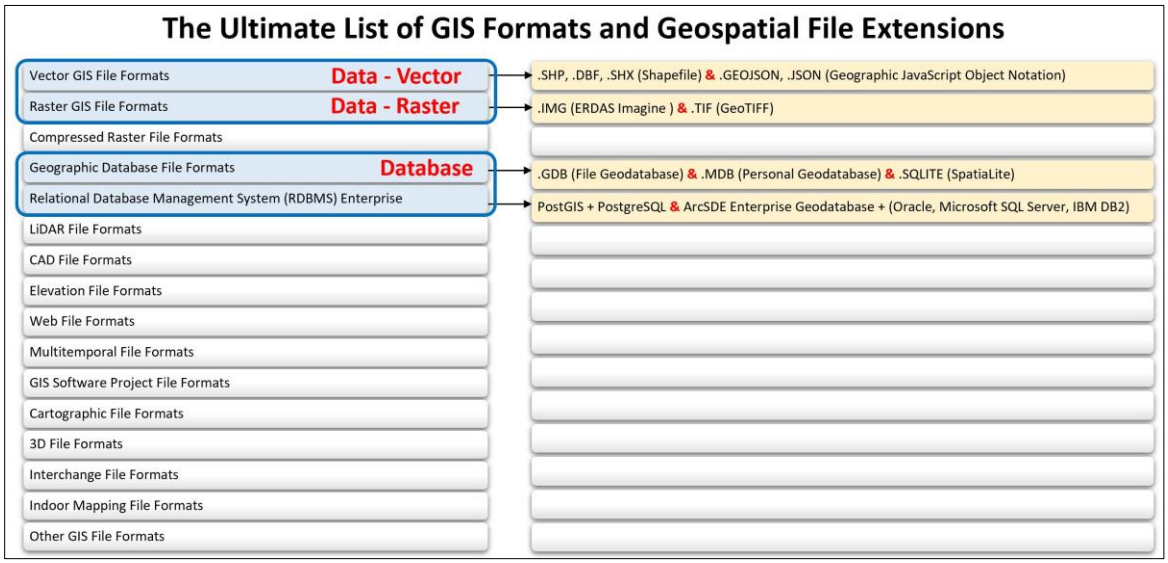


Figure 69: The ultimate list of GIS formats and geospatial file extensions

The difference between vector format and raster format has been explained in the image below.

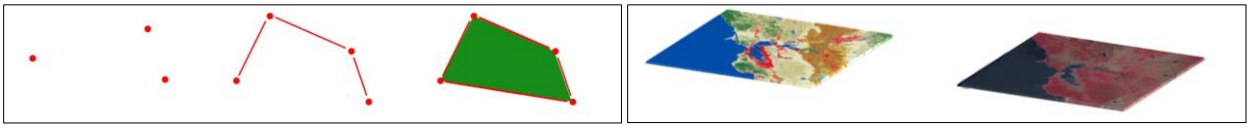


Figure 70: Difference between vector format and raster format

The figure 105 on the left shows the point, polyline and polygon, respectively, from left to right which all of them are under category of the vector format and the figure 105 on the right shows raster format of a GIS files. It should be mentioned that in our stored dataset we have only the vector format which includes points, polyline, and polygon. Also, it should be noted that as a first step of designing this plugin, in order to more focus on one type of data, only point format is considered for both analysis and calculations and also in the machine learning section.

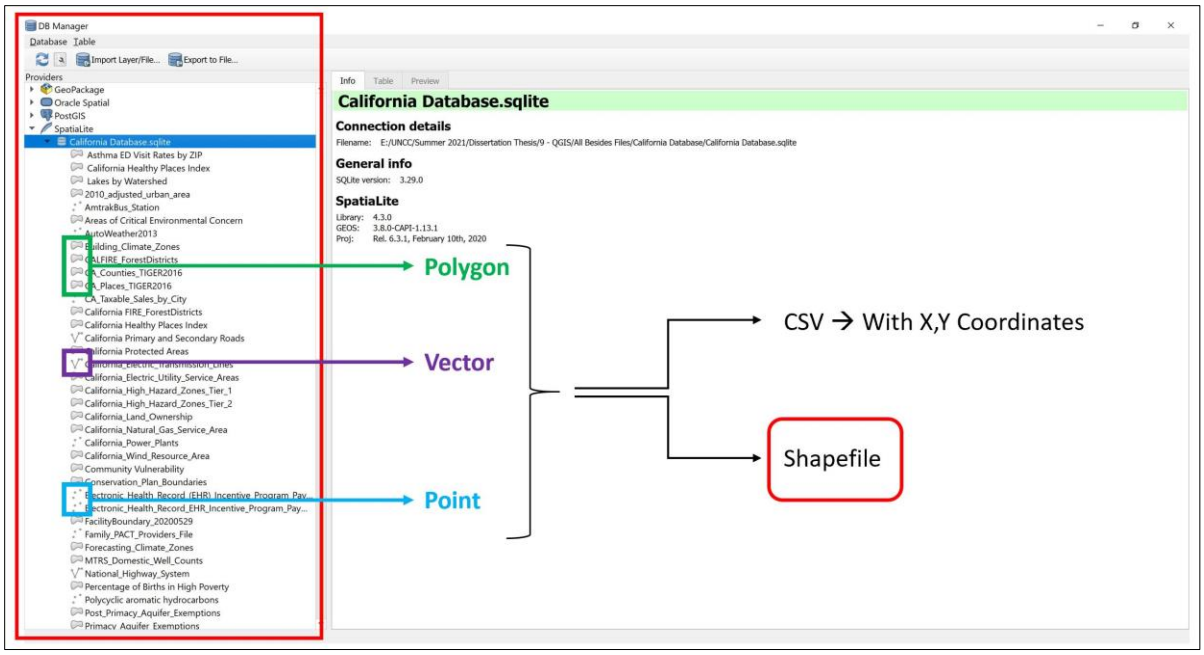


Figure 71: Different types of data format in the created database

As is mentioned before, for analyzing the layouts only the point format of vector type is used as we want to know about the exact location of different facilities.

Also, as we can see in 100 and 101 some analysis can be done in QGIS to get more familiar with the data and finding the concept of calculation for core parts of the system since QGIS has different various algorithms. One of these analysis that can be done in QGIS is finding Distance to Nearest Hub (line to hub), which is one of the algorithms of QGIS that can display lines to hubs, indicating the nearest facility to each polygon or point. Images of 100 and 101 can show some results of using this algorithm. But to reach the exact concept that we have designed in the initial steps we should implement our Python codes. We will discuss the Python code that has been written in the following sections.

One more important point that should be talked about that is QGIS has been written with applying three different code languages C++, Python, Qt. So, for developing this software it is logical to use python code programming.

5.6 – CREATING A DATABASE IN QGIS

In this step, a database was built in QGIS with all of the available and significant layouts. The database is in SQLite format and can convert to CSV format to do some analysis in Python. After creating the database in QGIS, an SQLite is saved in a selected directory and later can be imported to the QGIS for potential changes or modifications. This SQLite layer contains all of the layouts which we used for creating the database. Despite being a SQL database engine, SQLite is in no way comparable to any of the popular client/server databases like MySQL, Oracle, PostgreSQL, or SQL Server. In contrast to MySQL, Oracle, PostgreSQL, and SQL Server, SQLite is trying to solve a different problem than clients/server SQL database engines. Their core principles emphasize scalability, concurrency, centralization, and control. For each application and device, SQLite provides local data storage. In addition to simplicity and economy, SQLite emphasizes reliability, efficiency, and independence.

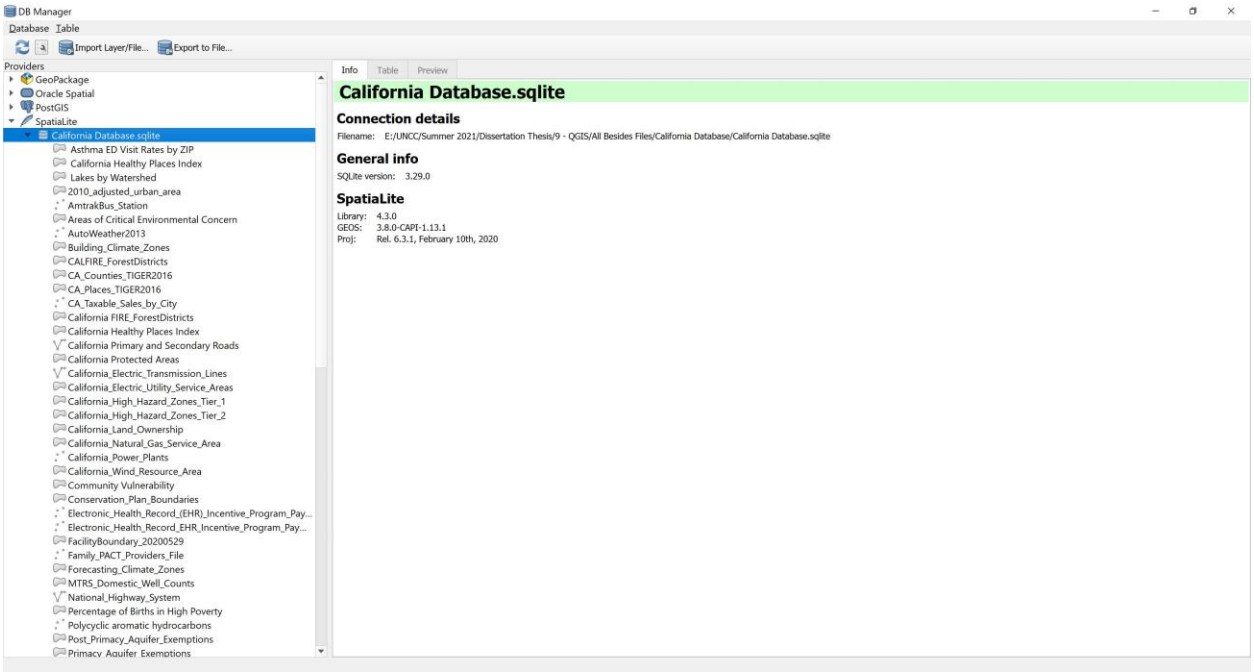


Figure 72: Creating California's database in QGIS

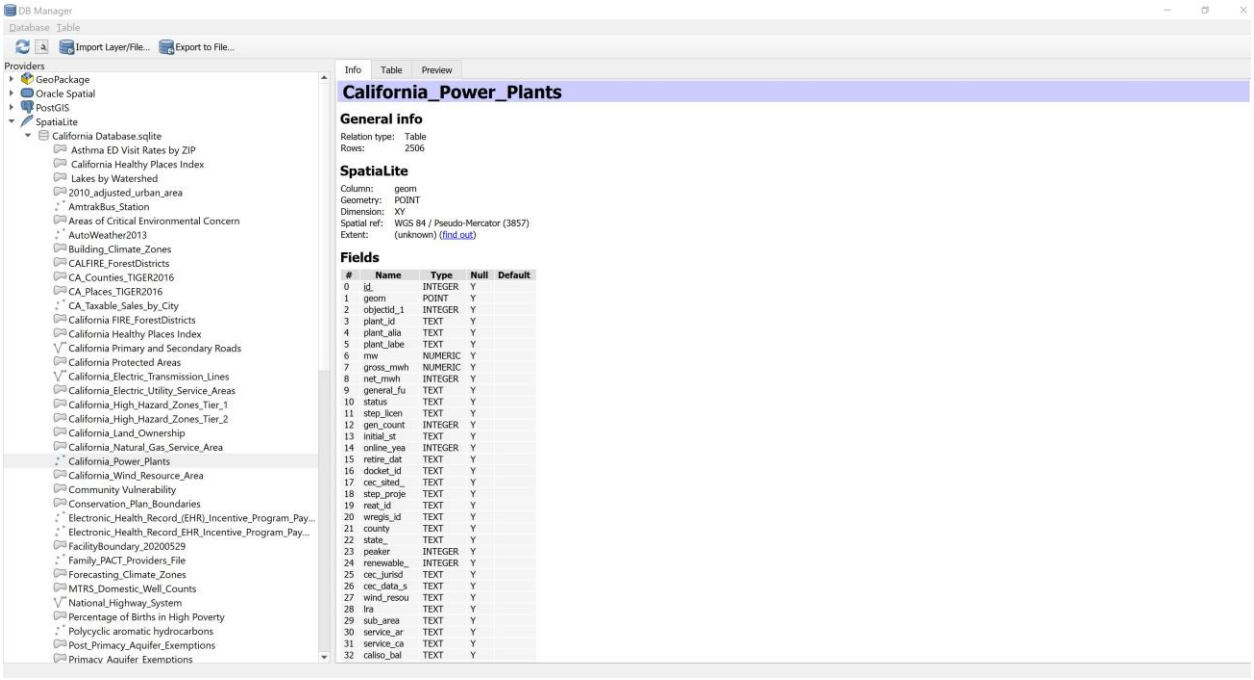


Figure 73: Showing one layout of California's database.

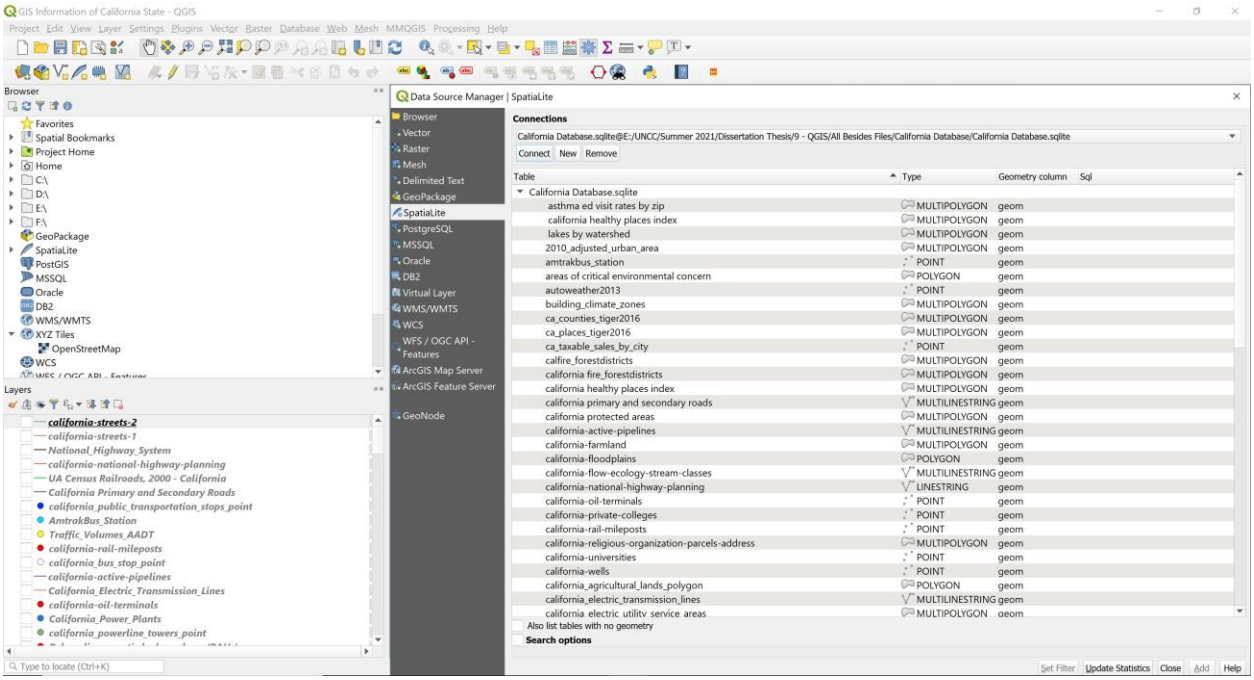


Figure 74: Opening SQLite file (database of California)

5.7 – LOGIC OF COMPUTING CORE PART OF THE SYSTEM

Before we move on to the implementation of the code with the Python code programming language, it is a good idea to explain the logic of computing and analyzing this system. For a better understanding of different analysis, a hexagonal mesh was created to calculate the number of the points, which can display the rank of each hexagon in terms of defining the specific data, parameter, or variable. Hexagon geometry with the 10-mile diameter (5-mile radius) was used for covering all the states, instead of using the circle. Also, for checking the general accessibility of the system, the hexagon grid was created in a 10-mile diameter which is high access defined for driving. This parameter can reveal that a person has access to vital facilities. According to the latest Center for Rural Studies analysis, the average distance to the nearest hospital for rural Americans is 10.5 miles, compared with 5.6 miles for suburbanites and 4.4 miles for urbanites. According to the 5-mile radius standard which can define the best access to essential facilities in urban design, we started to indexing California state with a 5-mile hexagonal grid mesh.

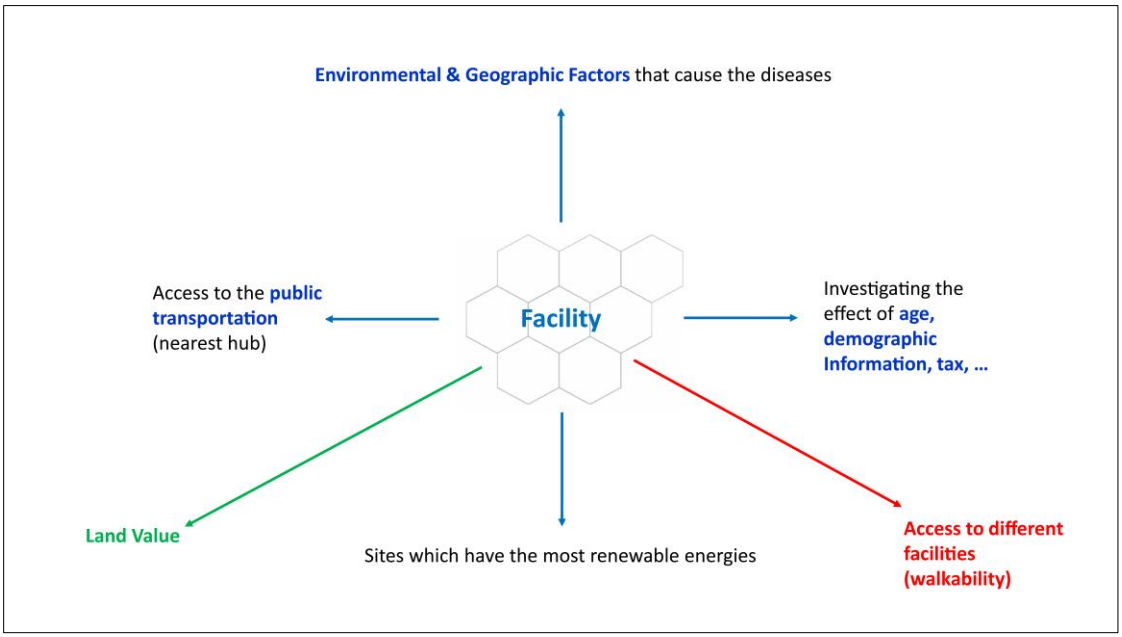


Figure 75: Schematic diagram of parameters that are involved in analysis in each cell of hexagonal mesh.

If we consider the layouts in point format, we can calculate the distance between each selected origin point and all other destination points.

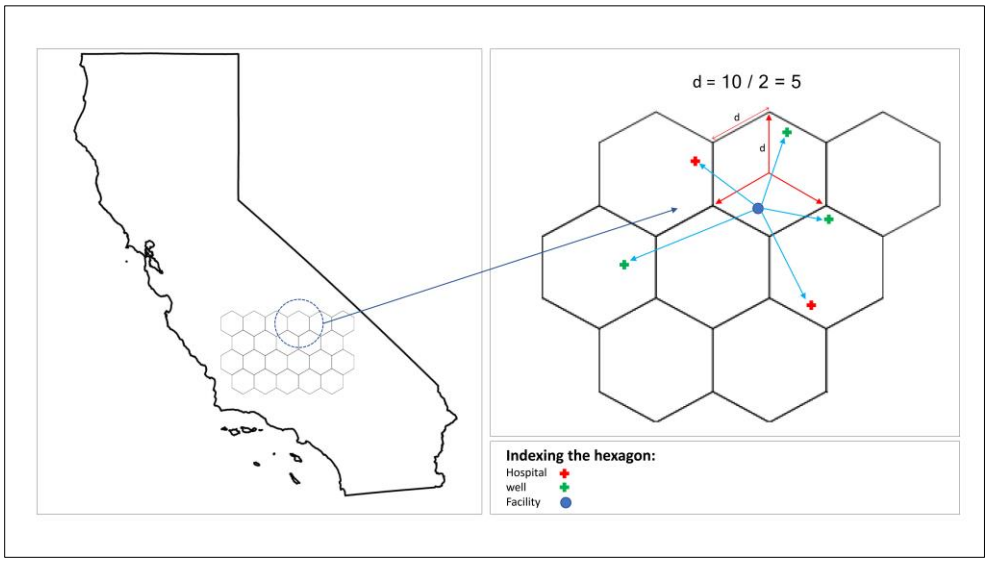


Figure 76: Indexing the state based on the results of each cell in hexagonal mesh with 10 miles diameter and calculating distance.

Also, we can calculate the number of points in each cell which can reveal how much a cell is important than other cells in terms of having more facilities inside it.

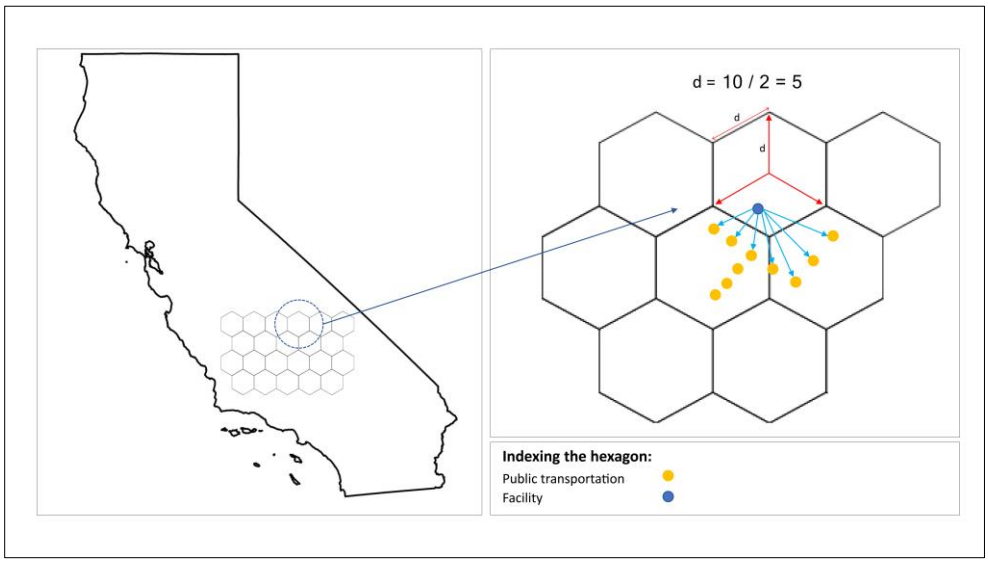


Figure 77: Indexing the state based on the results of each cell in hexagonal mesh with 10 miles diameter and counting the number of each facility in each cell.

By calculating the two parts that we have mentioned above, we can index all of the cells, so we can index the whole of the state based on the importance of each cell (parcel) which results have been derived from the above analysis. Then, by using these measurements, a map was made which presenting those cells that had a score higher than one standard deviation above the mean.

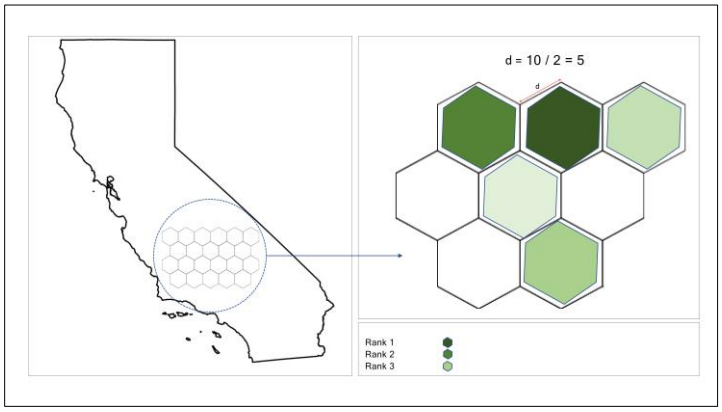


Figure 78: Indexing the state based on the results of each cell in hexagonal mesh with 10 miles diameter - calculating the distance

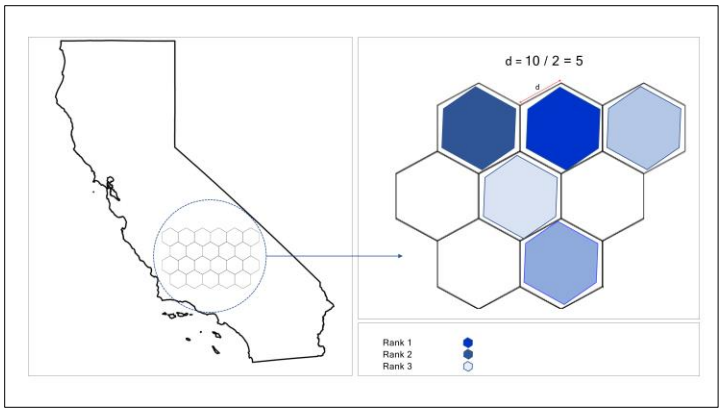


Figure 79: Indexing the state based on the results of each cell in hexagonal mesh with 10 miles diameter and counting the number of each facility in each cell.

Land parcel classification, which is an initial step towards the goal of determining the impact of our results on urban land management, hence, we used site size (parcel approaches) as a 10-mile grid mesh for our analysis. Using this homogeneous grid reduces the number of subdivisions such that boundaries better align with neighborhood units to which rule sets like land covenants apply.

5.8 – PYTHON CODE DEVELOPMENT & FEATURE ENGINEERING

After revealing the logic of the core part of this computational system, it is time to implementing this concept in the Python code programming language. For this purpose, Anaconda and Spyder which is a scientific Python development environment have been used due to their speed in calculating and returning the results. Spyder as a powerful Python IDE is boosted with advanced editing, interactive testing, debugging and introspection features so, it facilitates a more advanced level of code development. Python code in design this plugin includes 2 main tasks as following:

Task 1 - Finding all of the nearest points to the origin point that we select among all layers.

Task 2 - Finding the list of all important points based on the numbers of other facilities around the origin point that we select.

Regarding the mechanism of task 1, it should be mentioned that by running the code, it goes to the directory and path of the system that includes all of the CSV's files and reads all of them. Also, all these data (CSV's files) are saved in a single list in the code. So, by running the code, it asks the user to enter the desired facility he/she intends to perform calculations based on that facility. The selected facility is assumed to be the origin point, and the distance between the origin points to all destination points is calculated. Therefore, the nearest longitude and latitude of other facilities are obtained to the points of origin (source/home), and the latitude and longitude of each of them are stored in separate CSV files. Each of these files will have the ability to map on QGIS since all of those have X and Y coordinates. The formula for calculating the distance between two geographical points is obtained through the haversine formula. To calculate the shortest distance between two points over the earth's surface, the **Haversine formula** is applied, which gives the distance as seen from above.

Haversine formula: $a = \sin^2(\Delta\varphi/2) + \cos\varphi_1 \cdot \cos\varphi_2 \cdot \sin^2(\Delta\lambda/2)$

$c = 2 \cdot \operatorname{atan2}(\sqrt{a}, \sqrt{(1-a)})$

$d = R \cdot c$

where φ is latitude, λ is longitude, R is earth's radius (mean radius = 6,371km); note that angles must be in radians to pass through to trig functions!

Therefore, its formula in Python would be as following:

```
R = 6373.0    // Radius of the earth in km
dlon = lon2 – lon1
dlat = lat2 – lat1
a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)**2
c = 2 * atan2(sqrt(a), sqrt(1 – a))
distance = R * c    // Distance in km
```

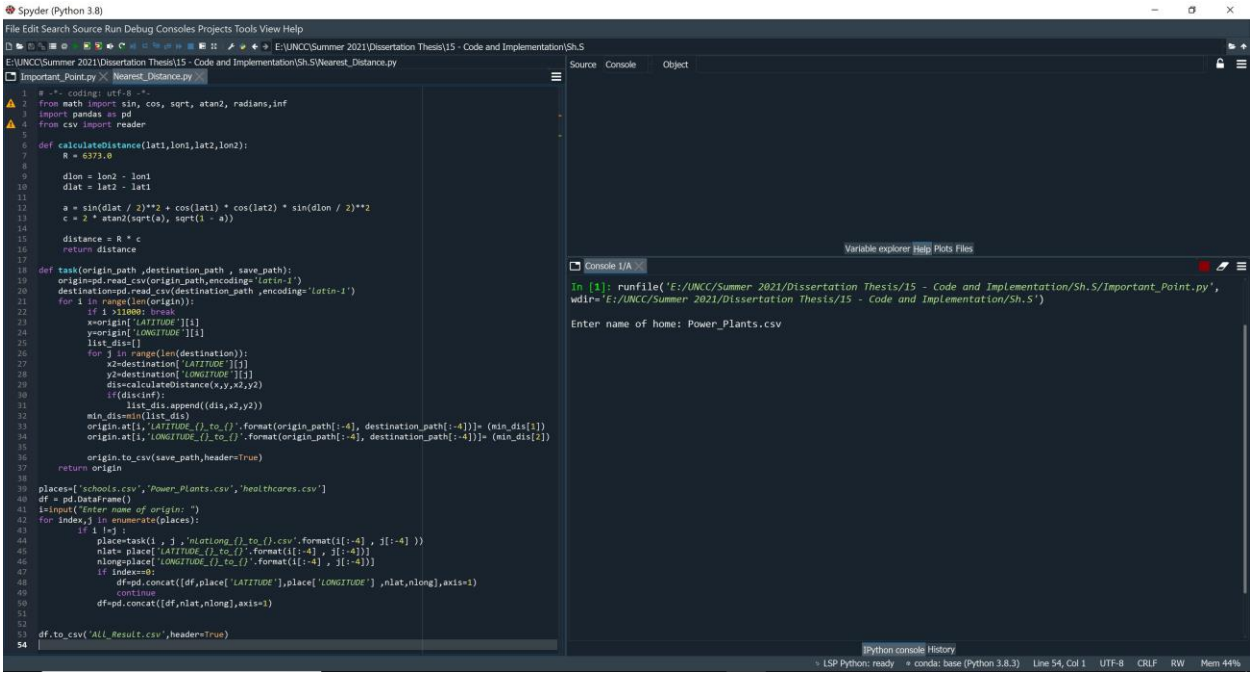
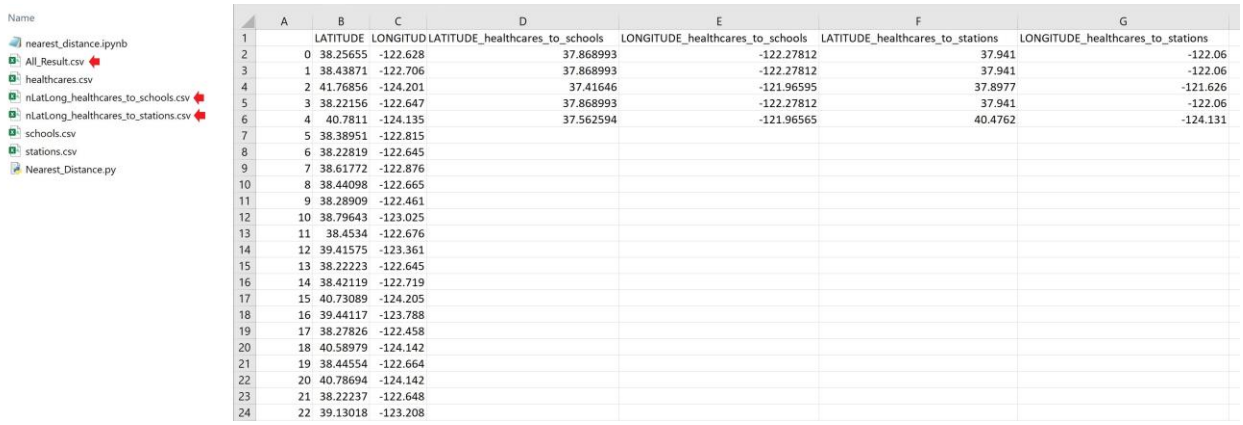


Figure 80: Implementing the code – finding the nearest facility to each origin point

	STATION_ID	STATION_N	FULL_STAT	STATION_N	STATION_T	LATITUDE	LONGITUDE	COUNTY_N	SAMPLE_C	SAMPLE_D	SAMPLE_D	LATITUDE_stations_to_schools	LONGITUDE_stations_to_schools
0	47023	Frank Siefe	Frank Siefe	01N01E01	Groundwat	33	-121	Yolo	1	########	########	37.868993	-122.27812
1	6134	01N01E33I	01N01E33I	01N01E33I	Groundwat	37.8868	-121.868	Contra Cos	1	########	########	37.712904	-122.11172
2	6135	01N01W03	01N01W03	01N01W03	Groundwat	37.9583	-121.967	Contra Cos	9	########	########	37.712904	-122.11172
3	6136	01N01W04	01N01W04	01N01W04	Groundwat	37.9663	-121.973	Contra Cos	9	########	########	37.712904	-122.11172
4	6137	01N01W07	01N01W07	01N01W07	Groundwat	37.946	-122.016	Contra Cos	9	########	########	37.712904	-122.11172

Figure 81: Results of running task 1



Name

- nearest_distance.ipynb
- All_Result.csv
- healthcares.csv
- nLatLong_healthcares_to_schools.csv
- nLatLong_healthcares_to_stations.csv
- schools.csv
- stations.csv
- Nearest_Distance.py

		LATITUDE	LONGITUD	LATITUDE_healthcares_to_schools	LONGITUDE_healthcares_to_schools	LATITUDE_healthcares_to_stations	LONGITUDE_healthcares_to_stations
2	0	38.25655	-122.628	37.868993	-122.27812	37.941	-122.06
3	1	38.43871	-122.706	37.868993	-122.27812	37.941	-122.06
4	2	41.76856	-124.201	37.41646	-121.96595	37.8977	-121.626
5	3	38.22156	-122.647	37.868993	-122.27812	37.941	-122.06
6	4	40.7811	-124.135	37.562594	-121.96565	40.4762	-124.131
7	5	38.38951	-122.815				
8	6	38.22819	-122.645				
9	7	38.61772	-122.876				
10	8	38.44098	-122.665				
11	9	38.28909	-122.461				
12	10	38.79643	-123.025				
13	11	38.4534	-122.676				
14	12	39.41575	-123.361				
15	13	38.22223	-122.645				
16	14	38.42119	-122.719				
17	15	40.73089	-124.205				
18	16	39.44117	-123.788				
19	17	38.27826	-122.458				
20	18	40.58979	-124.142				
21	19	38.44554	-122.664				
22	20	40.78694	-124.142				
23	21	38.22237	-122.648				
24	22	39.13018	-123.208				

Figure 82: Results of saving the different layouts

School		Station		Healthcare		Park		Power		Road		Public Transportation	
Longitude	Latitude	Longitude	Latitude	Longitude	Latitude	Longitude	Latitude	Longitude	Latitude	Longitude	Latitude	Longitude	Latitude
37.51521	-122.09705												
37.51521	-122.09705												
37.51521	-122.09705												

Figure 83: Schematic of code working mechanism.

According to the above images, when we run the code, it asks the user to enter the origin layout for all calculations. Origin points are points that are considered as base points and the distance of all other points with them is calculated. In the test above, we entered the schools' points as an origin point. So, after running the code longitude and latitude of the nearest station to the first point (first row – first school) will be fined and saved in front of the first school point. The same process is repeated for all points. After finding the nearest station for each school point, the code continues the calculations and finds the nearest healthcare to each schools' point, and so on.

Task 2 somehow is similar to task 1 but they have different logic. Regarding the mechanism of task 1, it should be mentioned that by running the code, it goes to the directory and path of the system that includes all of the CSV's files and reads all of them. Also, all of these data (CSV's files) are saved in a single list in the code. The difference between tasks 1 & 2 is: by running the code, it asks the user to enter the desired facility he/she intends to perform calculations based on that facility. So, the user enters the origin points. If there is at least one point from each of the other facilities within a 5-mile radius of the origin point, it stores that origin point in a separate file called Cluster 1. This is to determine the degree of importance of each point related to the other points and to store points of the same degree of importance in a separate cluster. And we can make sure that within a 5-mile (8.04 km which has demonstrated in the code below) radius so 10-mile diameter there is at least one point of each of the essential facilities. We can see this process in the images below.

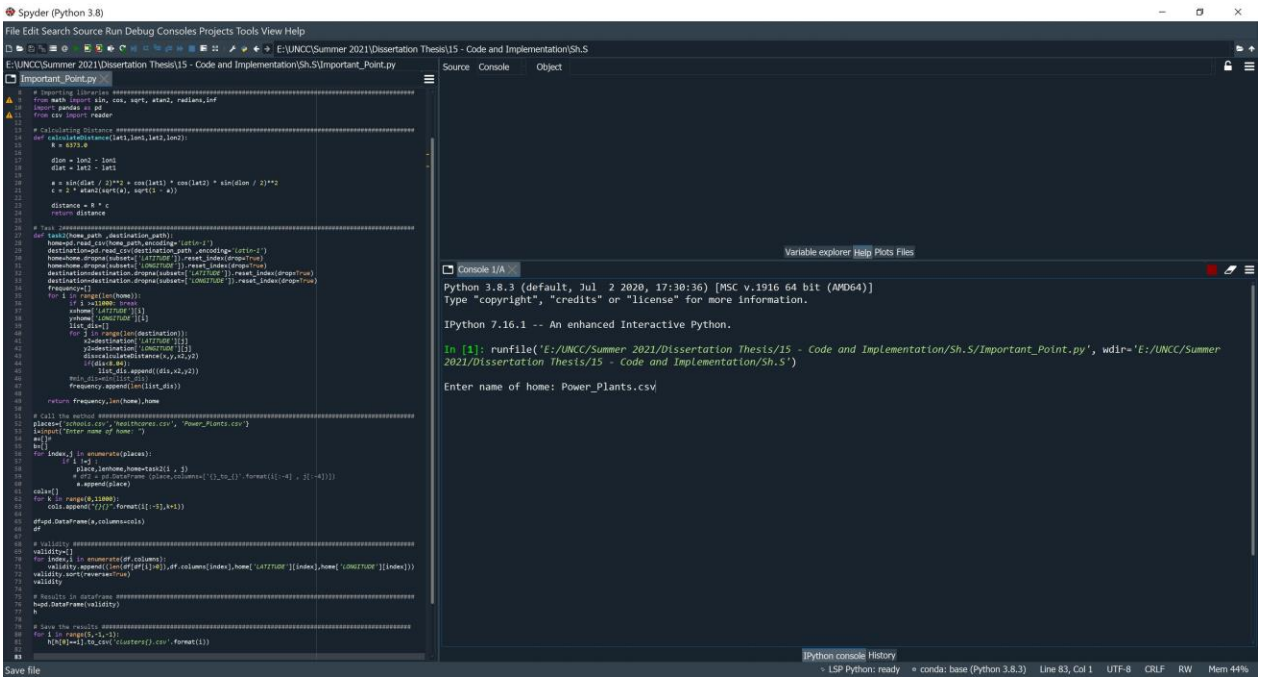


Figure 84: Python code of Task 2

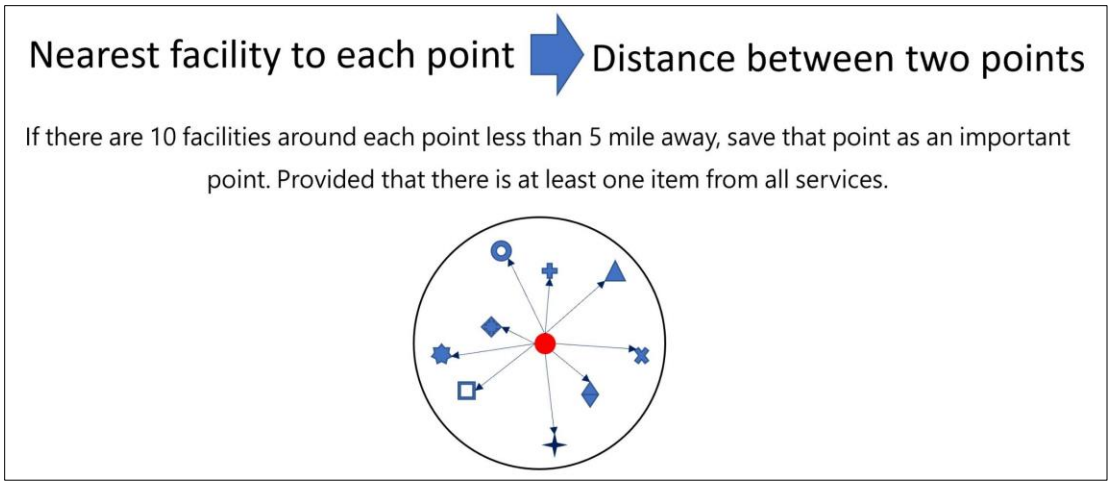


Figure 85: Checking availability of all facilities around each origin point

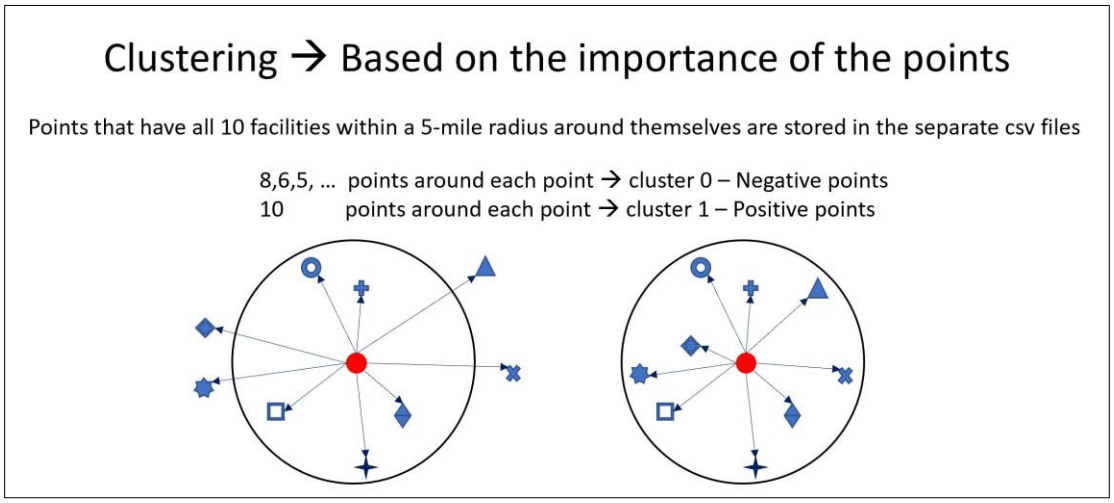


Figure 86: Clustering the points based on their importance

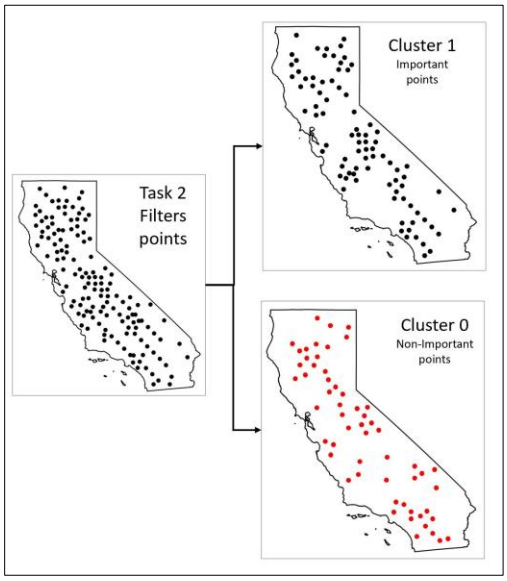
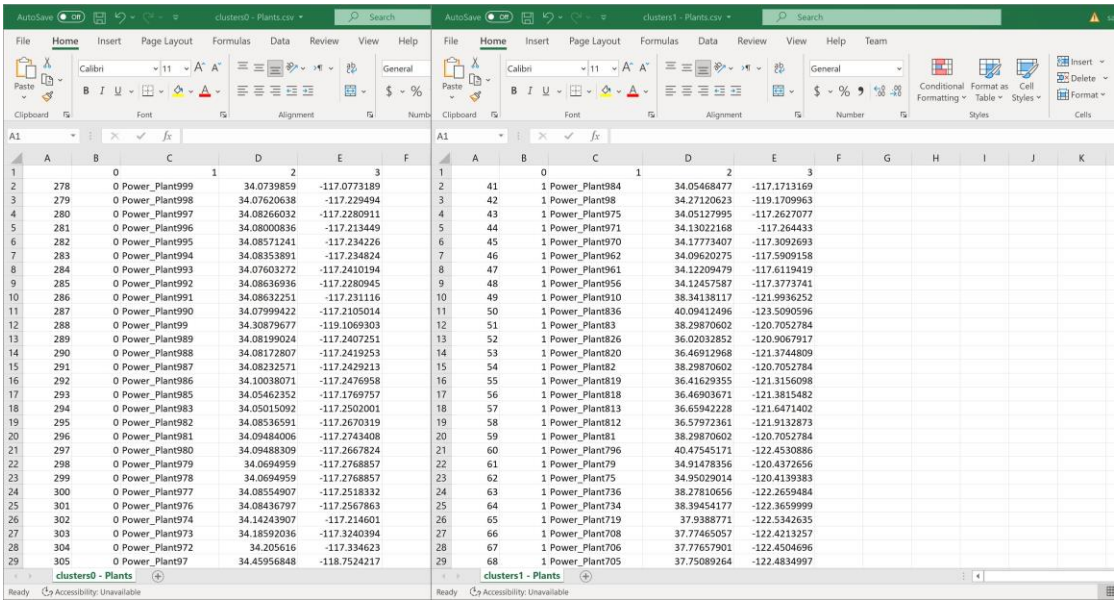


Figure 87: Process of filtering data to reach to the final result

Two final CSV files including cluster 0 which indicates nonimportant points and cluster 1 which indicates important points. Also, it is important to mention that in the machine learning section as we have the plan to use binary classification method for returning the result, so during implementing task 2 we need to add one column to each CSV file as a target column (0 for demonstrating non-important points and 1 for demonstrating important points) to show the difference of points.



	A	B	C	D	E	F
1		0	1	2	3	
2	278	0	Power_Plant999	34.0739859	-117.0773189	
3	279	0	Power_Plant998	34.07620638	-117.229494	
4	280	0	Power_Plant997	34.08266032	-117.2280911	
5	281	0	Power_Plant996	34.08000836	-117.213449	
6	282	0	Power_Plant995	34.08571241	-117.234226	
7	283	0	Power_Plant994	34.08353891	-117.234824	
8	284	0	Power_Plant993	34.07603272	-117.2410194	
9	285	0	Power_Plant992	34.08636936	-117.2280945	
10	286	0	Power_Plant991	34.08632251	-117.231116	
11	287	0	Power_Plant990	34.07999422	-117.2105014	
12	288	0	Power_Plant99	34.30879677	-119.1069303	
13	289	0	Power_Plant989	34.08199024	-117.2407251	
14	290	0	Power_Plant988	34.08172807	-117.2419253	
15	291	0	Power_Plant987	34.08232571	-117.2429213	
16	292	0	Power_Plant986	34.10038071	-117.2476958	
17	293	0	Power_Plant985	34.05462352	-117.1769757	
18	294	0	Power_Plant983	34.05015092	-117.2502001	
19	295	0	Power_Plant982	34.08536591	-117.2670319	
20	296	0	Power_Plant981	34.09484006	-117.2743408	
21	297	0	Power_Plant980	34.09488309	-117.2667824	
22	298	0	Power_Plant979	34.0694959	-117.2768857	
23	299	0	Power_Plant978	34.0694959	-117.2768857	
24	300	0	Power_Plant977	34.08554907	-117.2518332	
25	301	0	Power_Plant976	34.08436797	-117.2567863	
26	302	0	Power_Plant974	34.14243907	-117.214601	
27	303	0	Power_Plant973	34.18592036	-117.3240394	
28	304	0	Power_Plant972	34.205616	-117.334623	
29	305	0	Power_Plant97	34.45956848	-118.7524217	

	A	B	C	D	E	F	G	H	I	J	K
1		0	1	2	3						
2	41	1	Power_Plant984	34.05468477	-117.1713169						
3	42	1	Power_Plant98	34.27120623	-119.1709963						
4	43	1	Power_Plant975	34.05127995	-117.2627077						
5	44	1	Power_Plant971	34.13022168	-117.264433						
6	45	1	Power_Plant970	34.17773407	-117.3092693						
7	46	1	Power_Plant962	34.09620275	-117.5909158						
8	47	1	Power_Plant961	34.12209479	-117.6119419						
9	48	1	Power_Plant956	34.12457587	-117.3773741						
10	49	1	Power_Plant910	38.34138117	-121.9936252						
11	50	1	Power_Plant836	40.09412496	-123.5090596						
12	51	1	Power_Plant83	38.29870602	-120.7052784						
13	52	1	Power_Plant826	36.02032852	-120.9067917						
14	53	1	Power_Plant820	36.46912968	-121.3744809						
15	54	1	Power_Plant82	38.29870602	-120.7052784						
16	55	1	Power_Plant819	36.41629355	-121.3156098						
17	56	1	Power_Plant818	36.46903671	-121.3815482						
18	57	1	Power_Plant813	36.65942228	-121.6471402						
19	58	1	Power_Plant812	36.57972361	-121.9132873						
20	59	1	Power_Plant81	38.29870602	-120.7052784						
21	60	1	Power_Plant796	40.47545171	-122.4530886						
22	61	1	Power_Plant79	34.91478356	-120.4372656						
23	62	1	Power_Plant75	34.95029014	-120.4139383						
24	63	1	Power_Plant736	38.27810656	-122.2659484						
25	64	1	Power_Plant734	38.39454177	-122.3659999						
26	65	1	Power_Plant719	37.9388771	-122.5342635						
27	66	1	Power_Plant708	37.77465057	-122.4213257						
28	67	1	Power_Plant706	37.77657901	-122.4504696						
29	68	1	Power_Plant705	37.75089264	-122.4834997						

Figure 88: CSV files for cluster 0 and cluster1

As it has been shown in the image above, column 0 will be used for machine learning sections and indicates which points are important in the analysis, and which ones are not important.

The last point that should be mentioned in this section is that when task 2 runs it asks the user to enter the origin point (source layer) so, all calculations will be done based on the selected layer as the origin point. The logic behind this concept is that the selected layer will create further hubs. Therefore, this logic gives the opportunity to the user each desired layer to complete all calculations and returning the result.

5.9 – MACHINE LEARNING, MODEL EXPERIMENTS

After completing the python code in the previous section, we have the results of the important points and non-important points in separate clusters which have been saved in two CSV files. We used the cluster 1.CSV file for checking the method of clustering through KNN unsupervised machine learning. However, we merged two CSV files into one single CSV (All_Clusters) to implement a binary classification machine learning section. So, we used the machine learning algorithm to get the final results. For getting final results we used binary classification which is a supervised machine learning technique where the goal of categorical class label prediction is to predict discrete, unordered values such as Pass/Fail, Positive/Negative, and Default/Not-Default, etc and was boosted with KNN algorithm. The mechanism of working the machine learning algorithm is training based on the points obtained from cluster 0 and cluster 1. A new point with its Longitude and Latitude is entered, If the entered point is outside of all clusters, the answer is No. If the entered point is inside of one of the clusters, 2 condition occurs a) If the entered point is close to the range of label 0, the answer is No and If the entered point is close to the range of label 1, the answer is Yes. By this logic, even one point be inside the cluster for filtering the non-important and important point, with help of labeling we can be sure that we have reached the most correct answer. In this logic clustering will use to separate between 2 important points, one which is inside the cluster is more valuable than the one which is outside of the cluster. In image 90, colorful points show both categories inside each cluster. Highlighted colors (dark green) indicate important points and fainter colors (light green) indicate the non-important points. The last point which should be mentioned in this section is that If clustering was done before filtering, only all points would be placed in different clusters based on their location, without knowing their importance. This means that we may have had up to 10 hospitals in the area without

a school, a park, or other facilities, but after filtering we are sure that we have at least one number of other facilities within a 5-mile radius of each selected point.

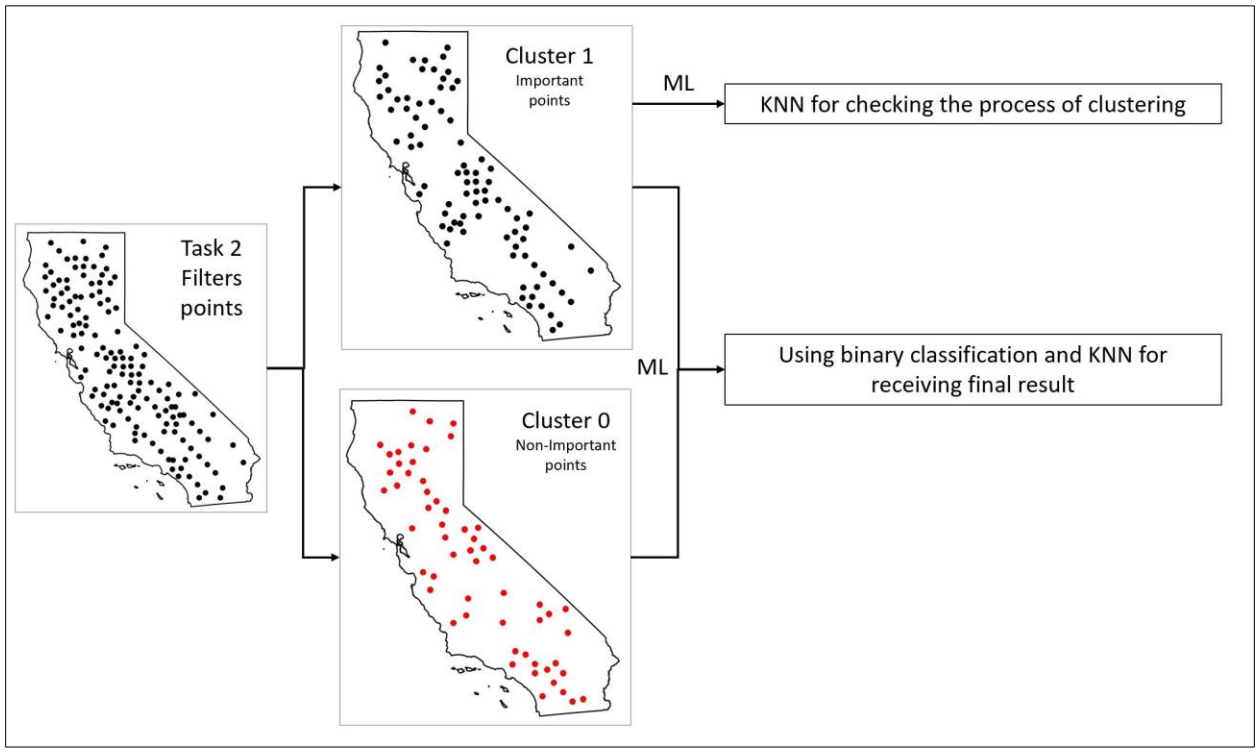


Figure 89: Mechanism of using machine learning algorithms for analyzing the data

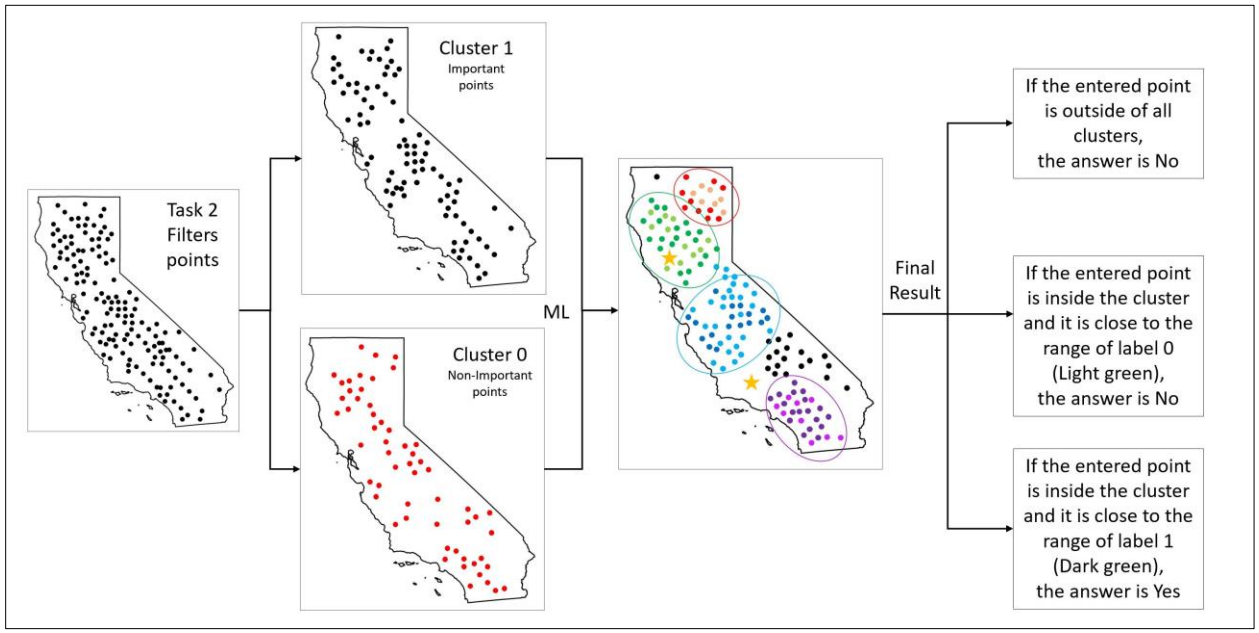


Figure 90: Workflow of using machine learning algorithms for analysis and receiving the result

After knowing about the mechanism of machine learning which have been used in this study, all of the processes of machine learning will be explained in detail in the following sections.

In the machine learning section, we used unsupervised clustering and supervised binary classification[55]. Pycaret library and KNN algorithm were used for both processes and its reasons will be discussed. PyCaret makes performing machine learning tasks as easy as possible by offering a low-code machine learning platform in Python that allows users to prepare the data and deploy the model within minutes in a notebook environment. Python version of the Caret machine learning package in R, with only a few lines of code, allows users and coders to evaluate, compare, and tune models on a dataset[56]. By utilizing the caret package, coders can automate the majority of the steps associated with evaluating and comparing machine learning algorithms in classification and regression. This library avoids the need for lots of manual configuration and some significant results will be achieved with a few lines of the code. So, the PyCaret library provides several advantages and capabilities to Python. Two other advantages of using PyCaret library is that this module can automatically fix the missing data in the dataset, so we do not need to take care of that, and also it checks the overfitting and underfitting with tunning the model. So, using this library we can see the slogan of less is more in the learning machine science. Hence, PyCaret helps architecture to implement machine learning code and receive the results without going to the depth of each part. Accordingly, clustering is an unsupervised machine learning module in PyCaret (pycaret.clustering) that combines objects in such a way that those that are in the same group (called clusters) are more alike as compared to those in other groups[56, 57]. Several preprocessing features are provided by PyCaret's clustering module and can be configured by initializing the setup through the setup() method. It has over 8 algorithms and several plots to display the analyzes and the results. Moreover, as part of its clustering module PyCaret's clustering

module implements an interesting feature known as tune_model(), which allows users to tune the hyperparameters of a clustering model to further enhance supervised learning functions, such as AUC or R2, in contrast, PyCaret's classification module (pycaret.classification) is a supervised machine learning module that uses various algorithms and techniques to classify elements into binary groups. In the case of classification problems, it is common to use them to predict customer default (yes or no), customer churn (whether a customer will leave or stay), and disease status (positive or negative). PyCaret has a classification module that can handle Binary or Multi-class problems. It has over 18 algorithms and 14 plots to show the analysis and performance of all models. PyCaret's classification module offers hyper-parameter tuning, assembling, and more advanced techniques like stacking. The investigation of exploratory data mining methods is a common task used for statistical data analysis in a wide range of fields such as machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression, and computer graphics, while the binary classification method is meant to predict categorical class labels, which are discrete and unordered, such as Pass/Fail, Positive/Negative, Default/Not-Default, for instance. To calculate the clustering model this process will be used[58].

```
Getting Data: How to import data from PyCaret repository
Setting up Environment: How to setup an experiment in PyCaret and get started with building
multiclass models
Create Model: How to create a model and assign cluster labels to the original dataset for
analysis
Plot Model: How to analyze model performance using various plots
Predict Model: How to assign cluster labels to new and unseen datasets based on a trained
model
Save / Load Model: How to save / load model for future use
------------------------------------------------------------------------------
Getting Data: How to import data from PyCaret repository
Setting up Environment: How to setup an experiment in PyCaret & get started with building
classification models
Comparing All Models: Comparing all model for selecting the best one
Create Model: How to create a model, perform stratified cross validation and evaluate
classification metrics
Tune Model: How to automatically tune the hyper-parameters of a classification model
Plot Model: How to analyze model performance using various plots
Finalize Model: How to finalize the best model at the end of the experiment
Predict Model: How to make predictions on new / unseen data
Save / Load Model: How to save / load a model for future use
```

After running task 2, we have three different CSV files. Cluster0.CSV demonstrates the non-important points, cluster1.CSV demonstrates the important points, and All_Clusters.CSV is a combination version of cluster0.CSV and cluster1.CSV. Once we selected the school layer as the origin layer in task 2 and got the three CSV files and after that we selected the plant layer as the origin layer in task 2 and got the three CSV files. Using the cluster1.CSV files which have been obtained from task 2 and demonstrate the important points we did clustering so we can compare their results.

1- School layer as an origin point (Cluster1.CSV) – Unsupervised Clustering

```
dataset
```

	Unnamed: 0	0	1	2	3
0	68	1	school996	38.708417	-120.83077
1	69	1	school9945	38.116982	-122.20194
2	70	1	school9876	38.446471	-121.83635
3	71	1	school9860	38.250114	-122.06617
4	72	1	school9845	41.964173	-121.92438
...	...	...	...	...	...
1062	1130	1	school10060	38.237801	-122.62785
1063	1131	1	school1006	38.728540	-120.79345
1064	1132	1	school1005	38.729000	-120.79240
1065	1133	1	school10048	38.255183	-122.62745
1066	1134	1	school1004	38.735085	-120.77778

1067 rows × 5 columns

Figure 91: Dataset – cluster1.CSV when school layer is as the origin point

```
models()
```

ID	Name	Reference
kmeans	K-Means Clustering	sklearn.cluster._kmeans.KMeans
ap	Affinity Propagation	sklearn.cluster._affinity_propagation.Affinity...
meanshift	Mean Shift Clustering	sklearn.cluster._mean_shift.MeanShift
sc	Spectral Clustering	sklearn.cluster._spectral.SpectralClustering
hclust	Agglomerative Clustering	sklearn.cluster._agglomerative.AgglomerativeCl...
dbscan	Density-Based Spatial Clustering	sklearn.cluster._dbscan.DBSCAN
optics	OPTICS Clustering	sklearn.cluster._optics.OPTICS
birch	Birch Clustering	sklearn.cluster._birch.Birch
kmodes	K-Modes Clustering	kmodes.kmodes.KModes

Figure 92: All possible models we can use for clustering

7. Assign a Model

Now that we have created a model, we would like to assign the cluster labels to our dataset to analyze the results. We will achieve this by using the assign_model() function.

```
kmean_results = assign_model(kmeans)
kmean_results.head()
```

	Unnamed: 0	0	1	2	3	Cluster
0	687	1	school4666	39.400328	-123.35301	Cluster 0
1	315	1	school7882	32.718755	-117.12338	Cluster 3
2	488	1	school6636	38.508144	-121.41886	Cluster 2
3	122	1	school9326	37.153907	-121.67771	Cluster 2
4	749	1	school3927	34.125524	-118.14579	Cluster 1

Figure 93: Assigning the model to the dataset when the school layer is as the origin point

```
plot_model(kmeans)
```

Figure 94: 2D cluster PCA plot when school layer is as the origin point

6 clusters have been used for analysis and creating the clusters, but Pycaret returned the 4 clusters as the optimum number of the cluster which can be applied for the entered dataset. In cluster analysis, Using the elbow method, it is possible to determine how many clusters there are in a dataset. By plotting explained variation as a function of cluster count, we can then determine how many clusters to use based on the elbow of the curve[59].

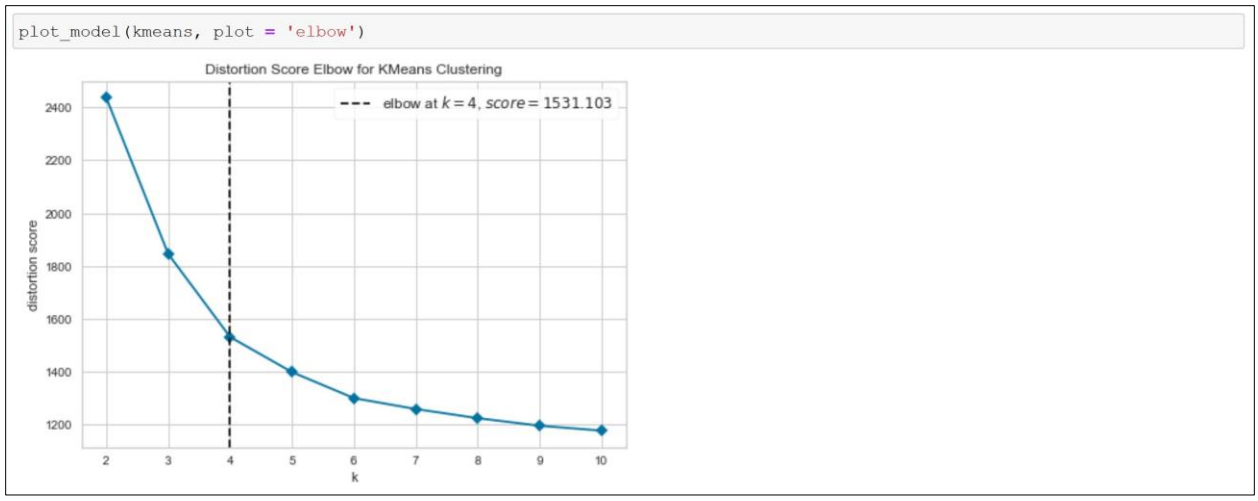


Figure 95: Elbow plot for showing the optimum number of clusters when the school layer is as the origin point

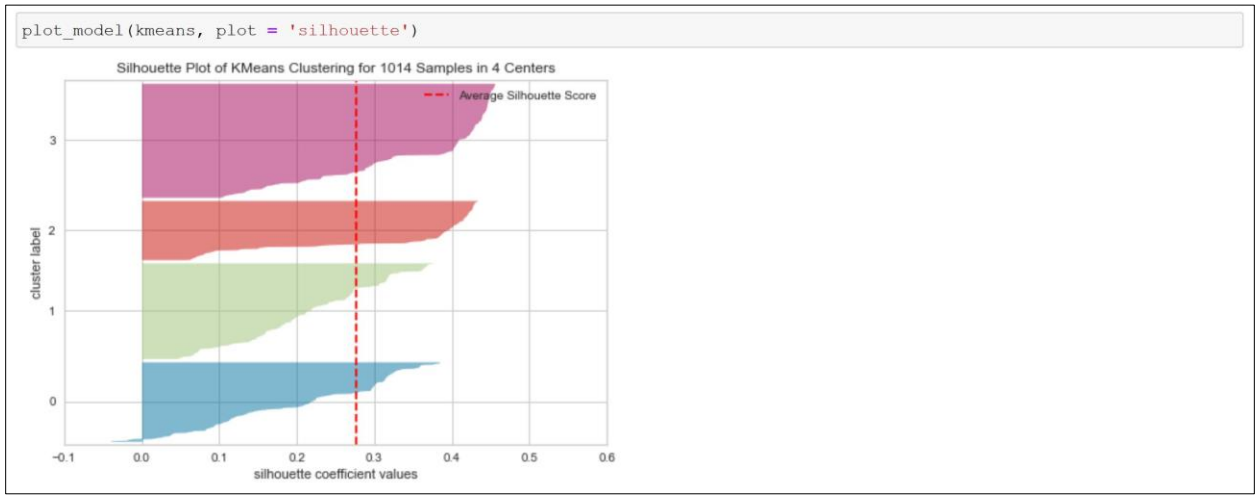


Figure 96: Silhouette plot when school layer is as the origin point

The silhouette plot provides an easy way to assess parameters like cluster number visually by providing a measure of how close one cluster is to its neighboring clusters. As the name suggests, a distribution plot indicates which values are most likely to occur adjacent to each other than values on either side of these neighboring values[60]. So, it shows a group of neighboring values that appear noticeably more frequently in the distribution of a numerical variable compared to values that occur on either side of them[61]. Based on the image 94 and 97 we can see that the amount of cluster 3 is more than other clusters but it has less area in PCA plot. This issue confirms that cluster 3 is denser than other clusters in this dataset.

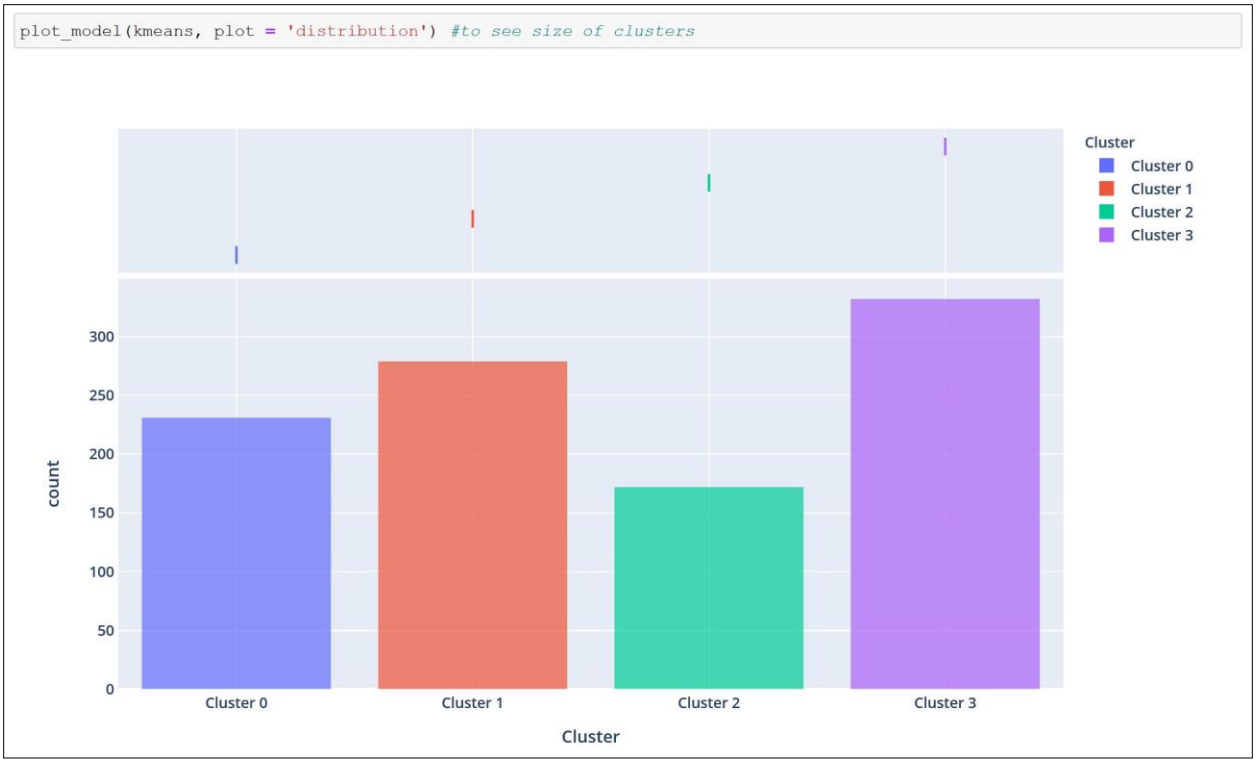


Figure 97: Distribution plot when school layer is as the origin point

Figure 98: Distribution plot when the parameter is 2 (X - Latitude) when school layer is as the origin point

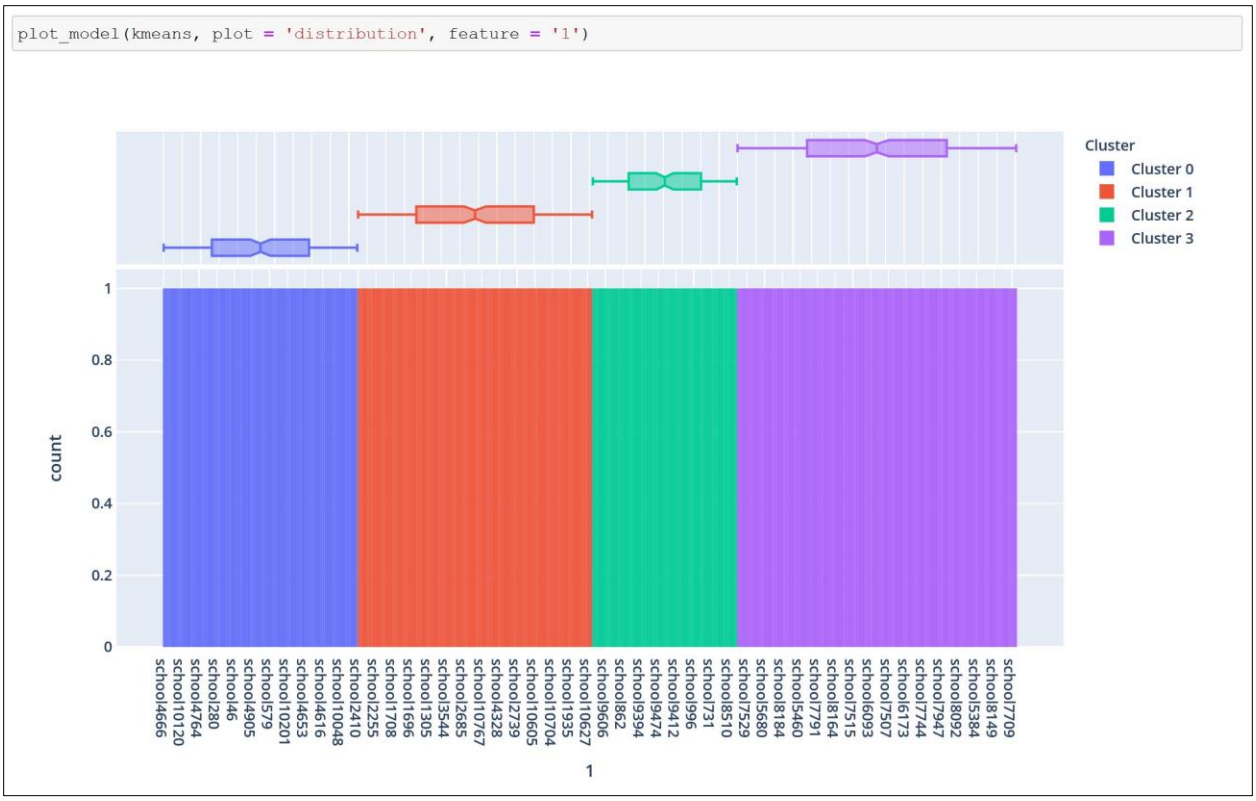


Figure 99: Figure 139: Distribution plot when the parameter is 1 (Y - Longitude) when school layer is as the origin point

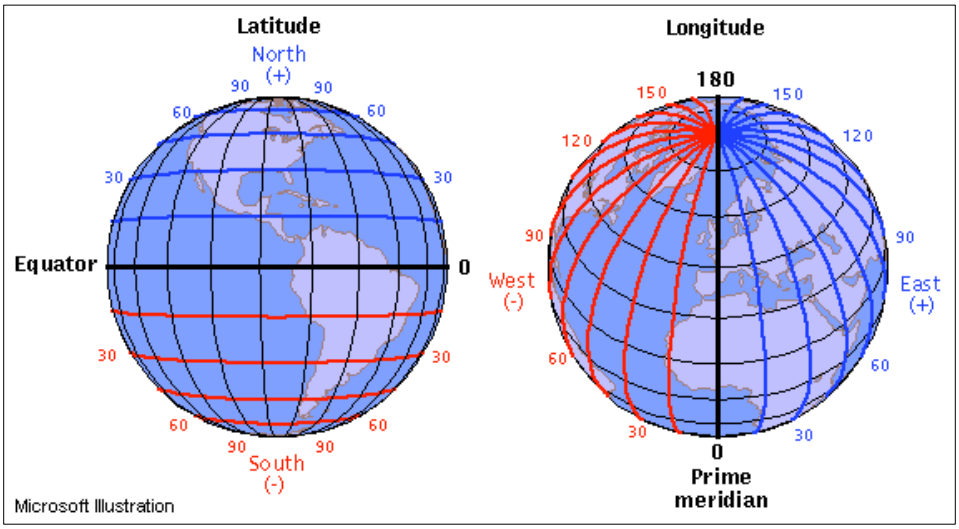


Figure 100: Schematic of Latitude and Longitude

As California state has been expanded from north to south, so it makes sense that the distribution plot with longitude parameters shows any difference but with latitude, parameters show some variations since points have a difference more in their X than their Y.

We selected 65% of data for training and 30% of data for testing and 5% had been selected as unseen data it means that the model never has been seen those data. So, the prediction was done based on unseen data and we can see that some data has been categorized under cluster 2.

9. Predict on unseen data

The predict_model() function is used to assign cluster labels to a new unseen dataset. We will now use our kmeans model to predict the data stored in data_unseen. This variable was created at the beginning of this test and contains 53 samples from the original dataset that were never exposed to PyCaret.

```
unseen_predictions = predict_model(kmeans, data=data_unseen)
unseen_predictions.head()
```

	Unnamed: 0	0	1	2	3	Cluster
0	73	1	school9844	41.964173	-121.92438	Cluster 2
1	116	1	school9384	37.446202	-122.15114	Cluster 2
2	154	1	school8884	37.490108	-122.23423	Cluster 2
3	157	1	school8808	37.694098	-122.48213	Cluster 2
4	165	1	school8629	38.002795	-121.28439	Cluster 2

The Cluster column indicating the cluster label predicted from the trained kmeans model is added onto data_unseen.

Figure 101: Prediction on unseen data when school layer is as the origin point

The same process was done when the plant's layer is at an origin point. We can see its results below and we can compare its results with the condition that the school was as the origin point. All of the conditions are the same in terms of the percentage of training, testing, and unseen data.

2- Plant layer as an origin point (Cluster1.CSV) – Unsupervised Clustering

```
dataset
```

	Unnamed: 0	0	1	2	3
0	41	1	Power_Plant984	34.054685	-117.171317
1	42	1	Power_Plant98	34.271206	-119.170996
2	43	1	Power_Plant975	34.051280	-117.262708
3	44	1	Power_Plant971	34.130222	-117.264433
4	45	1	Power_Plant970	34.177734	-117.309269
...	...	...	...	...	...
232	273	1	Power_Plant1005	34.073793	-117.703496
233	274	1	Power_Plant1004	34.080491	-117.653891
234	275	1	Power_Plant1001	34.106325	-117.658087
235	276	1	Power_Plant1000	34.102404	-117.638256
236	277	1	Power_Plant10	38.445789	-121.462359

237 rows × 5 columns

Figure 102: Dataset – cluster1.CSV when plant layer is as the origin point

```
models()
```

ID	Name	Reference
kmeans	K-Means Clustering	sklearn.cluster._kmeans.KMeans
ap	Affinity Propagation	sklearn.cluster._affinity_propagation.Affinity...
meanshift	Mean Shift Clustering	sklearn.cluster._mean_shift.MeanShift
sc	Spectral Clustering	sklearn.cluster._spectral.SpectralClustering
hclust	Agglomerative Clustering	sklearn.cluster._agglomerative.AgglomerativeCl...
dbscan	Density-Based Spatial Clustering	sklearn.cluster._dbscan.DBSCAN
optics	OPTICS Clustering	sklearn.cluster._optics.OPTICS
birch	Birch Clustering	sklearn.cluster._birch.Birch
kmodes	K-Modes Clustering	kmodes.kmodes.KModes

Figure 103: All possible models we can use for clustering

7. Assign a Model

Now that we have created a model, we would like to assign the cluster labels to our dataset to analyze the results. We will achieve this by using the assign_model() function.

```
kmean_results = assign_model(kmeans)
kmean_results.head()
```

	Unnamed: 0	0	1	2	3	Cluster
0	220	1	Power_Plant2014	36.064854	-119.028892	Cluster 0
1	178	1	Power_Plant231	34.154527	-118.254026	Cluster 0
2	111	1	Power_Plant545	32.912061	-117.104198	Cluster 2
3	183	1	Power_Plant2264	37.528018	-122.262574	Cluster 1
4	95	1	Power_Plant576	32.782404	-117.011621	Cluster 2

Figure 104: Assigning the model to the dataset when plant layer is as the origin point

```
plot_model(kmeans)
```

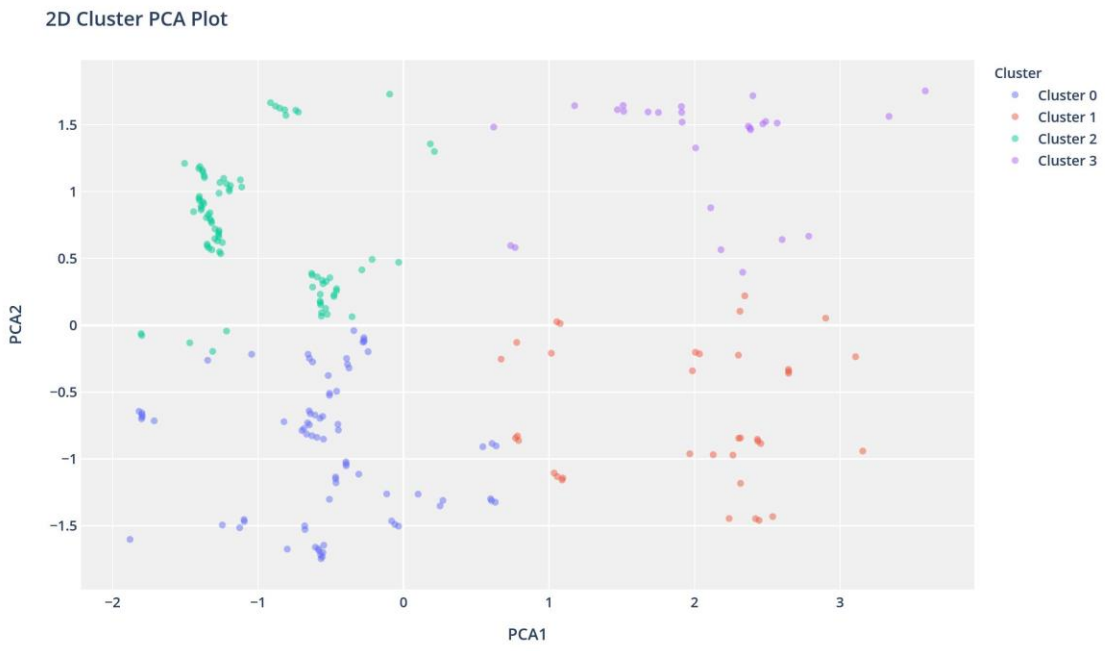


Figure 105: 2D cluster PCA plot when plant layer is as the origin point

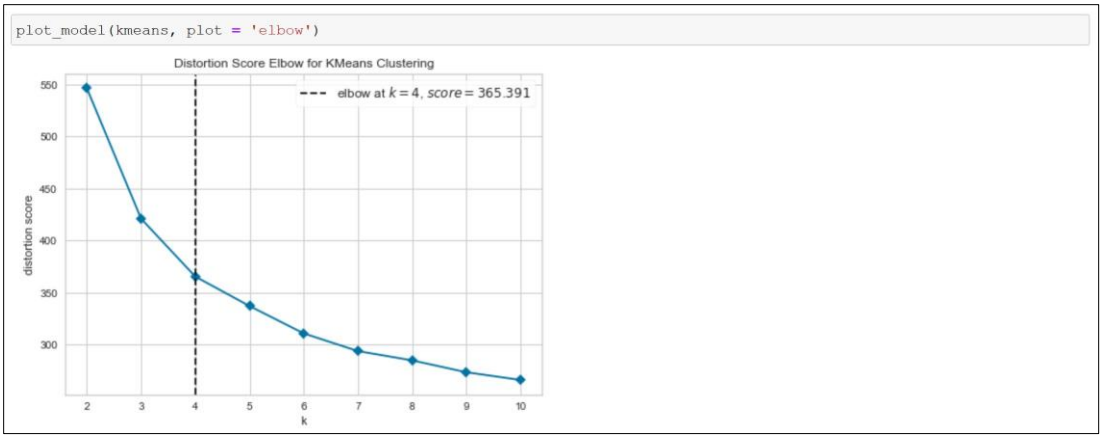


Figure 106: Elbow plot for showing the optimum number of clusters when plant layer is as the origin point

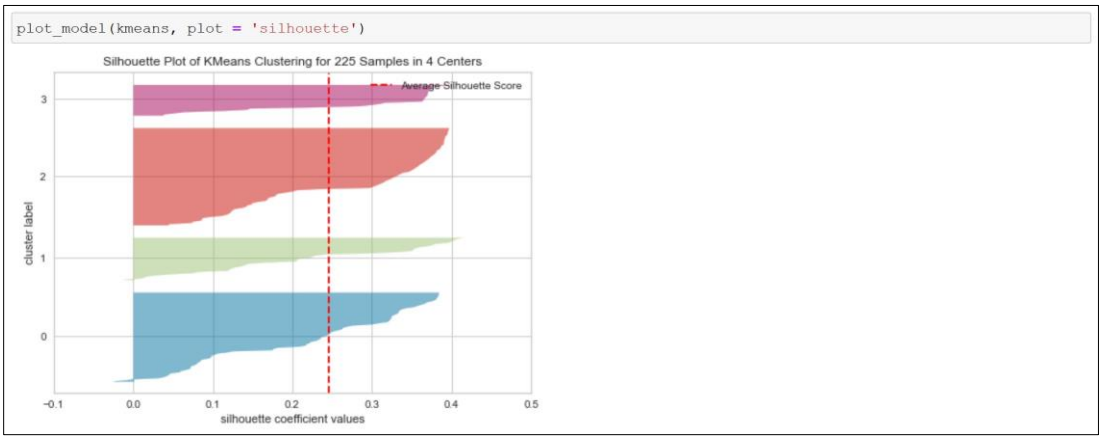


Figure 107: Silhouette plot when school layer is as the origin point

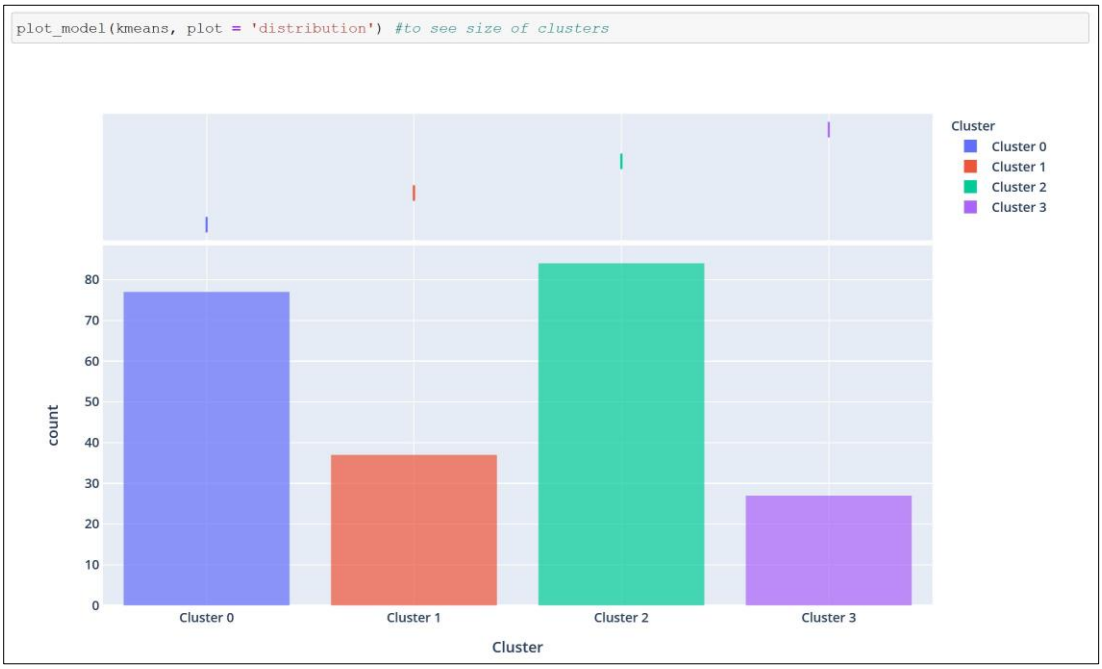


Figure 108: Distribution plot when plant layer is as the origin point

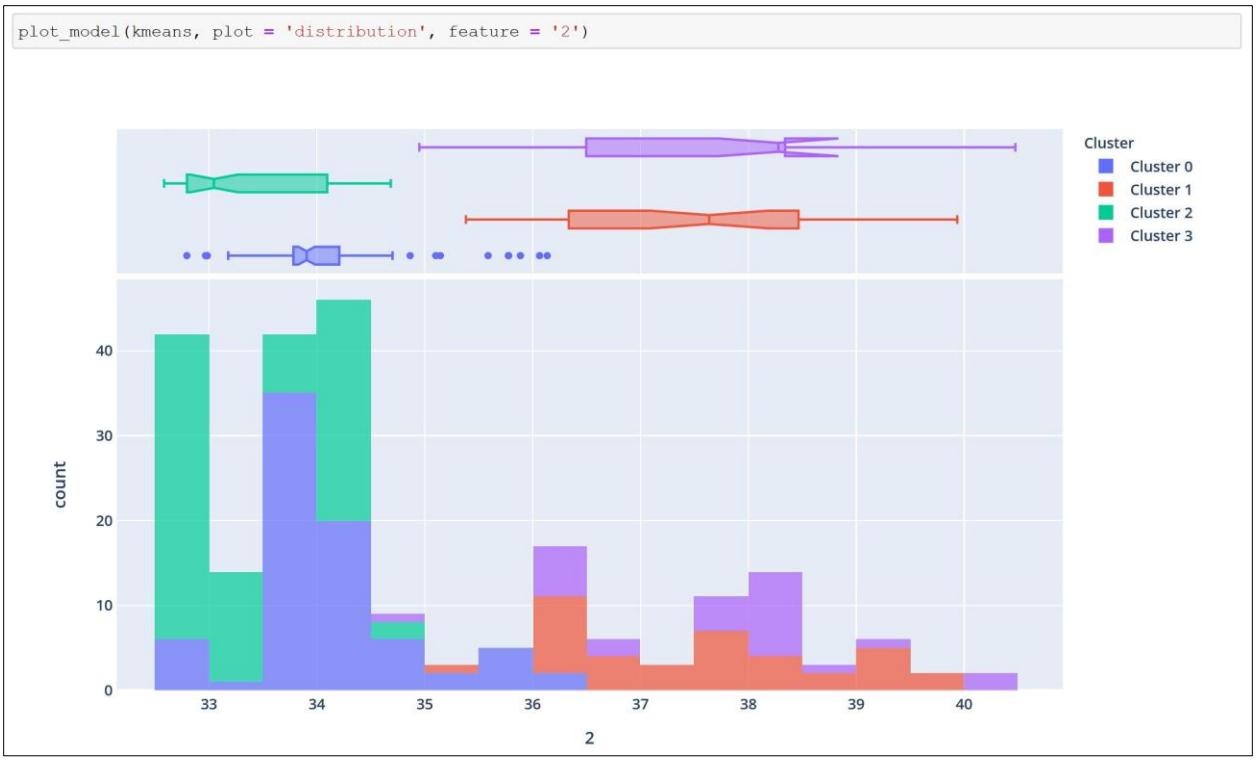


Figure 109: Distribution plot when the parameter is 2 (X - Latitude) when plant layer is as the origin point

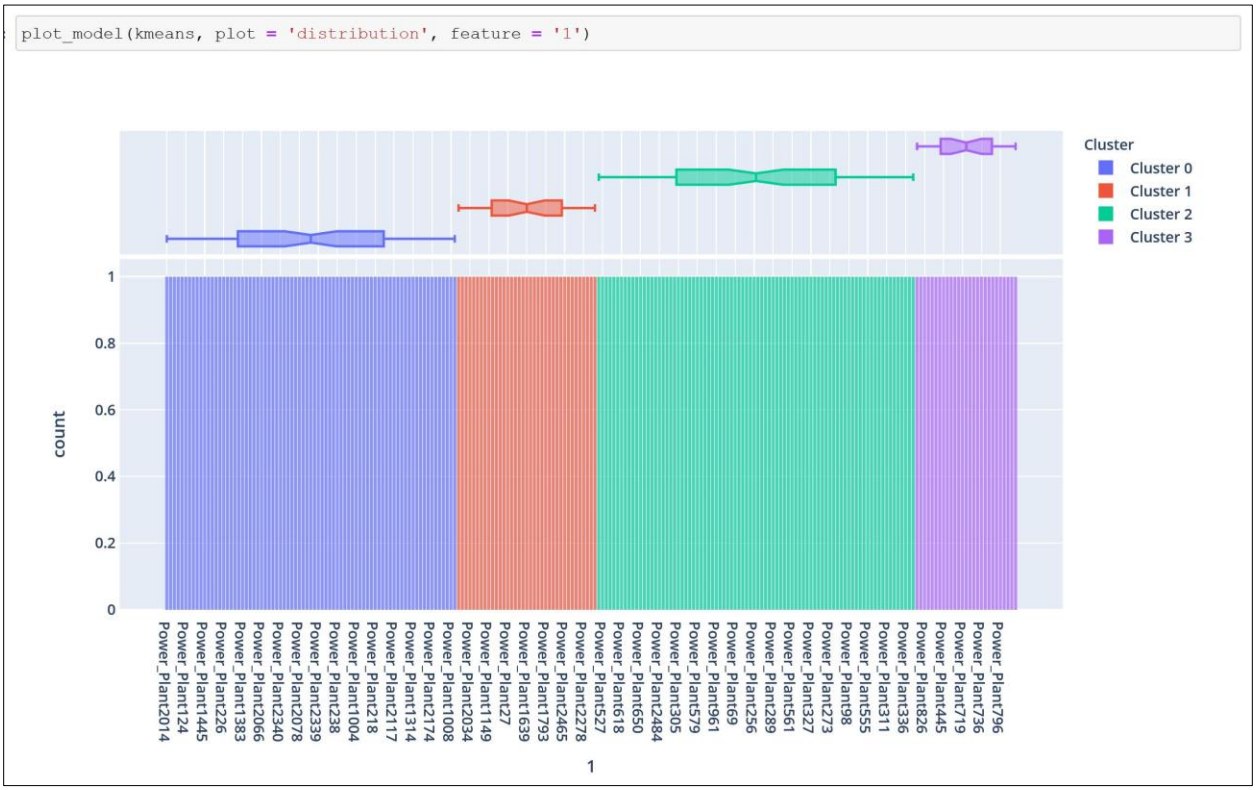


Figure 110: Distribution plot when the parameter is 1 (Y - Longitude) when plant layer is as the origin point

9. Predict on unseen data

The predict_model() function is used to assign cluster labels to a new unseen dataset. We will now use our kmeans model to predict the data stored in data_unseen. This variable was created at the beginning of this test and contains ... samples from the original dataset that were never exposed to PyCaret.

```
unseen_predictions = predict_model(kmeans, data=data_unseen)
unseen_predictions.head()
```

	Unnamed: 0	0	1	2	3	Cluster
0	61	1	Power_Plant79	34.914784	-120.437266	Cluster 3
1	71	1	Power_Plant677	39.291909	-120.214859	Cluster 3
2	105	1	Power_Plant558	32.936999	-117.113723	Cluster 2
3	122	1	Power_Plant486	36.137030	-119.553047	Cluster 3
4	128	1	Power_Plant381	34.573149	-117.935278	Cluster 2

The Cluster column indicating the cluster label predicted from the trained kmeans model is added onto data_unseen.

Figure 111: Prediction on unseen data when plant layer is as the origin point

We selected 65% of data for training and 30% of data for testing and 5% had been selected as unseen data it means that the model never has been seen those data. So, the prediction was done based on unseen data and we can see that some data has been categorized under clusters 2 and 3.

After finishing the test for unsupervised clustering, binary classification was done for getting the final result. In binary classification, we used the KNN algorithm since according to the comparing all the models can be applied for All_Clusters.CSV file, which was the best model due to the evaluating and analysis were done for all possible models. With help of the column 0 that we created in task 2, we can separate the nonimportant point from the important point for training the model. 0 demonstrates the non-important points and 1 demonstrates the important points. Again, similar to the previous parts supervised classification was done for both conditions; first, the school is as an origin point, and second when the plant is as an origin point. At the end, we can compare the results of both conditions. Moreover, for each try 4 models have been compared to select the best model for training and getting the results. Models have been evaluated according to different factors and visualization plots which all can show their performance. In both tries, the KNN algorithm was the best model for training and getting the result.

1- School layer as an origin point (All_Clusters.CSV) – Supervised Binary Classification

dataset

	Unnamed: 0	0	1	2	3
0	1135	0.0	school9999	38.807135	-123.02105
1	1136	0.0	school9998	38.807135	-123.02105
2	1137	0.0	school9997	38.665440	-123.02243
3	1138	0.0	school9996	38.257787	-122.66324
4	1139	0.0	school9995	38.113840	-122.66313
...	...	...	...	...	...
10929	1131	1.0	school1006	38.728540	-120.79345
10930	1132	1.0	school1005	38.729000	-120.79240
10931	1133	1.0	school10048	38.255183	-122.62745
10932	1134	1.0	school1004	38.735085	-120.77778
10933		NaN	NaN	NaN	NaN

10934 rows × 5 columns

Figure 112: Dataset – All_Clusters.CSV when school layer is as the origin point

PyCaret's setup() function initializes the environment and the transformation pipeline, which are needed for modeling and deploying the data. In PyCaret, setup() must be called before any other functions are executed. Two parameters must be provided for initializing the setup():

1- A pandas data frame and 2- The name of the target column (Column 0).

There are no mandatory parameters. We use the remaining options to customize the pre-processing pipeline. In PyCaret, upon executing setup(), the inference algorithm builds the data types of all features automatically based on specific properties. Although this should be done automatically, it isn't always the case. After setup() is executed, PyCaret displays a table containing the features and their inferred data types. Once all of the data types have been identified, enter and quit can be used to continue the experiment. The correct data type is crucial to PyCaret as it performs a few pre-processing tasks automatically, which are crucial for any machine learning experiment. For each type of data, these tasks are performed differently, which means they need to be configured correctly[55].

```
best_model = compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lightgbm	Light Gradient Boosting Machine	0.9044	0.7612	0.1625	0.5745	0.2513	0.2175	0.2685	1.0260
dt	Decision Tree Classifier	0.9043	0.5886	0.1942	0.5597	0.2872	0.2489	0.2899	4.4850
rf	Random Forest Classifier	0.9012	0.6818	0.0357	0.6090	0.0668	0.0562	0.1253	11.5150
gbc	Gradient Boosting Classifier	0.9008	0.6383	0.0124	0.4583	0.0240	0.0207	0.0663	20.2440
et	Extra Trees Classifier	0.9007	0.6305	0.0662	0.5715	0.1149	0.0955	0.1602	34.6020
lr	Logistic Regression	0.9001	0.5533	0.0000	0.0000	0.0000	0.0000	0.0000	3.7890
svm	SVM - Linear Kernel	0.9001	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	4.9170
ridge	Ridge Classifier	0.9001	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	2.7170
ada	Ada Boost Classifier	0.8997	0.6861	0.0124	0.3438	0.0233	0.0181	0.0494	5.4160
knn	K Neighbors Classifier	0.8994	0.6111	0.0096	0.3000	0.0187	0.0137	0.0424	5.5900
lda	Linear Discriminant Analysis	0.6298	0.3500	0.0000	0.0000	0.0000	0.0000	0.0000	110.7250
nb	Naive Bayes	0.0999	0.5000	1.0000	0.0999	0.1816	0.0000	0.0000	0.7080
qda	Quadratic Discriminant Analysis	0.0933	0.4482	0.8917	0.0895	0.1627	-0.0007	-0.0152	101.0800

Figure 113: Comparing all models - All_Clusters.CSV when school layer is as the origin point

Once the setup is complete, it is recommended to begin modeling by comparing all models to evaluate performance (unless you know exactly what kind of model is needed, which is not always the case). A stratified cross-validation approach is used for metric evaluation when all models in the library are trained and scored. The output shows Accuracy, AUC, Recall, Precision, F1, Kappa, and MCC as well as training times across the different folds (10 by default).

Accuracy - is the measurement used to determine which model is best at identifying relationships and patterns between variables in a dataset based on the input, or training, data.

Area Under Curve (AUC) - score represents the degree or measure of separability. A model with higher AUC is better at predicting True Positives and True Negatives. AUC score measures the total area underneath the ROC curve.

Recall - In an imbalanced classification problem with two classes, is calculated as the number of true positives divided by the total number of true positives and false negatives. The result is a value between 0.0 for no recall and 1.0 for full or perfect recall.

Precision - (also called positive predictive value) is the fraction of relevant instances among the retrieved instances, while recall (also known as sensitivity) is the fraction of relevant instances that were retrieved.

F1 Score - is the 2 * ((precision * recall) / (precision + recall)). It is also called the F Score or the F Measure. Put another way, the F1 score conveys the balance between the precision and the recall. The F1 for the All No Recurrence model is 2 * ((0*0)/0+0) or 0.

kappa - statistic is a measure of how closely the instances classified by the machine learning classifier matched the data labeled as ground truth, controlling for the accuracy of a random classifier as measured by the expected accuracy.

Matthews correlation coefficient (MCC) - or phi coefficient is used in machine learning as a measure of the quality of binary (two-class) classifications, introduced by biochemist Brian W.

PyCaret's create_model function is the most detailed and is often the basis for most of its features. So, Most PyCaret functionality is based on the create_model method, which is very granular. As the name suggests cross-validation is used as a means of evaluating and training a model with this function. Therefore, scores plotted by fold include Accuracy, AUC, Recall, Precision, F1, and Kappa. For the remaining part of this analysis, we will work with the below models as our candidate models. This selection is purely for illustration purposes and does not necessarily imply that they are ideal or optimal for our data.

- Decision Tree Classifier ('dt')
- K Neighbors Classifier ('knn')
- Random Forest Classifier ('rf')
- Light Gradient Boosting Machine

Just KNN algorithm was selected to be deployed in plugin for returning the final result. Below, we can see the results of comparing all 4 models and the process for selecting the best model.

```
models()
```

ID	Name	Reference	Turbo
lr	Logistic Regression	sklearn.linear_model._logistic.LogisticRegression	True
knn	K Neighbors Classifier	sklearn.neighbors._classification.KNeighborsCl...	True
nb	Naive Bayes	sklearn.naive_bayes.GaussianNB	True
dt	Decision Tree Classifier	sklearn.tree._classes.DecisionTreeClassifier	True
svm	SVM - Linear Kernel	sklearn.linear_model._stochastic_gradient.SGDC...	True
rbfsvm	SVM - Radial Kernel	sklearn.svm._classes.SVC	False
gpc	Gaussian Process Classifier	sklearn.gaussian_process._gpc.GaussianProcessC...	False
mlp	MLP Classifier	sklearn.neural_network._multilayer_perceptron....	False
ridge	Ridge Classifier	sklearn.linear_model._ridge.RidgeClassifier	True
rf	Random Forest Classifier	sklearn.ensemble._forest.RandomForestClassifier	True
qda	Quadratic Discriminant Analysis	sklearn.discriminant_analysis.QuadraticDiscrim...	True
ada	Ada Boost Classifier	sklearn.ensemble._weight_boosting.AdaBoostClas...	True
gbc	Gradient Boosting Classifier	sklearn.ensemble._gb.GradientBoostingClassifier	True
lda	Linear Discriminant Analysis	sklearn.discriminant_analysis.LinearDiscrimina...	True
et	Extra Trees Classifier	sklearn.ensemble._forest.ExtraTreesClassifier	True
lightgbm	Light Gradient Boosting Machine	lightgbm.sklearn.LGBMClassifier	True

Figure 114: False & True for all models. All_Clusters.CSV when school layer is as the origin point

5.1. Decision Tree Classifier

```
dt = create_model('dt')
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.9106	0.6042	0.2222	0.6400	0.3299	0.2938	0.3418
1	0.9023	0.5935	0.2083	0.5172	0.2970	0.2546	0.2854
2	0.8982	0.5665	0.1528	0.4583	0.2292	0.1890	0.2222
3	0.9010	0.5804	0.1806	0.5000	0.2653	0.2246	0.2585
4	0.8955	0.5464	0.1096	0.4211	0.1739	0.1382	0.1748
5	0.9133	0.6172	0.2466	0.6923	0.3636	0.3282	0.3793
6	0.9037	0.5753	0.1644	0.5714	0.2553	0.2203	0.2703
7	0.9106	0.5974	0.2055	0.6818	0.3158	0.2824	0.3417
8	0.9092	0.6088	0.2329	0.6296	0.3400	0.3022	0.3458
9	0.8982	0.5966	0.2192	0.4848	0.3019	0.2553	0.2789
Mean	0.9043	0.5886	0.1942	0.5597	0.2872	0.2489	0.2899
SD	0.0059	0.0204	0.0399	0.0919	0.0541	0.0544	0.0599

Figure 115: Decision Tree Classifier - All_Clusters.CSV when school layer is as the origin point

5.2. K Neighbors Classifier

```
knn = create_model('knn')
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.9037	0.6124	0.0278	1.0000	0.0541	0.0490	0.1584
1	0.8955	0.5962	0.0000	0.0000	0.0000	-0.0105	-0.0247
2	0.8996	0.5727	0.0139	0.3333	0.0267	0.0189	0.0505
3	0.9010	0.6292	0.0000	0.0000	0.0000	0.0000	0.0000
4	0.8996	0.6382	0.0137	0.5000	0.0267	0.0214	0.0698
5	0.9010	0.6437	0.0274	0.6667	0.0526	0.0451	0.1213
6	0.8996	0.5619	0.0137	0.5000	0.0267	0.0214	0.0698
7	0.8996	0.6253	0.0000	0.0000	0.0000	0.0000	0.0000
8	0.8996	0.6394	0.0000	0.0000	0.0000	0.0000	0.0000
9	0.8955	0.5922	0.0000	0.0000	0.0000	-0.0080	-0.0215
Mean	0.8994	0.6111	0.0096	0.3000	0.0187	0.0137	0.0424
SD	0.0023	0.0276	0.0108	0.3399	0.0208	0.0200	0.0594

Figure 116: K Neighbors Classifier - All_Clusters.CSV when school layer is as the origin point

5.3. Random Forest Classifier

```
rf = create_model('rf')
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.9010	0.7267	0.0417	0.5000	0.0769	0.0626	0.1225
1	0.8941	0.6847	0.0000	0.0000	0.0000	-0.0130	-0.0276
2	0.9023	0.6439	0.0139	1.0000	0.0274	0.0248	0.1119
3	0.9051	0.6679	0.0417	1.0000	0.0800	0.0727	0.1942
4	0.8982	0.6645	0.0137	0.3333	0.0263	0.0185	0.0499
5	0.9037	0.7337	0.0548	0.8000	0.1026	0.0909	0.1937
6	0.9065	0.6962	0.0822	0.8571	0.1500	0.1348	0.2483
7	0.9023	0.6700	0.0548	0.6667	0.1013	0.0873	0.1719
8	0.8982	0.6929	0.0137	0.3333	0.0263	0.0185	0.0499
9	0.9010	0.6378	0.0411	0.6000	0.0769	0.0649	0.1383
Mean	0.9012	0.6818	0.0357	0.6090	0.0668	0.0562	0.1253
SD	0.0035	0.0301	0.0239	0.3069	0.0437	0.0417	0.0786

Figure 117: Random Forest Classifier - All_Clusters.CSV when school layer is as the origin point

5.4. Light Gradient Boosting Machine

```
lightgbm = create_model('lightgbm')
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.9092	0.8048	0.1944	0.6364	0.2979	0.2637	0.3178
1	0.9023	0.7651	0.1806	0.5200	0.2680	0.2287	0.2659
2	0.8982	0.7527	0.0972	0.4375	0.1591	0.1277	0.1700
3	0.8982	0.8162	0.1806	0.4643	0.2600	0.2165	0.2447
4	0.8955	0.7502	0.0685	0.3846	0.1163	0.0886	0.1276
5	0.9161	0.7731	0.2329	0.7727	0.3579	0.3266	0.3952
6	0.8996	0.7327	0.1233	0.5000	0.1978	0.1646	0.2118
7	0.9147	0.7582	0.1781	0.8667	0.2955	0.2705	0.3701
8	0.9133	0.7311	0.2329	0.7083	0.3505	0.3166	0.3737
9	0.8968	0.7276	0.1370	0.4545	0.2105	0.1720	0.2081
Mean	0.9044	0.7612	0.1625	0.5745	0.2513	0.2175	0.2685
SD	0.0077	0.0285	0.0521	0.1537	0.0755	0.0751	0.0877

Figure 118: Light Gradient Boosting Machine - All_Clusters.CSV when school layer is as the origin point

We can compare all factors including accuracy, AUC, recall, precision, F1, kappa, and MCC for all four models. However, this step is not enough for determining which method can be the best method for predicting and receiving the final result. After creating all models and comparing them based on all possible factors, we tuned all four models. The default hyperparameters are used to train a model created with the create_model() function. A tuning function called tune_model() is used to tune hyperparameters. Automatic tuning of a model's hyperparameters is performed through Random Grid Search on a pre-defined search space. For each fold, the best model is scored according to Accuracy, AUC, Recall, Precision, F1, Kappa, and MCC. In the tune_model function, we can pass custom_grid as a parameter (see KNN tuning below). Optimizing accuracy is the default behavior of tune_model, and this can be changed with optimizing parameter. For instance: Tune_model (dt, optimize = 'AUC') allows the user to determine which decision tree classification parameters produced the highest AUC instead of Accuracy[55]. In this test, we have used the Accuracy metric only for simplicity's sake. In general, Accuracy should not be considered when there is an imbalance in the dataset. When we select a model to produce, metrics alone should not be the only criteria. Additionally, standard deviations of K-folds, training time and others should be considered. Again, after tunning all four models we can compare them in terms of all possible factors including Accuracy, AUC, Recall, Precision, F1, Kappa, and MCC, moreover we can compare all of these factors for each model before and after the tunning. All of the results have been provided in the following sections.

6.1. Decision Tree Classifier

```
tuned_dt = tune_model(dt)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.9037	0.6527	0.0833	0.6000	0.1463	0.1252	0.1981
1	0.9037	0.6413	0.1250	0.5625	0.2045	0.1748	0.2327
2	0.8927	0.6200	0.1528	0.3929	0.2200	0.1742	0.1969
3	0.9051	0.6677	0.1528	0.5789	0.2418	0.2090	0.2632
4	0.9023	0.6242	0.1507	0.5500	0.2366	0.2021	0.2516
5	0.9023	0.6620	0.0548	0.6667	0.1013	0.0873	0.1719
6	0.8968	0.6307	0.0822	0.4286	0.1379	0.1091	0.1530
7	0.9065	0.6549	0.1644	0.6316	0.2609	0.2289	0.2895
8	0.9037	0.7007	0.1233	0.6000	0.2045	0.1764	0.2413
9	0.9051	0.6759	0.0685	0.8333	0.1266	0.1131	0.2225
Mean	0.9022	0.6530	0.1158	0.5844	0.1880	0.1600	0.2221
SD	0.0040	0.0238	0.0382	0.1158	0.0526	0.0457	0.0401

Figure 119: Tunning of Decision Tree Classifier - All_Clusters.CSV when school layer is as the origin point

6.2. K Neighbors Classifier ¶

```
tuned_knn = tune_model(knn)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.9010	0.6646	0.0000	0.0000	0.0000	0.0000	0.0000
1	0.9010	0.6819	0.0000	0.0000	0.0000	0.0000	0.0000
2	0.9010	0.6133	0.0000	0.0000	0.0000	0.0000	0.0000
3	0.9010	0.6443	0.0000	0.0000	0.0000	0.0000	0.0000
4	0.8996	0.6283	0.0000	0.0000	0.0000	0.0000	0.0000
5	0.8996	0.6843	0.0000	0.0000	0.0000	0.0000	0.0000
6	0.8996	0.6518	0.0000	0.0000	0.0000	0.0000	0.0000
7	0.8996	0.6567	0.0000	0.0000	0.0000	0.0000	0.0000
8	0.8996	0.6443	0.0000	0.0000	0.0000	0.0000	0.0000
9	0.8996	0.6361	0.0000	0.0000	0.0000	0.0000	0.0000
Mean	0.9001	0.6506	0.0000	0.0000	0.0000	0.0000	0.0000
SD	0.0007	0.0213	0.0000	0.0000	0.0000	0.0000	0.0000

Figure 120: Tunning of K Neighbors Classifier - All_Clusters.CSV when school layer is as the origin point

```
import numpy as np
tuned_knn = tune_model(knn, custom_grid = {'n_neighbors' : np.arange(0,10,1)})
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.9010	0.5774	0.0278	0.5000	0.0526	0.0427	0.0998
1	0.9010	0.5745	0.0000	0.0000	0.0000	0.0000	0.0000
2	0.9010	0.5435	0.0000	0.0000	0.0000	0.0000	0.0000
3	0.9023	0.5976	0.0278	0.6667	0.0533	0.0458	0.1223
4	0.8982	0.5986	0.0000	0.0000	0.0000	-0.0027	-0.0124
5	0.8968	0.6001	0.0137	0.2500	0.0260	0.0157	0.0370
6	0.9010	0.5489	0.0137	1.0000	0.0270	0.0244	0.1111
7	0.9010	0.6192	0.0411	0.6000	0.0769	0.0649	0.1383
8	0.9023	0.5852	0.0274	1.0000	0.0533	0.0482	0.1572
9	0.8996	0.5805	0.0274	0.5000	0.0519	0.0420	0.0989
Mean	0.9004	0.5826	0.0179	0.4517	0.0341	0.0281	0.0752
SD	0.0017	0.0221	0.0138	0.3643	0.0262	0.0227	0.0599

Figure 121: Tunning of K Neighbors Classifier(Using custom grid) - All_Clusters.CSV when school layer is as the origin point

6.3. Random Forest Classifier

```
tuned_rf = tune_model(rf)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.9010	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
1	0.9010	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
2	0.9010	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
3	0.9010	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
4	0.8996	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
5	0.8996	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
6	0.8996	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
7	0.8996	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
8	0.8996	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
9	0.8996	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
Mean	0.9001	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
SD	0.0007	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Figure 122: Tunning of Random Forest Classifier - All_Clusters.CSV when school layer is as the origin point

6.4. Light Gradient Boosting Machine

```
tuned_lightgbm = tune_model(lightgbm)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.9023	0.7498	0.0139	1.0000	0.0274	0.0248	0.1119
1	0.8955	0.7357	0.0278	0.2500	0.0500	0.0308	0.0533
2	0.9051	0.7102	0.0417	1.0000	0.0800	0.0727	0.1942
3	0.9051	0.7294	0.0417	1.0000	0.0800	0.0727	0.1942
4	0.8968	0.6797	0.0274	0.3333	0.0506	0.0359	0.0707
5	0.9037	0.6747	0.0685	0.7143	0.1250	0.1093	0.2014
6	0.8941	0.7035	0.0000	0.0000	0.0000	-0.0105	-0.0249
7	0.9010	0.6921	0.0685	0.5556	0.1220	0.1022	0.1695
8	0.9023	0.6632	0.0685	0.6250	0.1235	0.1057	0.1841
9	0.8982	0.6682	0.0137	0.3333	0.0263	0.0185	0.0499
Mean	0.9004	0.7007	0.0372	0.5812	0.0685	0.0562	0.1204
SD	0.0038	0.0286	0.0238	0.3342	0.0426	0.0398	0.0755

Figure 123: Tunning of Light Gradient Boosting Machine - All_Clusters.CSV when school layer is as the origin point

After tunning all four models for decision-making about the best model, we used different plots to compare all models and selecting the best model for training the data and receiving the final result. So, Plot_model() is useful for analyzing the performance of a model before it is finalized from various perspectives including AUC, confusion_matrix[62], decision boundary, etc. By using this function, a trained model object will be plotted against the test / hold-out set. Each of the four models has been evaluated using different plots, including the AUC plot, the Precision-Recall curve, the feature importance plot, the confusion matrix, and the calibration curve plot.

ROC curves are summarized by the Area Under the Curve (AUC) which indicates the efficiency of a classifier in identifying classes[63]. With higher AUC, the model performs better at distinguishing between the positive and negative. So, positive, and negative classes can be distinguished in the model more effectively with the higher AUC.

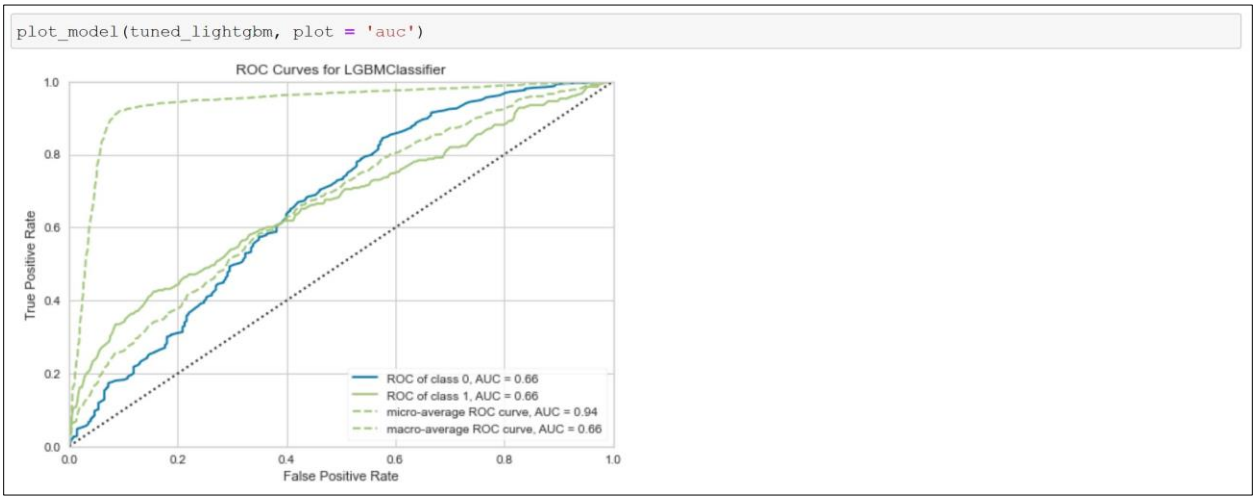


Figure 124: AUC plot for tuned Lightgbm model - All_Clusters.CSV when school layer is as the origin point

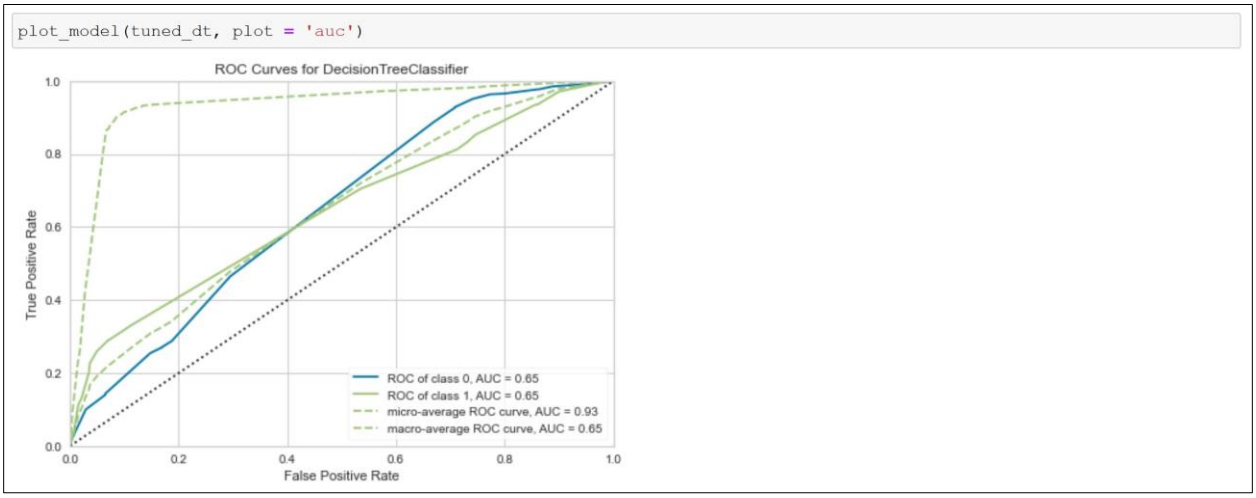


Figure 125: AUC plot for tuned decision tree model - All_Clusters.CSV when school layer is as the origin point

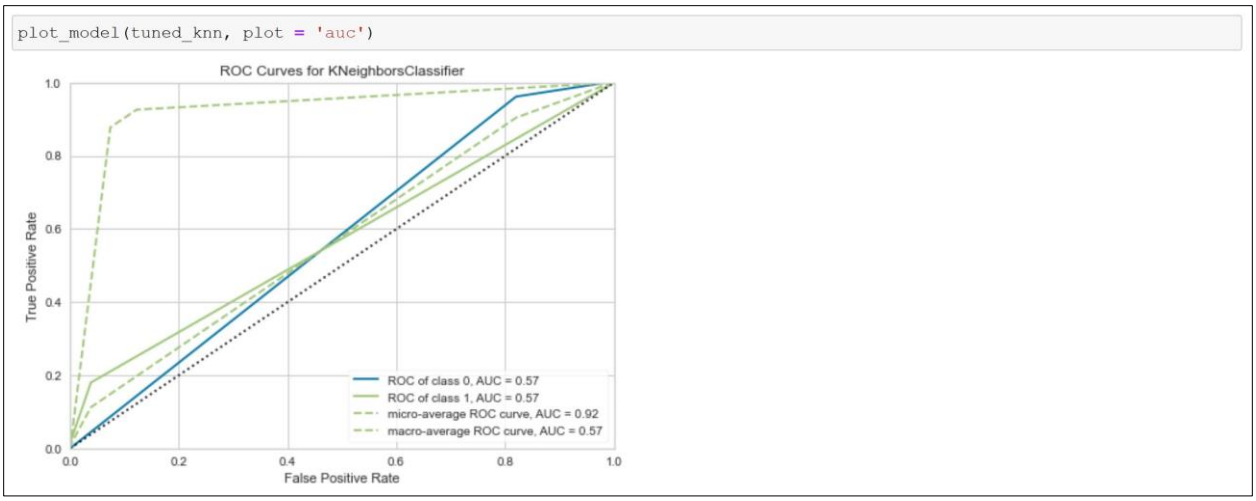


Figure 126: AUC plot for tuned K Neighbors model - All_Clusters.CSV when school layer is as the origin point

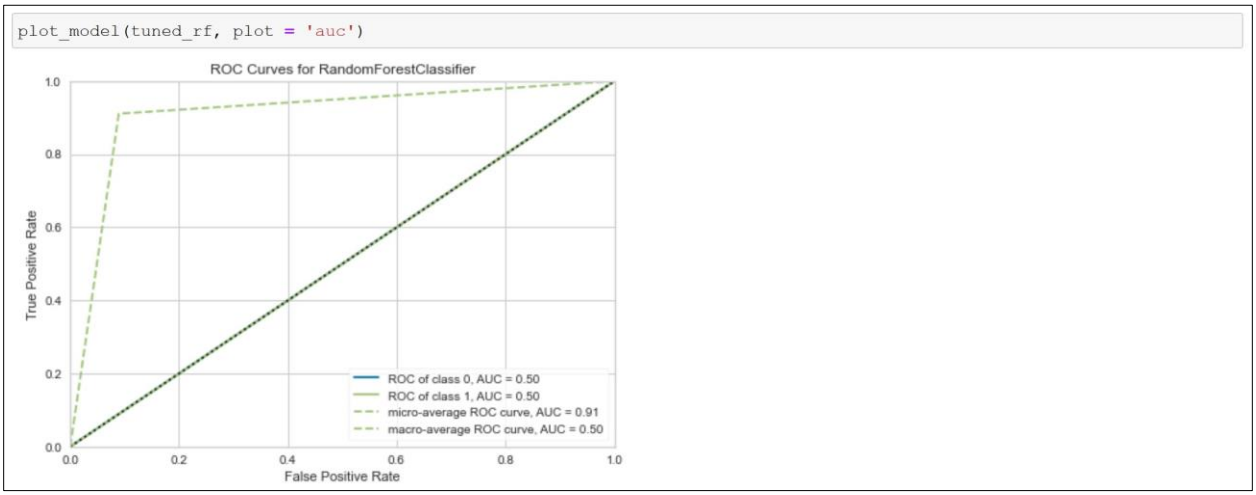


Figure 127: AUC plot for tuned Random Forest model - All_Clusters.CSV when school layer is as the origin point

Investigating and comparing the AUC plot for all four models shows that the tuned K Neighbors model has the best performance.

The next plot we examined for our four models was the precision-recall curve. Like the ROC curve, a precision-recall curve plots the precision (y-axis) and the recall (x-axis) based on a range of thresholds[64]. In no-skill classification, a classifier can't differentiate between the classes in a given set and predicts a random class or a constant class in every case. We can see the results for our four models in the following images.

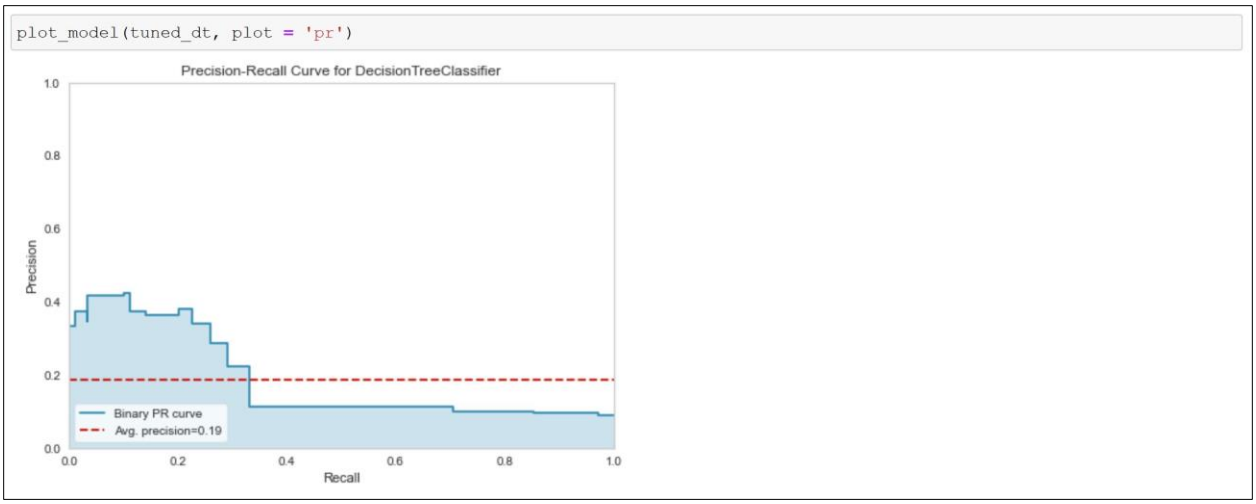


Figure 128: Precision-recall curve plot for tuned decision tree model - All_Clusters.CSV when school layer is as the origin point

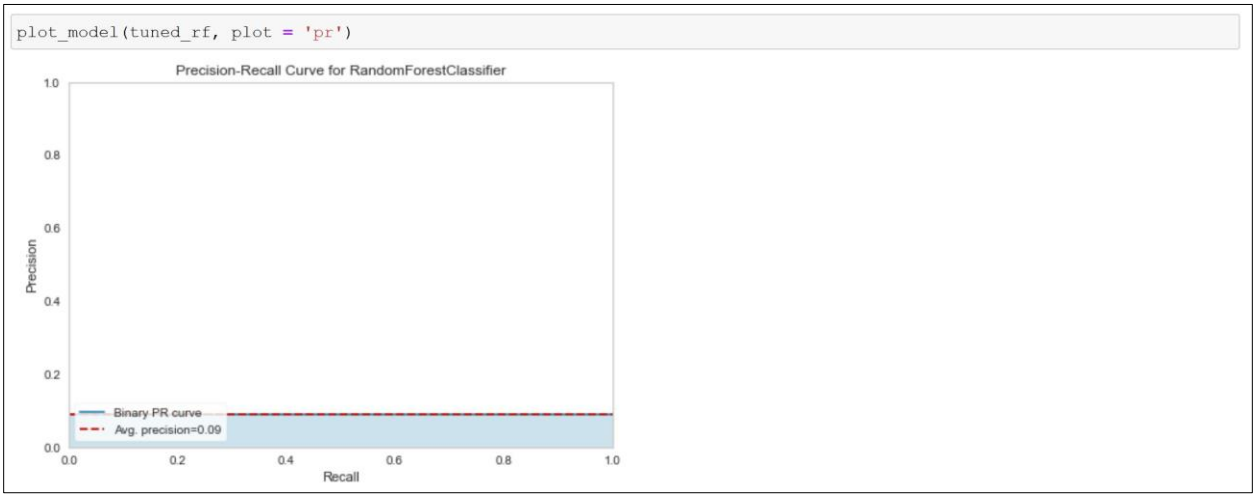


Figure 129: Precision-recall curve plot for tuned Random Forest model - All_Clusters.CSV when school layer is as the origin point

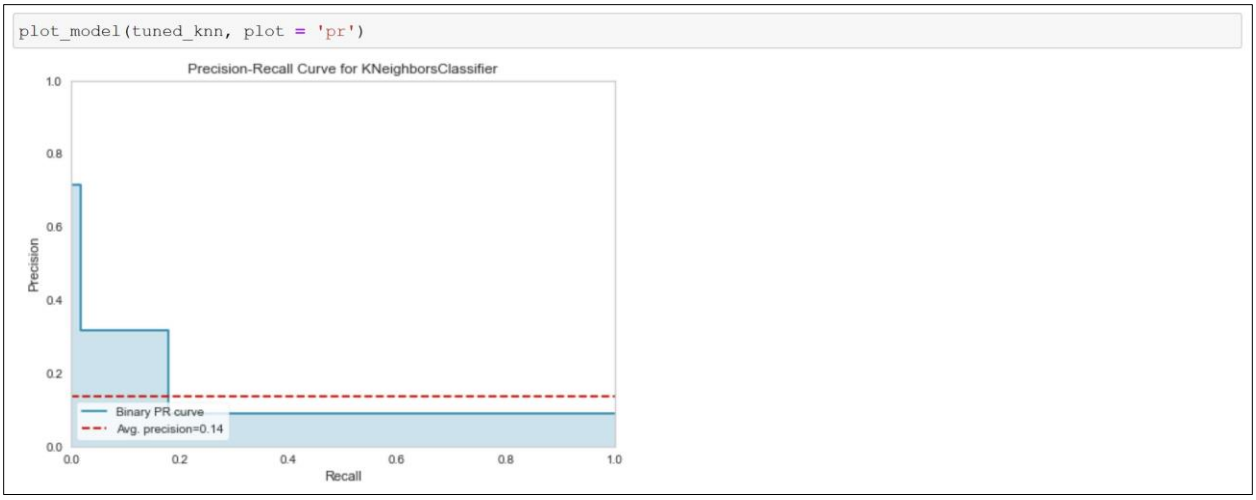


Figure 130: Precision-recall curve plot for tuned K Neighbors model - All_Clusters.CSV when school layer is as the origin point

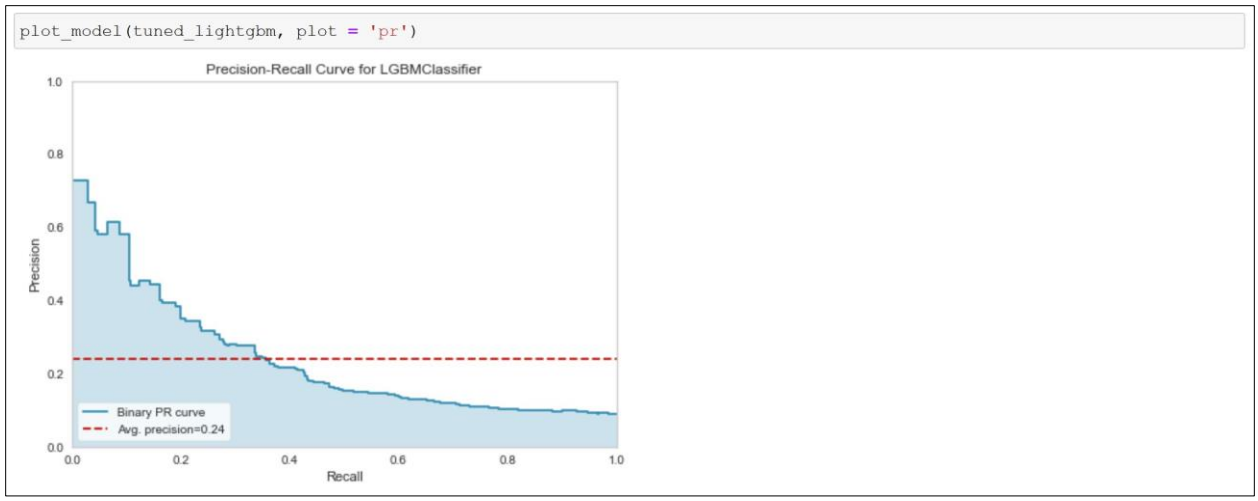


Figure 131: Precision-recall curve plot for tuned Lightgbm model - All_Clusters.CSV when school layer is as the origin point

In this type of the plot, Lightgbm model has the best performance, after that decision tree model and then, K Neighbors model.

There is also a plot called Feature Importance. This plot indicates the weight that each input feature was assigned in a predictive model so that each feature was ranked in accordance with its importance. We can see the results for our four models in the following images.

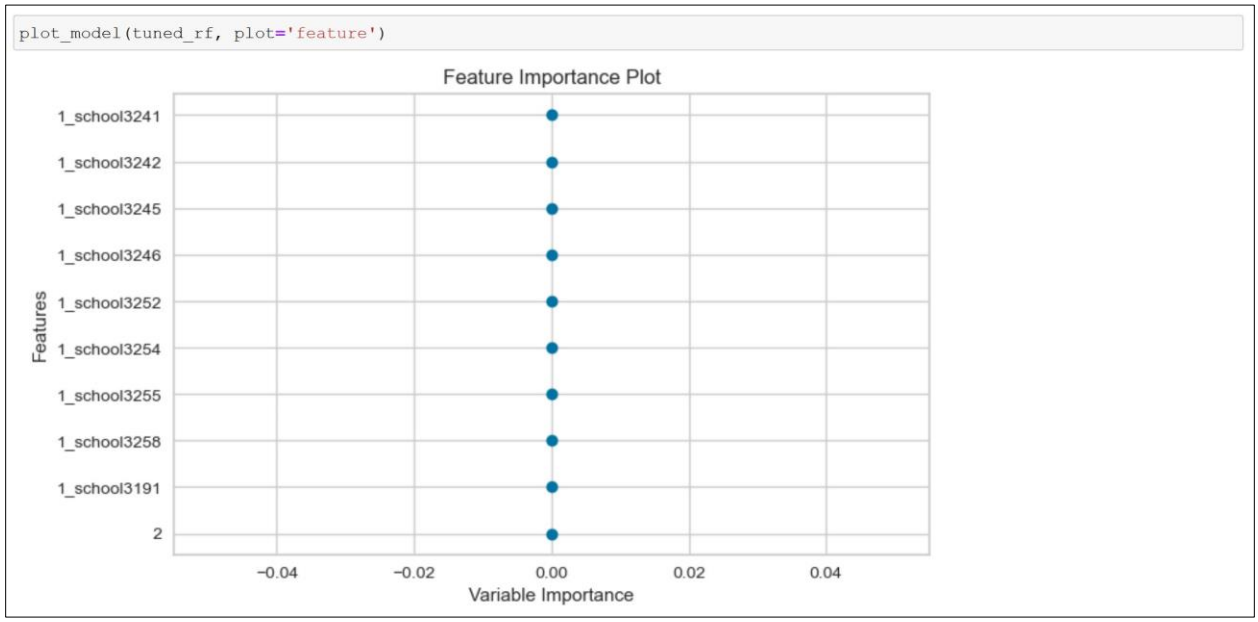


Figure 132: Feature importance plot for tuned Random Forest model - All_Clusters.CSV when school layer is as origin point

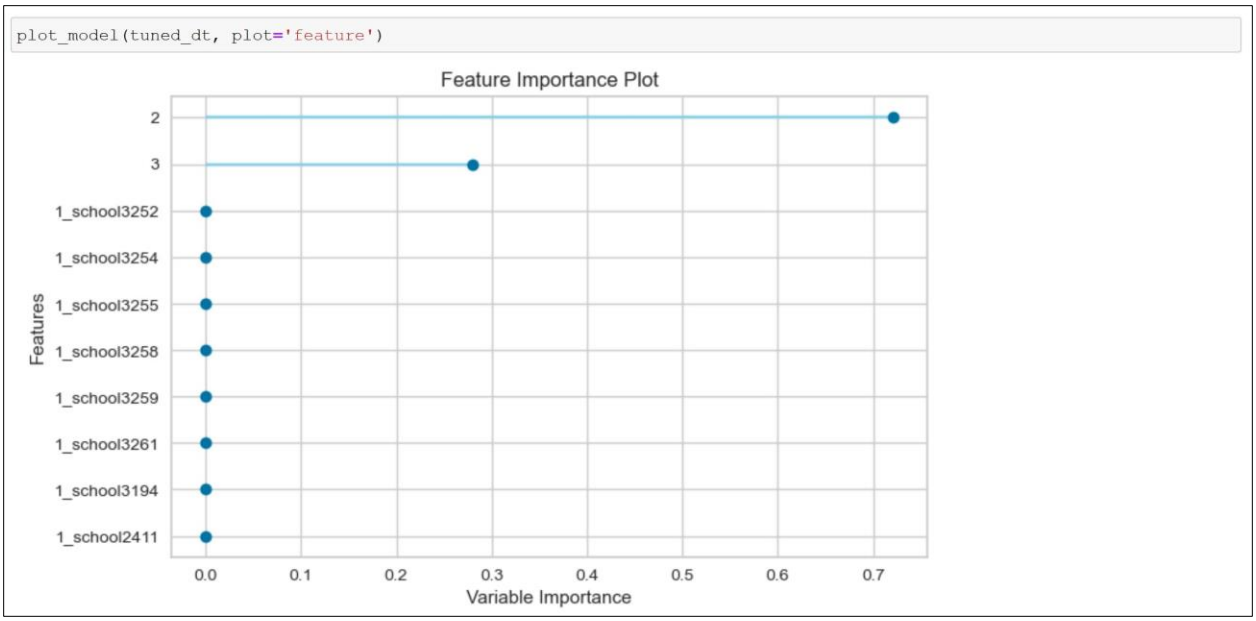


Figure 133: Feature importance plot for tuned decision tree model - All_Clusters.CSV when school layer is as the origin point

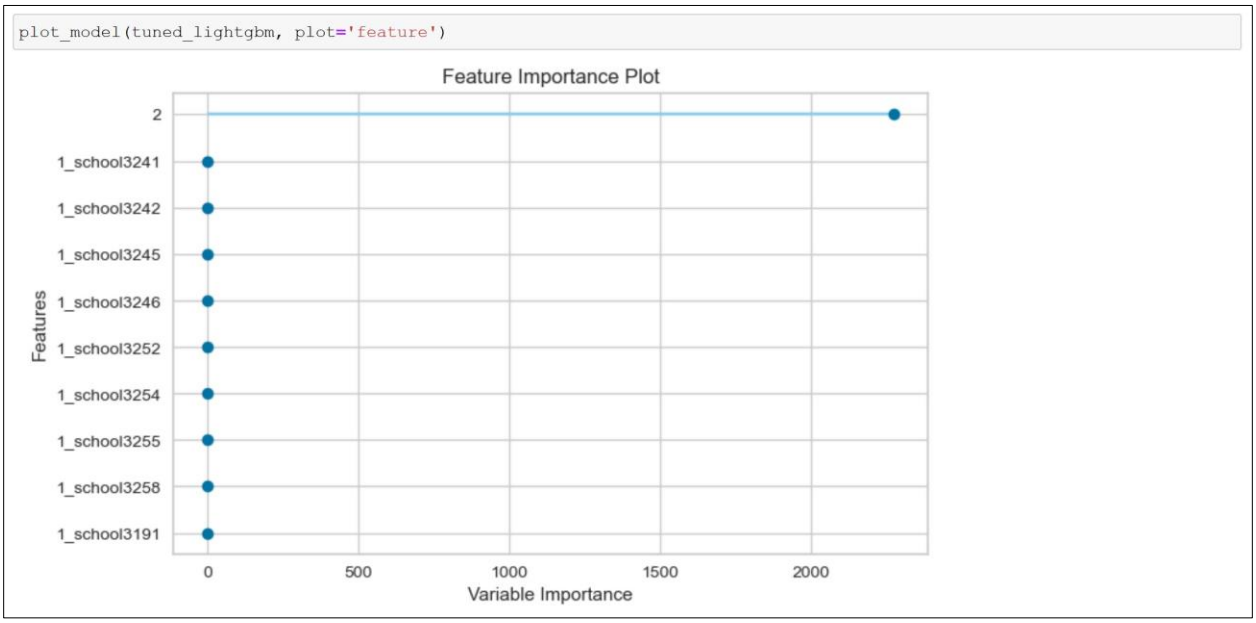


Figure 134: Feature importance plot for tuned Lightgbm model - All_Clusters.CSV when school layer is as the origin point

In image 173, feature 2 which is latitude (X) of points has been more impact than feature 3 which is longitude (Y). In Figure 171, all features are of equal importance. And we should mention that the feature importance plot is not available for K neighbor's model. So, this type of plot decision tree model has the best performance since it can show the importance of all parameters which are in the dataset and are involved in the analysis.

In the context of classification models' performance, a confusion matrix is an N x N matrix used for assessing the accuracy of the model when there are multiple classes being considered. In this regard, comparing the predicted values to the actual target value is done using the matrix. According to the variable predictions, the rows represent the predicted values.

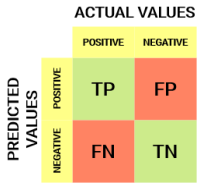


Figure 135: Confusion matrix plot

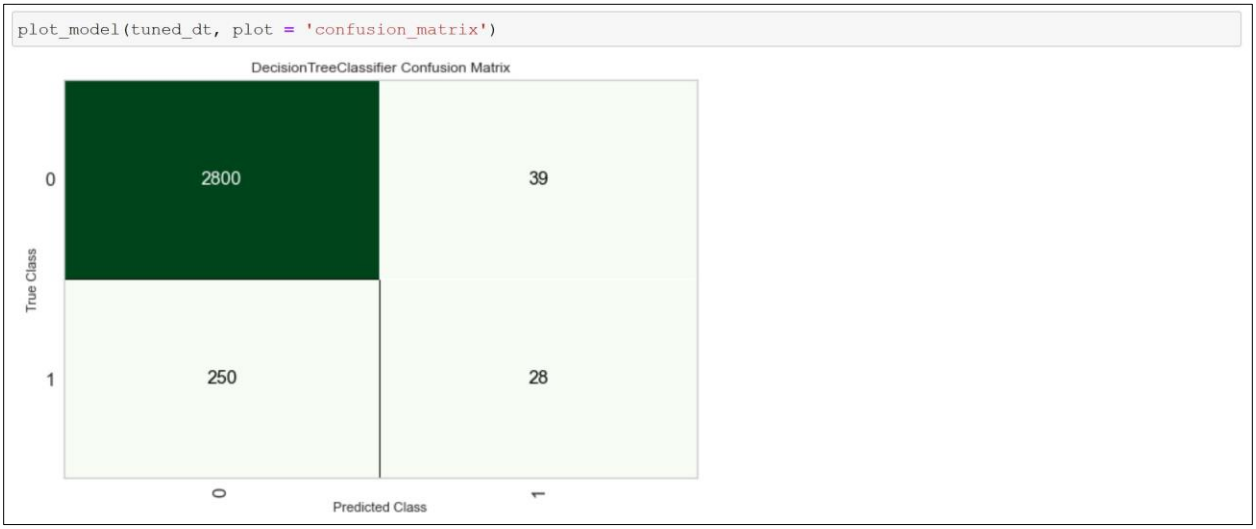


Figure 136: Confusion matrix plot for tuned decision tree model - All_Clusters.CSV when school layer is as the origin point

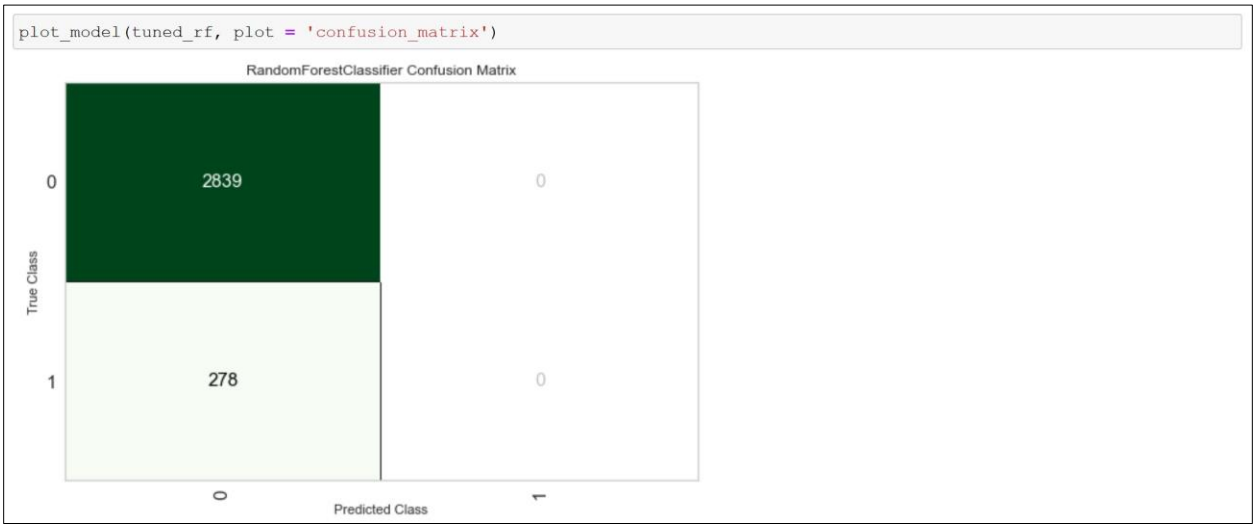


Figure 137: Confusion matrix plot for tuned Random Forest model - All_Clusters.CSV when school layer is as the origin point

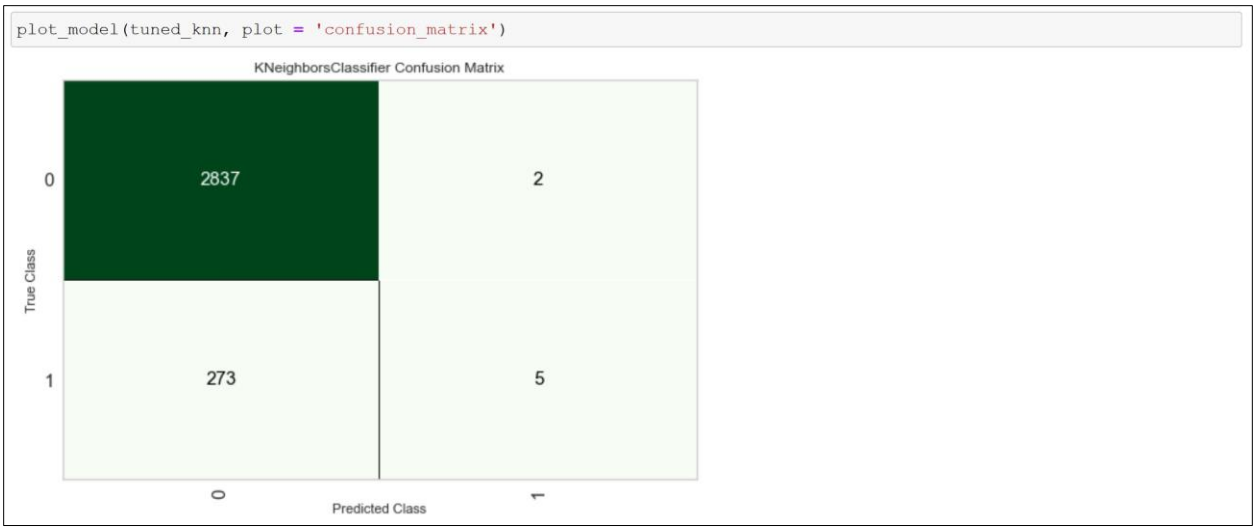


Figure 138: Confusion matrix plot for tuned K Neighbors model - All_Clusters.CSV when school layer is as the origin point

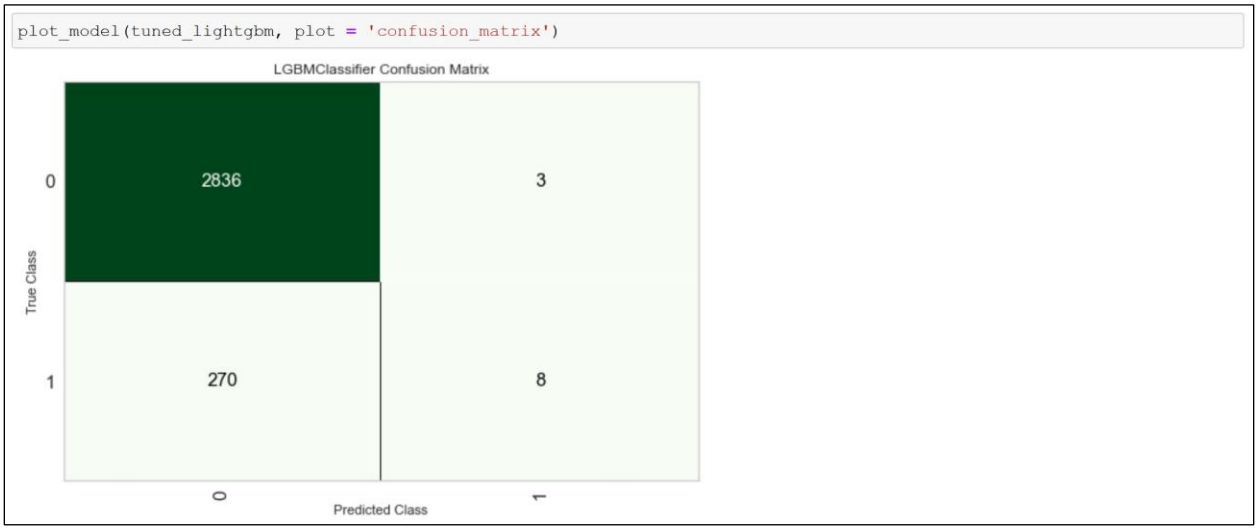


Figure 139: Confusion matrix plot for tuned Lightgbm model - All_Clusters.CSV when school layer is as the origin point

According to the results we can see that in the confusion matrix plot, the random forest has the best performance, after that K Neighbors model is in the second place, then the K Neighbors model, then the Lightgbm model, and finally the decision tree model.

The next plot is the calibration curve. Classifier calibration curves determine how well a classifier is calibrated, i.e., how much each class label is predicted by the classifier. While a plot of the average expected probability in each bin is shown along the x-axis, the y-axis shows, the ratio of the positive (the proportion of positive predictions).

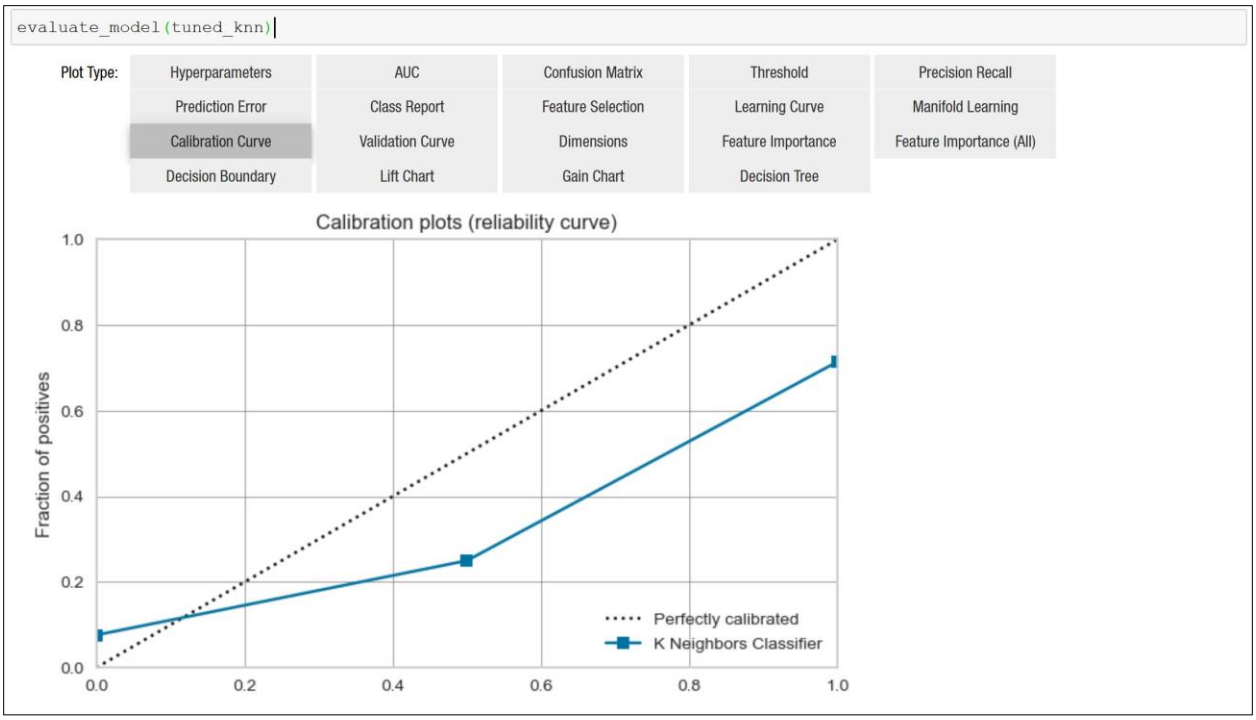


Figure 140: Calibration curves plot for tuned K Neighbors model - All_Clusters.CSV when school layer is as the origin point

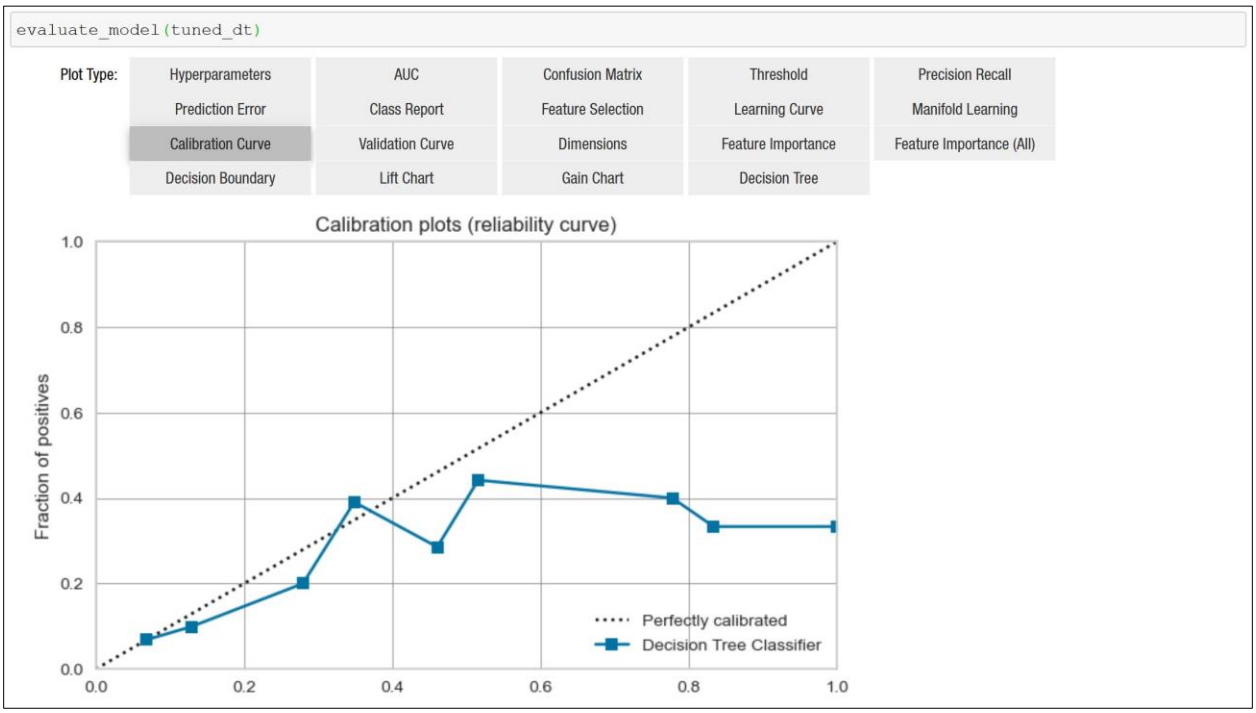


Figure 141: Calibration curves plot for tuned decision tree model - All_Clusters.CSV when school layer is as the origin point

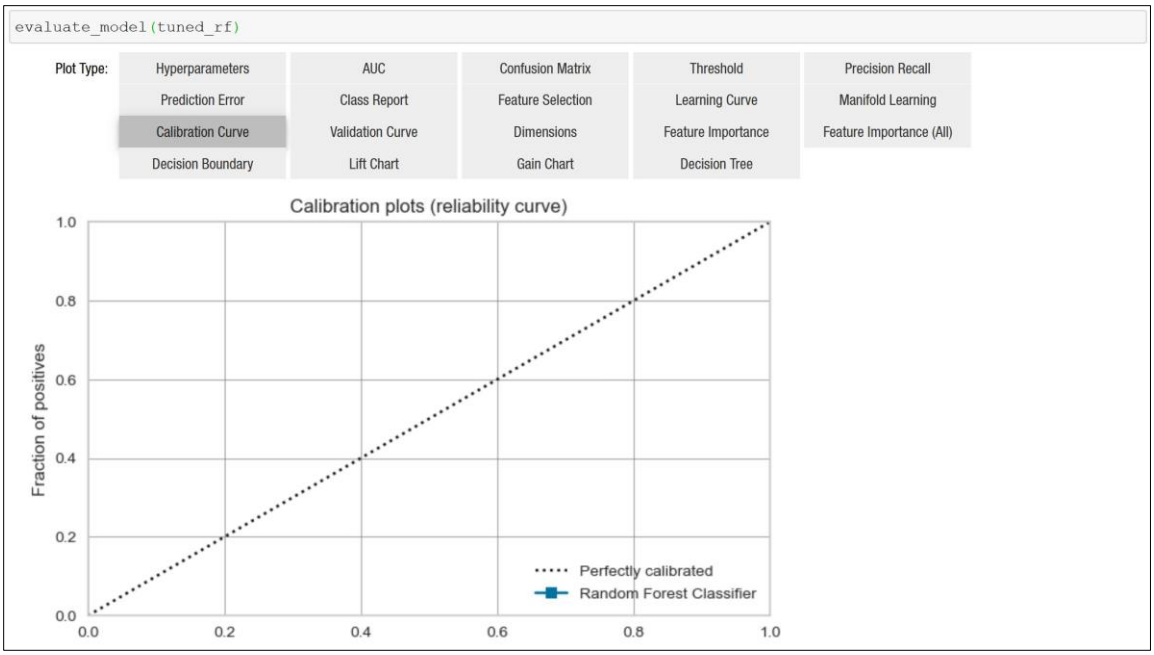


Figure 142: Calibration curves plot for tuned Random Forest model - All_Clusters.CSV when school layer is as the origin point

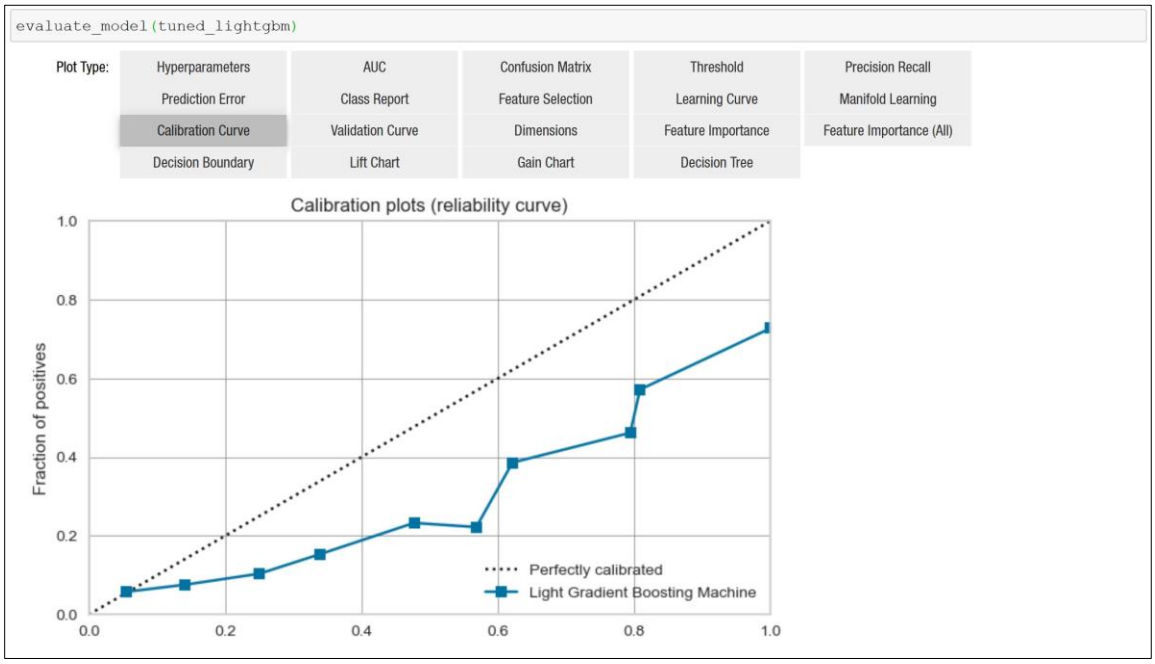


Figure 143: Calibration curves plot for tuned Lightgbm model - All_Clusters.CSV when school layer is as the origin point

According to the results we can see that in the calibration curves plot, the K Neighbors model has the best performance, then the Lightgbm model is in the second place, and finally the decision tree model in the third place.

The next plot is the validation curve. Validation Curve is an important diagnostic tool that displays the relationship between changes in the accuracy of a Machine Learning model with changes in its parameters. Validation curves typically draw a connection between a model parameter and a model score.

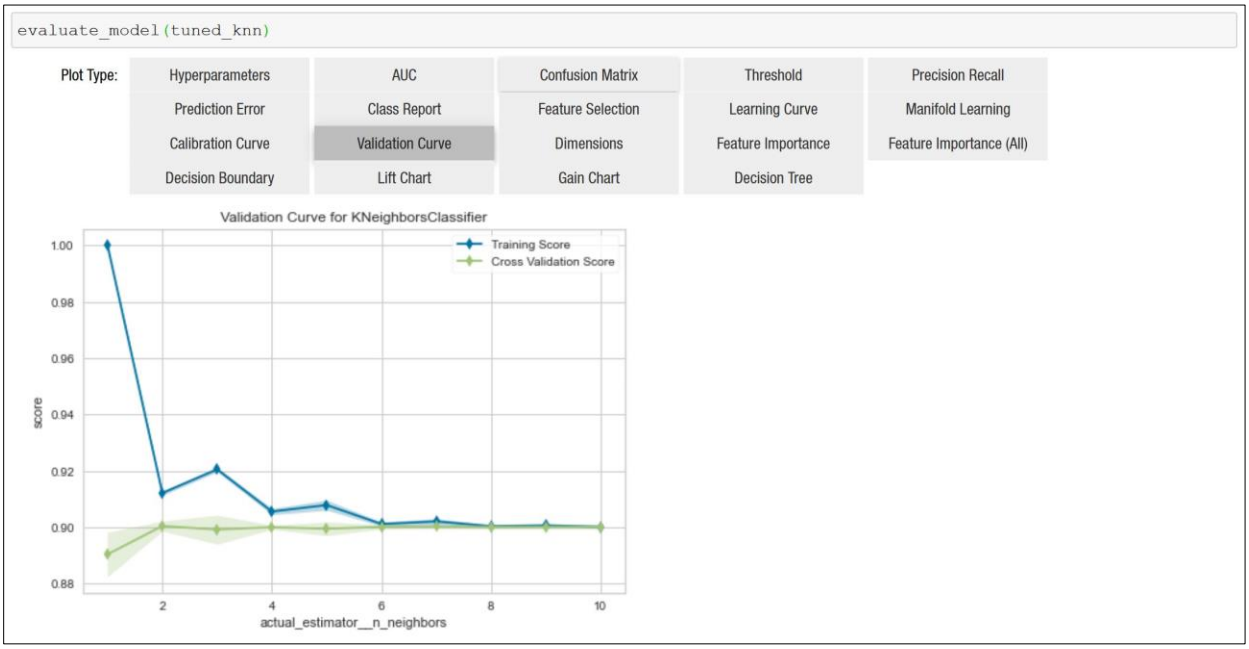


Figure 144: Validation curve plot for tuned K Neighbors model - All_Clusters.CSV when school layer is as the origin point

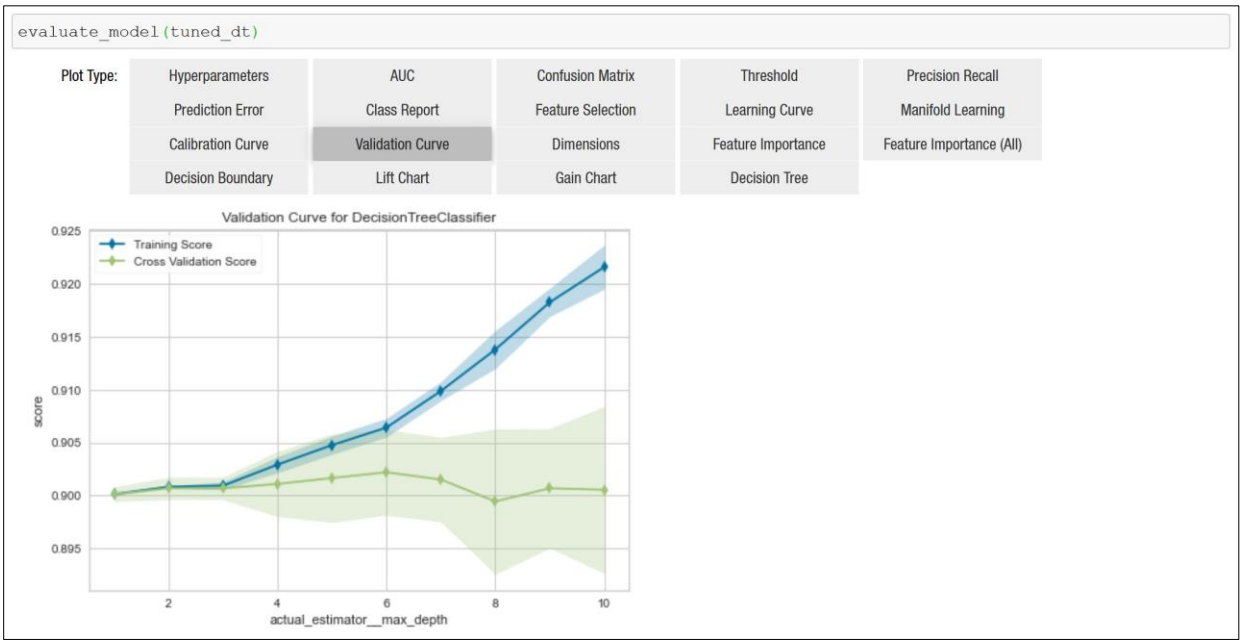


Figure 145: Validation curve plot for decision tree model - All_Clusters.CSV when school layer is as the origin point

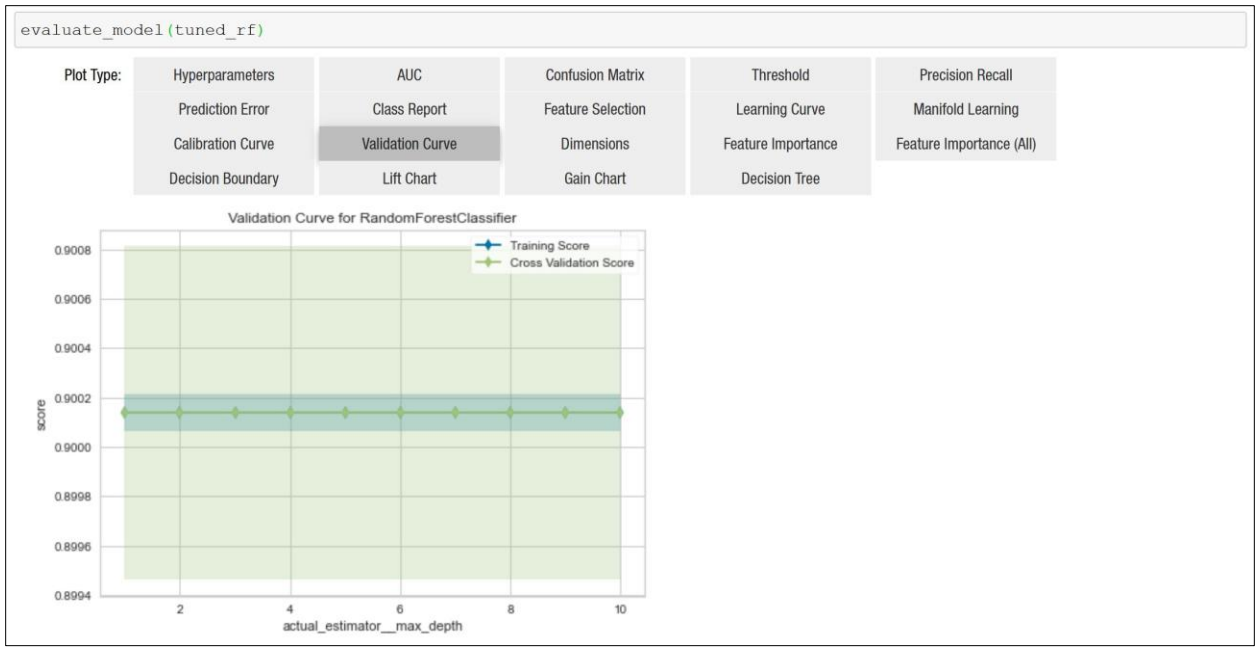


Figure 146: Validation curve plot for Random Forest model - All_Clusters.CSV when school layer is as the origin point

Figure 147: Validation curve plot for Lightgbm model - All_Clusters.CSV when school layer is as the origin point

According to the results we can see that in the calibration curves plot, the K Neighbors model has the best performance, and the Lightgbm model has been placed in the second level.

After checking all of these types of the plot, in average we made sure that KNN can be one of the good options for deploying it in plugin's code to do prediction. So, we finalize our model with the KNN algorithm and deployed it to the python code of the plugin for recalling.

10. Predict on unseen data

The predict_model() function is also used to predict on the unseen dataset. The only difference from section 9 above is that this time we will pass the data_unseen parameter. data_unseen is the variable created at the beginning of the tutorial and contains (547 samples) of the original dataset which was never exposed to PyCaret. (see section 5 for explanation).

```
unseen_predictions = predict_model(final_knn, data=data_unseen)
unseen_predictions.head()
```

	Unnamed: 0	0	1	2	3	Label	Score
0	1140	0.0	school9994	38.434145	-122.66160	0.0	0.5
1	1153	0.0	school9982	38.482107	-122.89015	0.0	1.0
2	1172	0.0	school9965	38.147148	-122.24586	0.0	1.0
3	1174	0.0	school9963	38.097903	-122.23270	0.0	1.0
4	1192	0.0	school9946	38.150639	-122.24559	0.0	1.0

Figure 148: Prediction on unseen data - All_Clusters.CSV when school layer is as the origin point

The prediction section returns the label 0 and 1 as the final answer to our question based on that whether a place is suitable for the construction of a factory or not. The result of the 0 demonstrates that a place is not suitable for establishing a new factory, and label 1 demonstrates that a place is suitable for establishing a new factory. Also, we can see that the score of most of the answers is 1 which shows the highest accuracy of prediction. We did the same process again for when the factories areas the origin points. In the following sections all the relative plots have been provided but as the whole of the process is exactly similar to when schools are as the origin point, we do not explain them, just as a comparison we can compare their results.

2- Plant layer as an origin point (All_Clusters.CSV) – Supervised Binary Classification

```
dataset
```

	Unnamed: 0	0	1	2	3
0	278	0.0	Power_Plant999	34.073986	-117.077319
1	279	0.0	Power_Plant998	34.076206	-117.229494
2	280	0.0	Power_Plant997	34.082660	-117.228091
3	281	0.0	Power_Plant996	34.080008	-117.213449
4	282	0.0	Power_Plant995	34.085712	-117.234226
...	...	...	...	...	...
2451	274	1.0	Power_Plant1004	34.080491	-117.653891
2452	275	1.0	Power_Plant1001	34.106325	-117.658087
2453	276	1.0	Power_Plant1000	34.102404	-117.638256
2454	277	1.0	Power_Plant10	38.445789	-121.462359
2455		NaN	NaN	NaN	NaN

2456 rows × 5 columns

Figure 149: Dataset – All_Clusters.CSV when plant layer is as the origin point

```
best_model = compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
ada	Ada Boost Classifier	0.9099	0.7111	0.0842	0.5167	0.1424	0.1252	0.1850	0.3200
gbc	Gradient Boosting Classifier	0.9069	0.6769	0.0062	0.1000	0.0118	0.0107	0.0238	1.0520
lr	Logistic Regression	0.9063	0.6750	0.0000	0.0000	0.0000	0.0000	0.0000	0.8780
knn	K Neighbors Classifier	0.9063	0.5105	0.0000	0.0000	0.0000	0.0000	0.0000	0.2140
ridge	Ridge Classifier	0.9063	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0950
rf	Random Forest Classifier	0.9063	0.7145	0.0000	0.0000	0.0000	0.0000	0.0000	0.4670
lda	Linear Discriminant Analysis	0.9063	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	1.4050
et	Extra Trees Classifier	0.9026	0.5002	0.0067	0.1000	0.0125	0.0035	0.0084	1.2830
dt	Decision Tree Classifier	0.9014	0.5643	0.1496	0.4283	0.2107	0.1733	0.2040	0.1370
lightgbm	Light Gradient Boosting Machine	0.8965	0.7399	0.2088	0.3944	0.2691	0.2201	0.2345	0.1990
svm	SVM - Linear Kernel	0.7112	0.0000	0.2533	0.0290	0.0520	0.0030	0.0087	0.1780
nb	Naive Bayes	0.0937	0.5000	1.0000	0.0937	0.1714	0.0000	0.0000	0.0690
qda	Quadratic Discriminant Analysis	0.0937	0.5000	1.0000	0.0937	0.1714	0.0000	0.0000	1.2270

Figure 150: Comparing all models - All_Clusters.CSV when plant layer is as the origin point

```
models()
```

ID	Name	Reference	Turbo
lr	Logistic Regression	sklearn.linear_model._logistic.LogisticRegression	True
knn	K Neighbors Classifier	sklearn.neighbors._classification.KNeighborsCl...	True
nb	Naive Bayes	sklearn.naive_bayes.GaussianNB	True
dt	Decision Tree Classifier	sklearn.tree._classes.DecisionTreeClassifier	True
svm	SVM - Linear Kernel	sklearn.linear_model._stochastic_gradient.SGDC...	True
rbfsvm	SVM - Radial Kernel	sklearn.svm._classes.SVC	False
gpc	Gaussian Process Classifier	sklearn.gaussian_process._gpc.GaussianProcessC...	False
mlp	MLP Classifier	sklearn.neural_network._multilayer_perceptron....	False
ridge	Ridge Classifier	sklearn.linear_model._ridge.RidgeClassifier	True
rf	Random Forest Classifier	sklearn.ensemble._forest.RandomForestClassifier	True
qda	Quadratic Discriminant Analysis	sklearn.discriminant_analysis.QuadraticDiscrim...	True
ada	Ada Boost Classifier	sklearn.ensemble._weight_boosting.AdaBoostClas...	True
gbc	Gradient Boosting Classifier	sklearn.ensemble._gb.GradientBoostingClassifier	True
lda	Linear Discriminant Analysis	sklearn.discriminant_analysis.LinearDiscrimina...	True
et	Extra Trees Classifier	sklearn.ensemble._forest.ExtraTreesClassifier	True
lightgbm	Light Gradient Boosting Machine	lightgbm.sklearn.LGBMClassifier	True

Figure 151: False & True for all models. All_Clusters.CSV when plant layer is as the origin point

5.1. Decision Tree Classifier

```
dt = create_model('dt')
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.8963	0.5524	0.1250	0.4000	0.1905	0.1510	0.1807
1	0.9024	0.6115	0.2500	0.5000	0.3333	0.2870	0.3071
2	0.9018	0.6464	0.3333	0.4545	0.3846	0.3327	0.3374
3	0.9018	0.5865	0.2000	0.4286	0.2727	0.2275	0.2466
4	0.8957	0.5232	0.0667	0.2500	0.1053	0.0692	0.0867
5	0.9080	0.5300	0.0667	0.5000	0.1176	0.0981	0.1573
6	0.9080	0.5300	0.0667	0.5000	0.1176	0.0981	0.1573
7	0.8773	0.4831	0.0000	0.0000	0.0000	-0.0482	-0.0566
8	0.9080	0.5899	0.2000	0.5000	0.2857	0.2461	0.2759
9	0.9141	0.5903	0.1875	0.7500	0.3000	0.2714	0.3475
Mean	0.9014	0.5643	0.1496	0.4283	0.2107	0.1733	0.2040
SD	0.0097	0.0464	0.0969	0.1846	0.1166	0.1131	0.1194

Figure 152: Decision Tree Classifier - All_Clusters.CSV when plant layer is as the origin point

5.2. K Neighbors Classifier

```
knn = create_model('knn')
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.9024	0.4155	0.0000	0.0000	0.0000	0.0000	0.0000
1	0.9024	0.5363	0.0000	0.0000	0.0000	0.0000	0.0000
2	0.9080	0.5568	0.0000	0.0000	0.0000	0.0000	0.0000
3	0.9080	0.5748	0.0000	0.0000	0.0000	0.0000	0.0000
4	0.9080	0.4919	0.0000	0.0000	0.0000	0.0000	0.0000
5	0.9080	0.4446	0.0000	0.0000	0.0000	0.0000	0.0000
6	0.9080	0.5025	0.0000	0.0000	0.0000	0.0000	0.0000
7	0.9080	0.5495	0.0000	0.0000	0.0000	0.0000	0.0000
8	0.9080	0.5383	0.0000	0.0000	0.0000	0.0000	0.0000
9	0.9018	0.4945	0.0000	0.0000	0.0000	0.0000	0.0000
Mean	0.9063	0.5105	0.0000	0.0000	0.0000	0.0000	0.0000
SD	0.0026	0.0482	0.0000	0.0000	0.0000	0.0000	0.0000

Figure 153: K Neighbors Classifier - All_Clusters.CSV when plant layer is as the origin point

5.3. Random Forest Classifier

```
rf = create_model('rf')
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.9024	0.7076	0.0000	0.0000	0.0000	0.0000	0.0000
1	0.9024	0.7663	0.0000	0.0000	0.0000	0.0000	0.0000
2	0.9080	0.7957	0.0000	0.0000	0.0000	0.0000	0.0000
3	0.9080	0.6532	0.0000	0.0000	0.0000	0.0000	0.0000
4	0.9080	0.7275	0.0000	0.0000	0.0000	0.0000	0.0000
5	0.9080	0.5782	0.0000	0.0000	0.0000	0.0000	0.0000
6	0.9080	0.7739	0.0000	0.0000	0.0000	0.0000	0.0000
7	0.9080	0.6227	0.0000	0.0000	0.0000	0.0000	0.0000
8	0.9080	0.7250	0.0000	0.0000	0.0000	0.0000	0.0000
9	0.9018	0.7953	0.0000	0.0000	0.0000	0.0000	0.0000
Mean	0.9063	0.7145	0.0000	0.0000	0.0000	0.0000	0.0000
SD	0.0026	0.0710	0.0000	0.0000	0.0000	0.0000	0.0000

Figure 154: Random Forest Classifier - All_Clusters.CSV when plant layer is as the origin point

5.4. Light Gradient Boosting Machine

```
lightgbm = create_model('lightgbm')
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.9024	0.6643	0.1875	0.5000	0.2727	0.2319	0.2643
1	0.8780	0.7492	0.1250	0.2500	0.1667	0.1087	0.1163
2	0.8957	0.7703	0.3333	0.4167	0.3704	0.3143	0.3166
3	0.8834	0.7032	0.1333	0.2500	0.1739	0.1174	0.1242
4	0.9018	0.7917	0.1333	0.4000	0.2000	0.1614	0.1895
5	0.8957	0.6203	0.1333	0.3333	0.1905	0.1455	0.1632
6	0.9080	0.7696	0.2667	0.5000	0.3478	0.3032	0.3206
7	0.8834	0.7011	0.1333	0.2500	0.1739	0.1174	0.1242
8	0.9018	0.7435	0.2667	0.4444	0.3333	0.2839	0.2947
9	0.9141	0.8856	0.3750	0.6000	0.4615	0.4176	0.4312
Mean	0.8965	0.7399	0.2088	0.3944	0.2691	0.2201	0.2345
SD	0.0110	0.0698	0.0895	0.1157	0.0986	0.1007	0.1014

Figure 155: Light Gradient Boosting Machine - All_Clusters.CSV when plant layer is as the origin point

6.1. Decision Tree Classifier

```
tuned_dt = tune_model(dt)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.9024	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
1	0.9024	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
2	0.9080	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
3	0.9080	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
4	0.9080	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
5	0.9080	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
6	0.9080	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
7	0.9080	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
8	0.9080	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
9	0.9018	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
Mean	0.9063	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
SD	0.0026	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Figure 156: Tunning of Decision Tree Classifier - All_Clusters.CSV when plant layer is as the origin point

6.2. K Neighbors Classifier

```
tuned_knn = tune_model(knn)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.9024	0.5498	0.0000	0.0000	0.0000	0.0000	0.0000
1	0.9024	0.6119	0.0000	0.0000	0.0000	0.0000	0.0000
2	0.9080	0.7595	0.0000	0.0000	0.0000	0.0000	0.0000
3	0.9141	0.6032	0.1333	0.6667	0.2222	0.1976	0.2722
4	0.9018	0.6468	0.0000	0.0000	0.0000	-0.0116	-0.0250
5	0.9080	0.5782	0.0000	0.0000	0.0000	0.0000	0.0000
6	0.9080	0.6887	0.0667	0.5000	0.1176	0.0981	0.1573
7	0.9080	0.5919	0.0000	0.0000	0.0000	0.0000	0.0000
8	0.9202	0.6838	0.1333	1.0000	0.2353	0.2184	0.3501
9	0.9018	0.7109	0.0000	0.0000	0.0000	0.0000	0.0000
Mean	0.9075	0.6425	0.0333	0.2167	0.0575	0.0502	0.0755
SD	0.0057	0.0632	0.0537	0.3500	0.0925	0.0844	0.1285

Figure 157: Tunning of K Neighbors Classifier - All_Clusters.CSV when plant layer is as the origin point

```
import numpy as np
tuned_knn = tune_model(knn, custom_grid = {'n_neighbors' : np.arange(0,10,1)})
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.9024	0.4899	0.0000	0.0000	0.0000	0.0000	0.0000
1	0.9024	0.4764	0.0000	0.0000	0.0000	0.0000	0.0000
2	0.9080	0.4899	0.0000	0.0000	0.0000	0.0000	0.0000
3	0.9080	0.5063	0.0000	0.0000	0.0000	0.0000	0.0000
4	0.9080	0.5097	0.0000	0.0000	0.0000	0.0000	0.0000
5	0.9080	0.4392	0.0000	0.0000	0.0000	0.0000	0.0000
6	0.9080	0.4865	0.0000	0.0000	0.0000	0.0000	0.0000
7	0.9080	0.5029	0.0000	0.0000	0.0000	0.0000	0.0000
8	0.9080	0.4392	0.0000	0.0000	0.0000	0.0000	0.0000
9	0.9018	0.4796	0.0000	0.0000	0.0000	0.0000	0.0000
Mean	0.9063	0.4819	0.0000	0.0000	0.0000	0.0000	0.0000
SD	0.0026	0.0238	0.0000	0.0000	0.0000	0.0000	0.0000

Figure 158: Tunning of K Neighbors Classifier(Using custom grid) - All_Clusters.CSV when plant layer is as the origin point

6.3. Random Forest Classifier

```
tuned_rf = tune_model(rf)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.8902	0.6421	0.0625	0.2500	0.1000	0.0635	0.0812
1	0.9085	0.7016	0.1250	0.6667	0.2105	0.1854	0.2618
2	0.9141	0.7577	0.2667	0.5714	0.3636	0.3241	0.3513
3	0.9080	0.6908	0.0000	0.0000	0.0000	0.0000	0.0000
4	0.8957	0.6698	0.0000	0.0000	0.0000	-0.0221	-0.0355
5	0.9080	0.6977	0.0000	0.0000	0.0000	0.0000	0.0000
6	0.9018	0.6200	0.0000	0.0000	0.0000	-0.0116	-0.0250
7	0.9080	0.6097	0.0000	0.0000	0.0000	0.0000	0.0000
8	0.9141	0.6471	0.0667	1.0000	0.1250	0.1148	0.2468
9	0.9202	0.8027	0.1875	1.0000	0.3158	0.2939	0.4150
Mean	0.9069	0.6839	0.0708	0.3488	0.1115	0.0948	0.1296
SD	0.0085	0.0574	0.0895	0.4021	0.1333	0.1240	0.1630

Figure 159: Tunning of Random Forest Classifier - All_Clusters.CSV when plant layer is as the origin point

6.4. Light Gradient Boosting Machine

```
tuned_lightgbm = tune_model(lightgbm)
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.9024	0.6429	0.0625	0.5000	0.1111	0.0914	0.1507
1	0.9024	0.6822	0.0000	0.0000	0.0000	0.0000	0.0000
2	0.9080	0.7739	0.0000	0.0000	0.0000	0.0000	0.0000
3	0.9080	0.7541	0.0000	0.0000	0.0000	0.0000	0.0000
4	0.9141	0.7525	0.0667	1.0000	0.1250	0.1148	0.2468
5	0.9080	0.7061	0.0000	0.0000	0.0000	0.0000	0.0000
6	0.9080	0.6995	0.0000	0.0000	0.0000	0.0000	0.0000
7	0.9018	0.6545	0.0000	0.0000	0.0000	-0.0116	-0.0250
8	0.9141	0.7941	0.0667	1.0000	0.1250	0.1148	0.2468
9	0.9018	0.8459	0.0000	0.0000	0.0000	0.0000	0.0000
Mean	0.9069	0.7306	0.0196	0.2500	0.0361	0.0309	0.0619
SD	0.0045	0.0613	0.0299	0.4031	0.0553	0.0503	0.1033

Figure 160: Tunning of Light Gradient Boosting Machine - All_Clusters.CSV when school layer is as the origin point

```
plot_model(tuned_lightgbm, plot = 'auc')
```

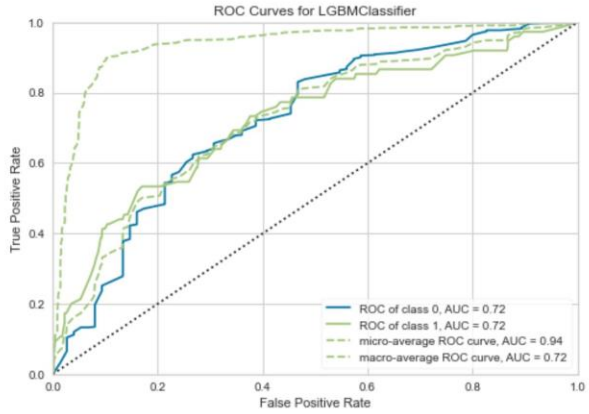


Figure 161: AUC plot for tuned Lightgbm model - All_Clusters.CSV when plant layer is as the origin point

```
plot_model(tuned_dt, plot = 'auc')
```

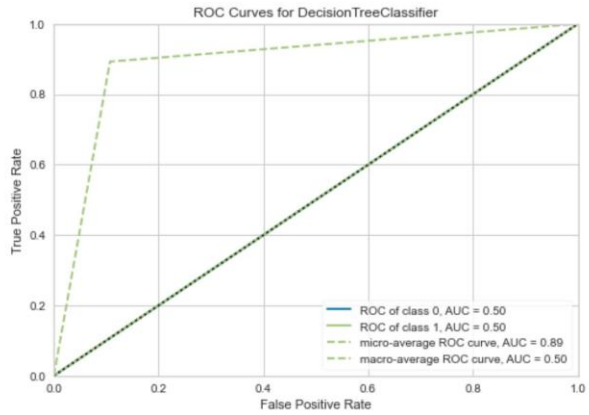


Figure 162: AUC plot for tuned decision tree model - All_Clusters.CSV when plant layer is as the origin point

```
plot_model(tuned_knn, plot = 'auc')
```

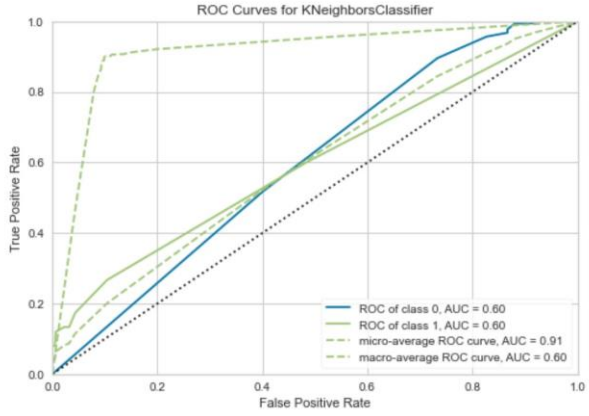


Figure 163: AUC plot for tuned K Neighbors model - All_Clusters.CSV when plant layer is as the origin point

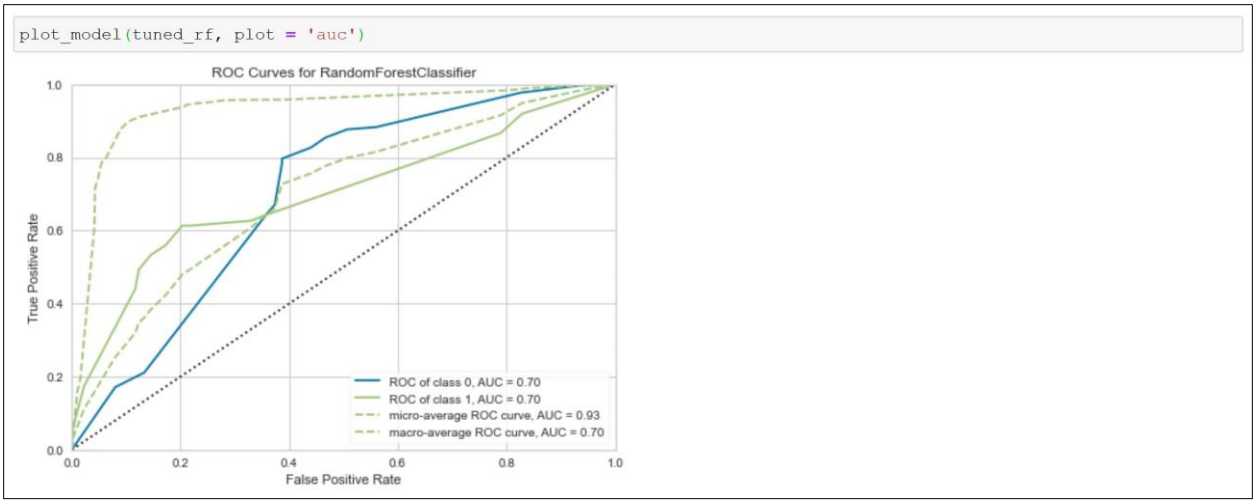


Figure 164: AUC plot for tuned Random Forest model - All_Clusters.CSV when plant layer is as the origin point

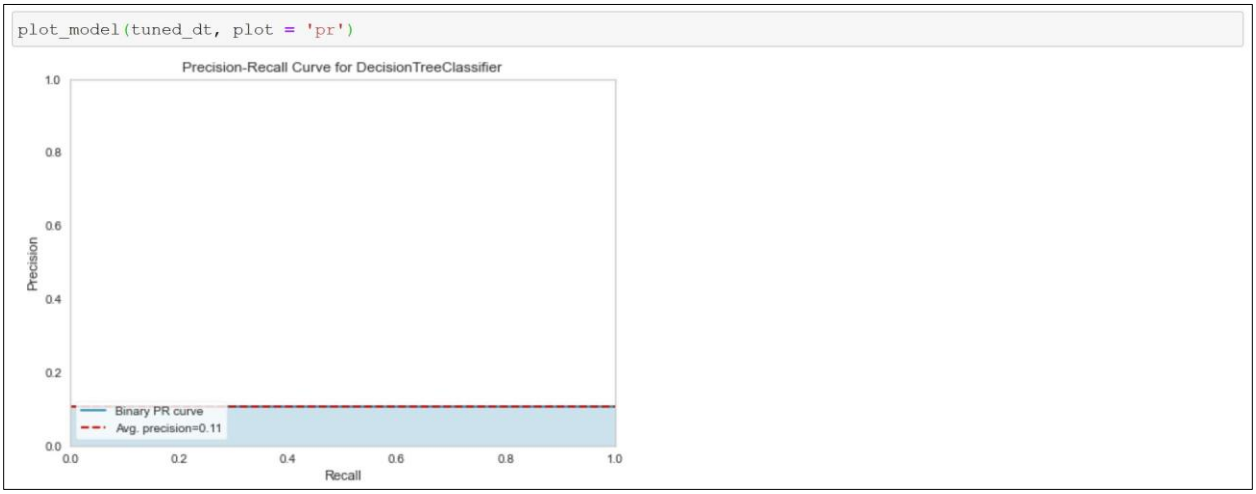


Figure 165: Precision-recall curve plot for tuned decision tree model - All_Clusters.CSV when plant layer is as the origin point

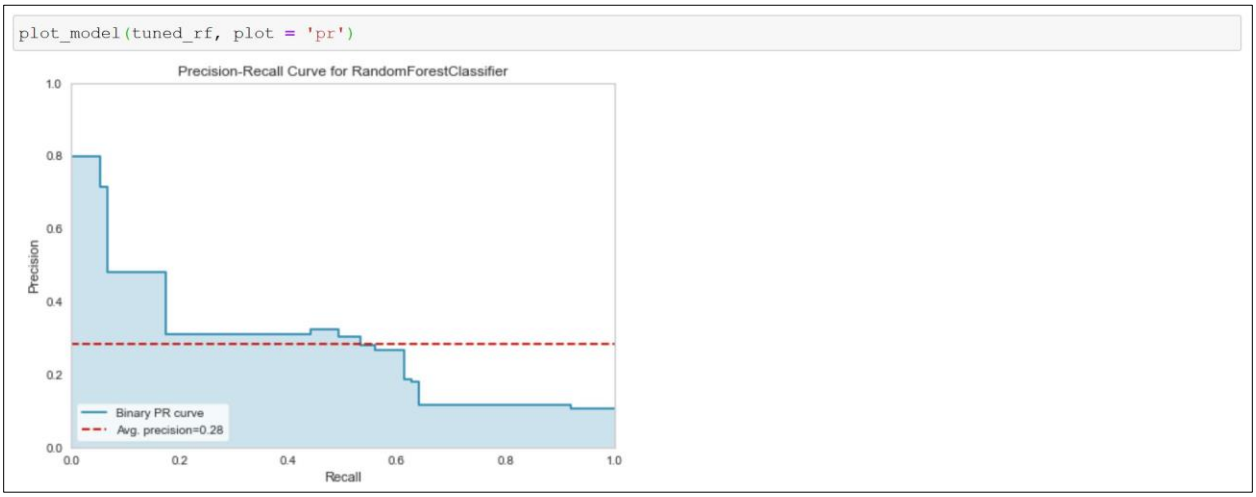


Figure 166: Precision-recall curve plot for tuned Random Forest model - All_Clusters.CSV when plant layer is as the origin point

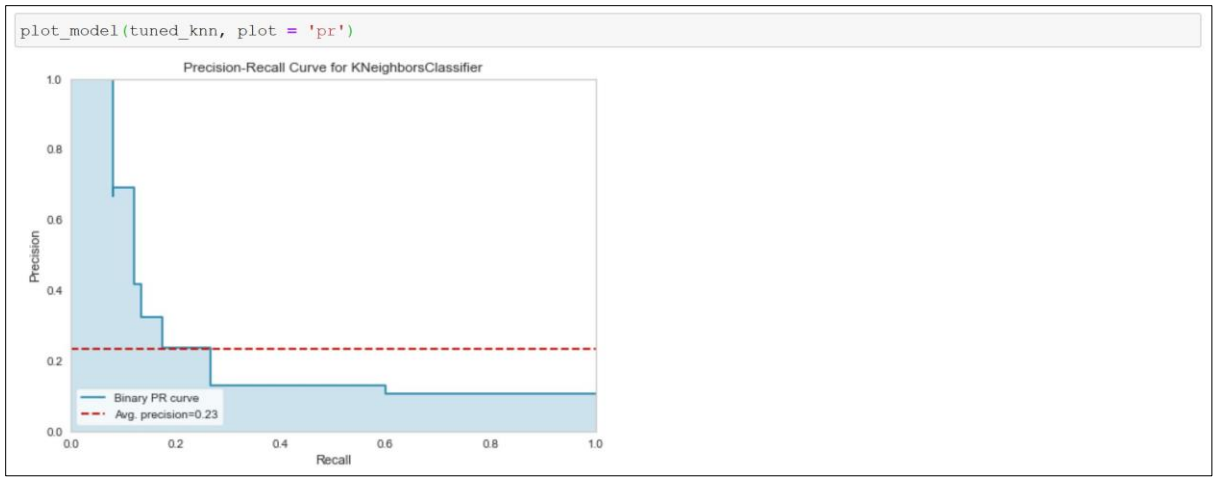


Figure 167: Precision-recall curve plot for tuned K Neighbors model - All_Clusters.CSV when plant layer is as the origin point

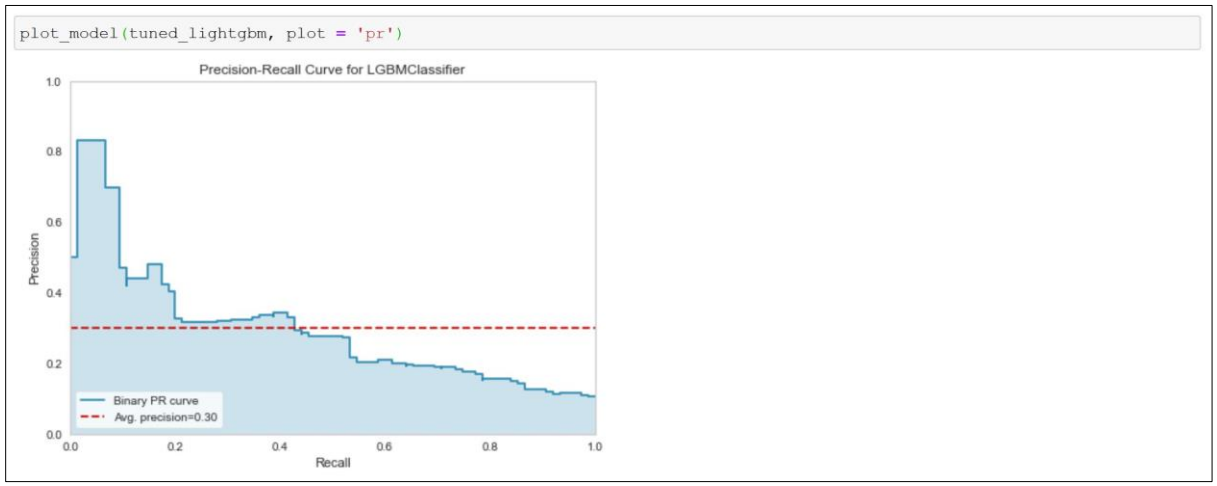


Figure 168: Precision-recall curve plot for tuned Lightgbm model - All_Clusters.CSV when plant layer is as the origin point

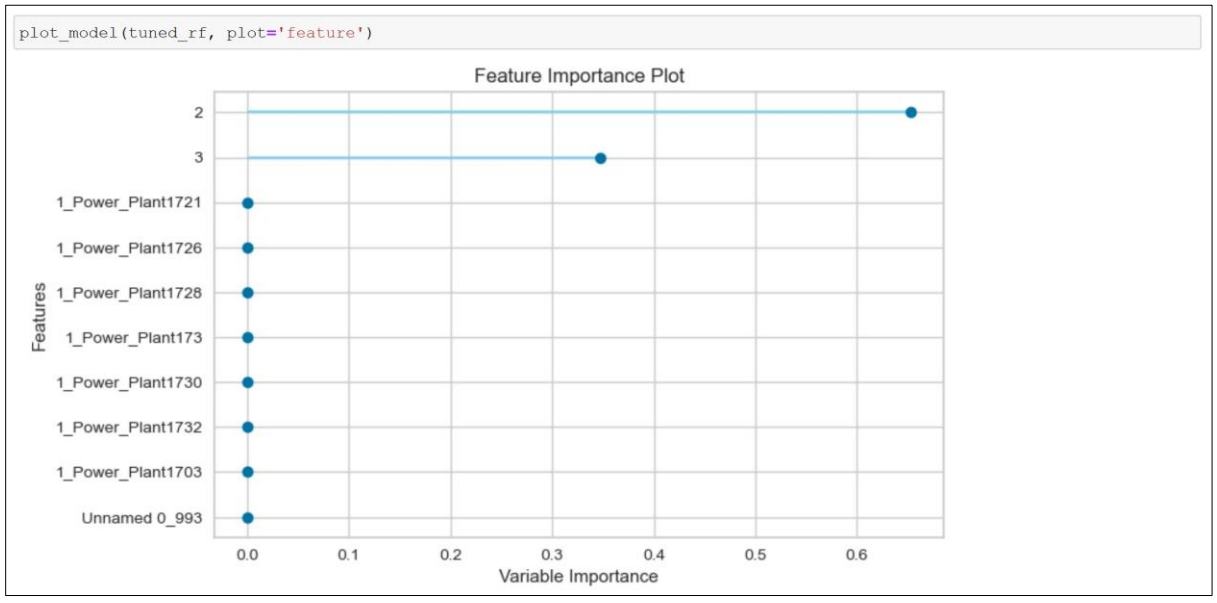


Figure 169: Feature importance plot for tuned Random Forest model - All_Clusters.CSV when plant layer is as the origin point

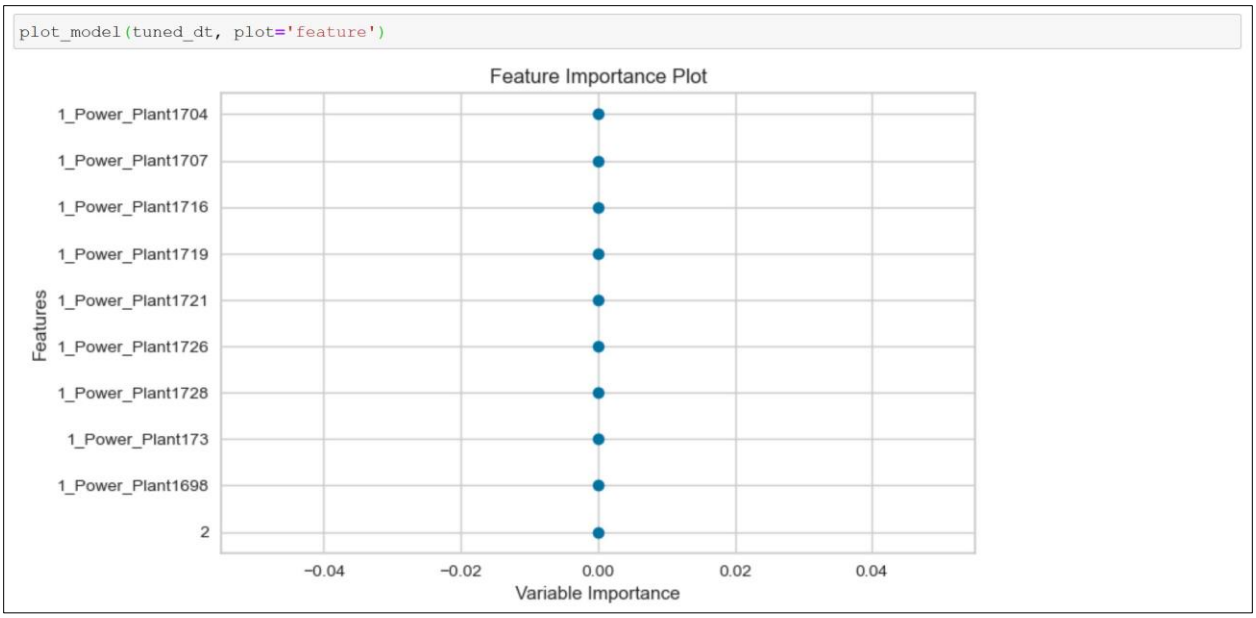


Figure 170: Feature importance plot for tuned decision tree model - All_Clusters.CSV when plant layer is as the origin point

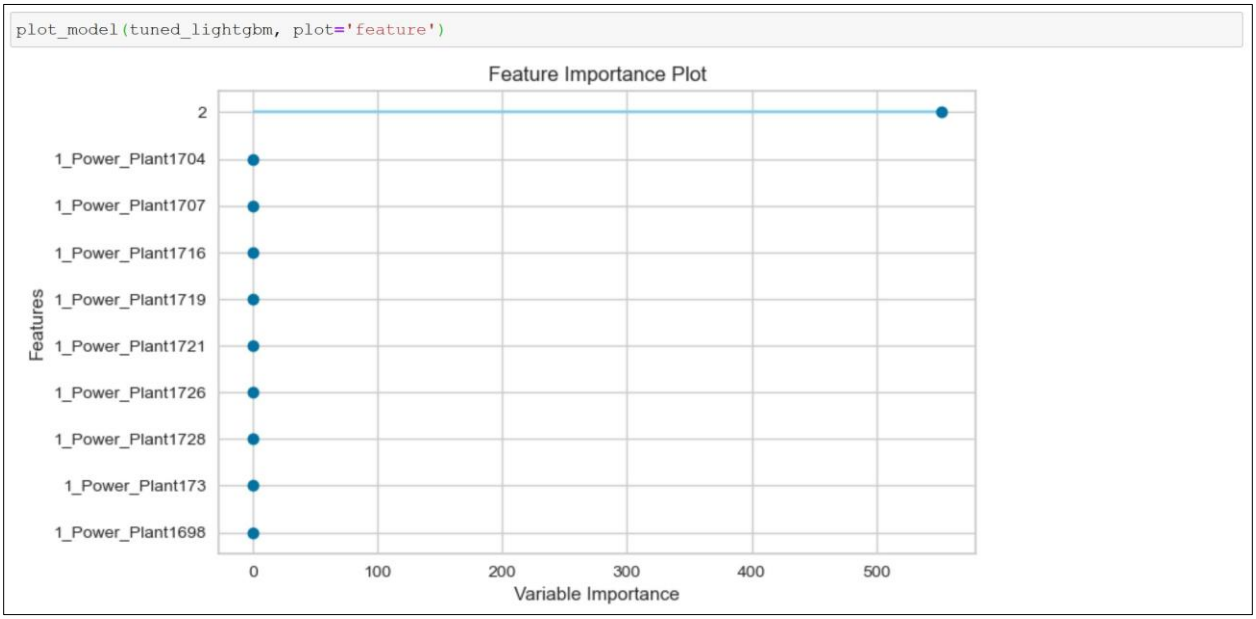


Figure 171: Feature importance plot for tuned Lightgbm model - All_Clusters.CSV when plant layer is as the origin point

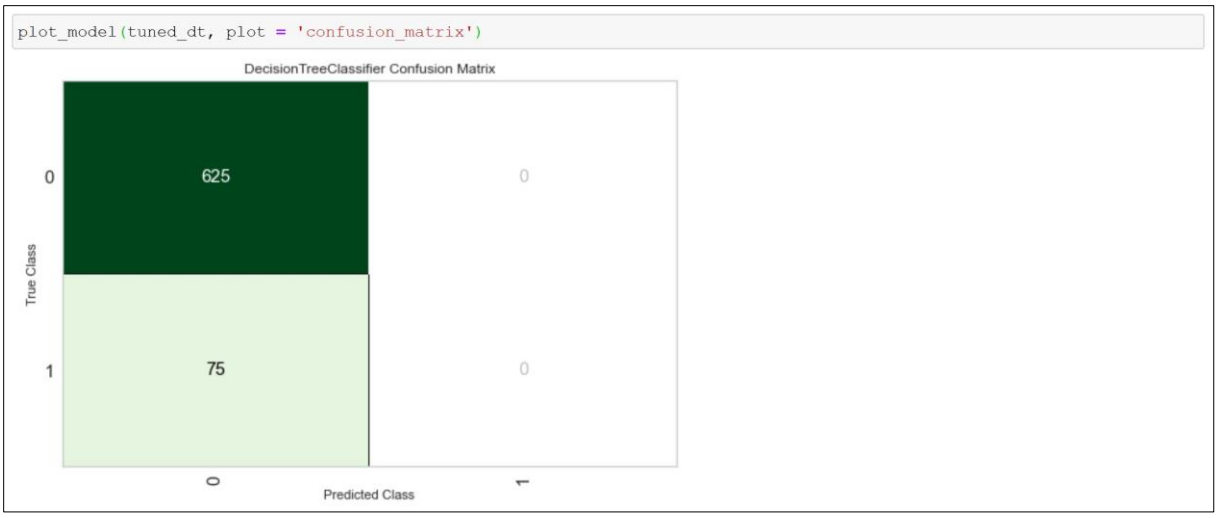


Figure 172: Confusion matrix plot for tuned decision tree model - All_Clusters.CSV when plant layer is as the origin point

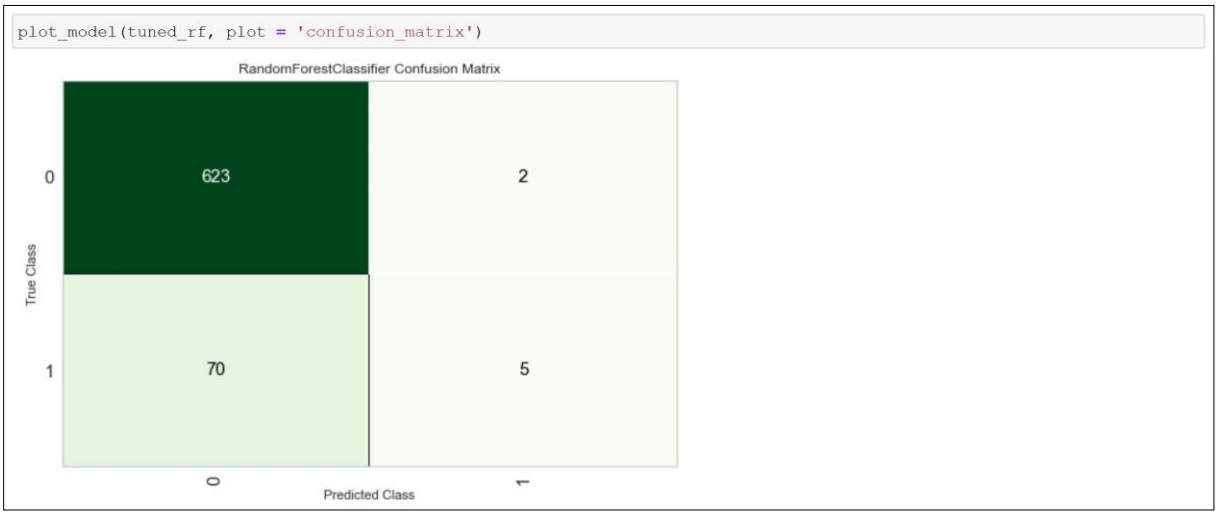


Figure 173: Confusion matrix plot for tuned Random Forest model - All_Clusters.CSV when school layer is as the origin point

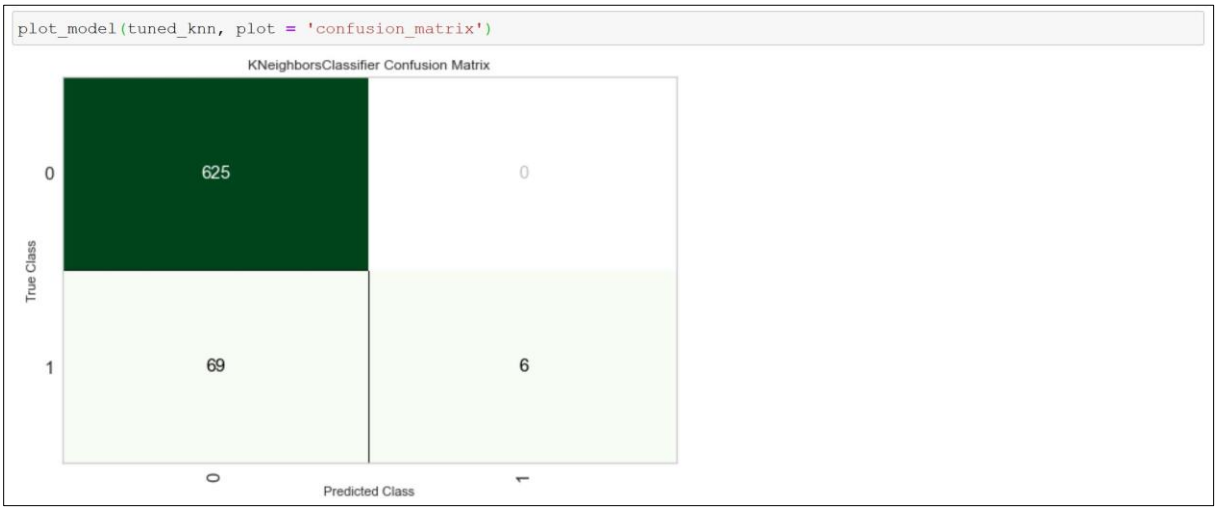


Figure 174: Confusion matrix plot for tuned K Neighbors model - All_Clusters.CSV when plant layer is as the origin point

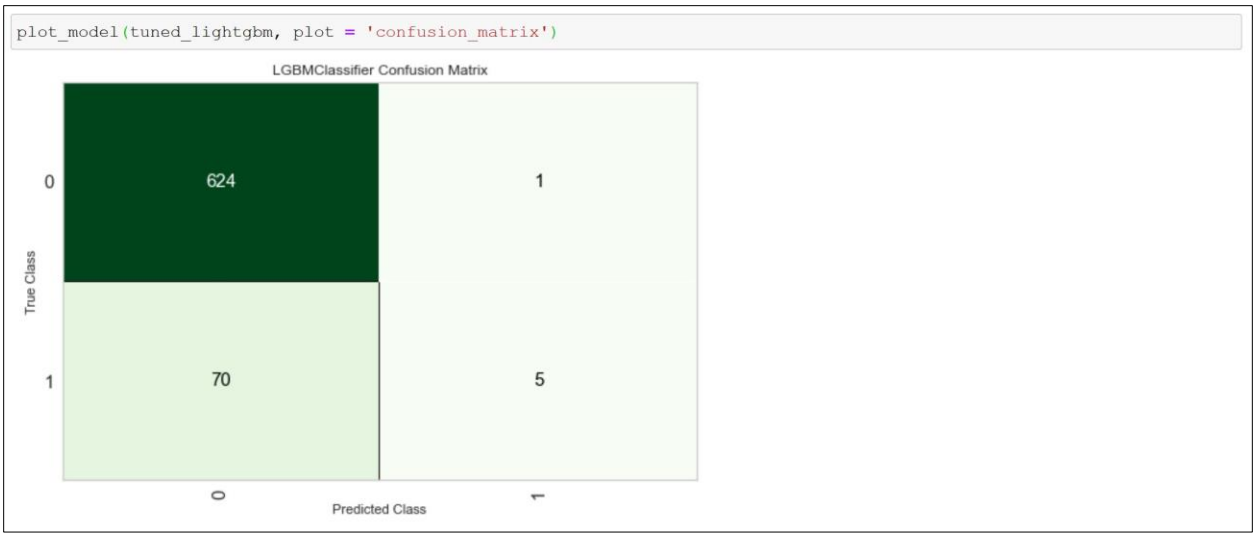


Figure 175: Confusion matrix plot for tuned Lightgbm model - All_Clusters.CSV when plant layer is as the origin point

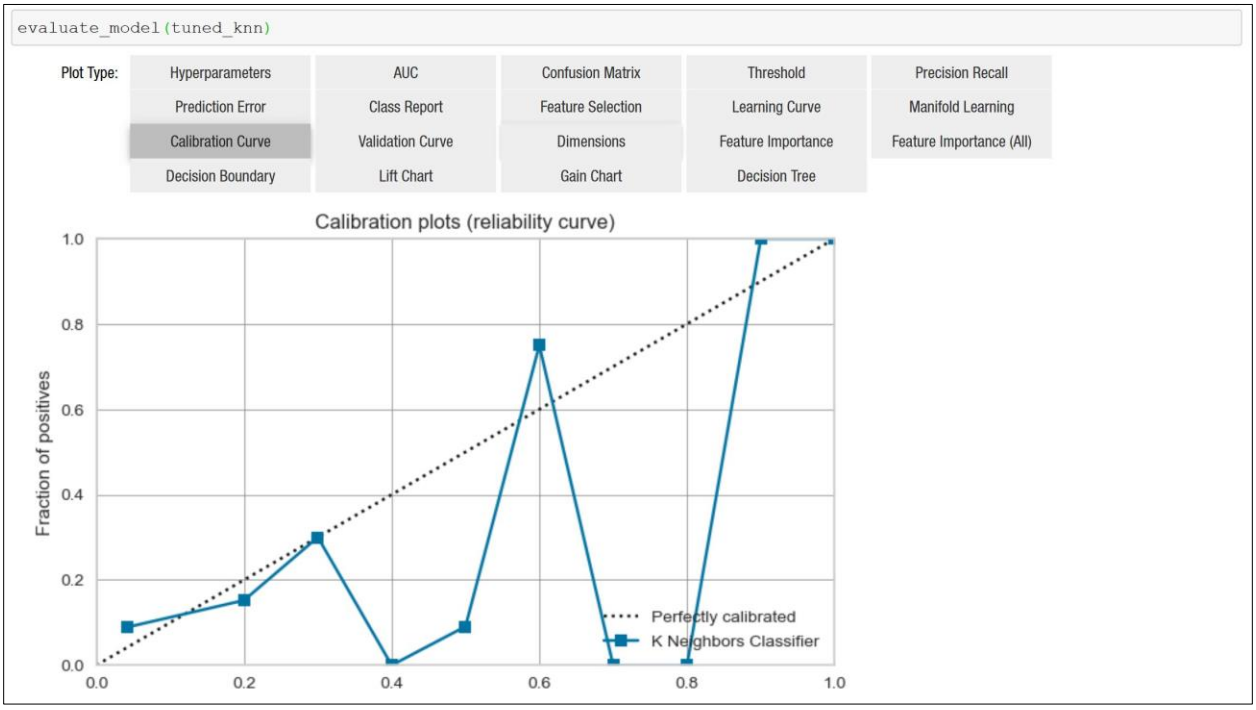


Figure 176: Calibration curves plot for tuned K Neighbors model - All_Clusters.CSV when plant layer is as the origin point

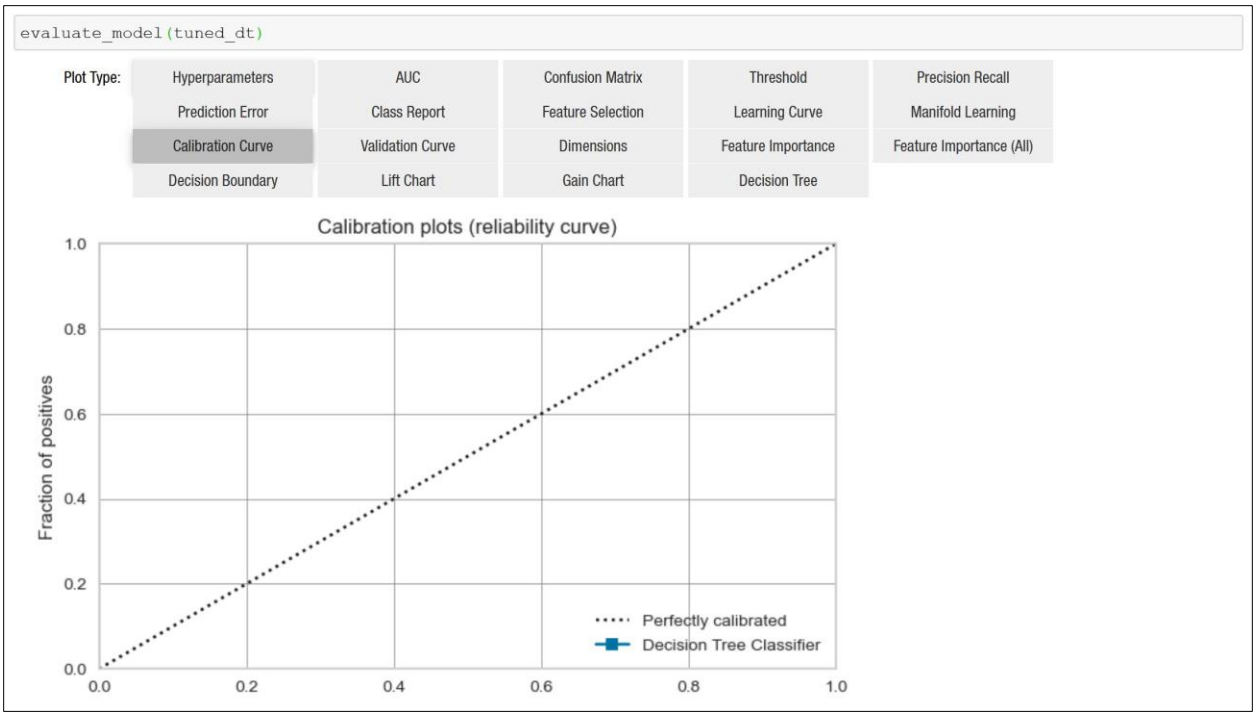


Figure 177: Calibration curves plot for tuned decision tree model - All_Clusters.CSV when plant layer is as the origin point

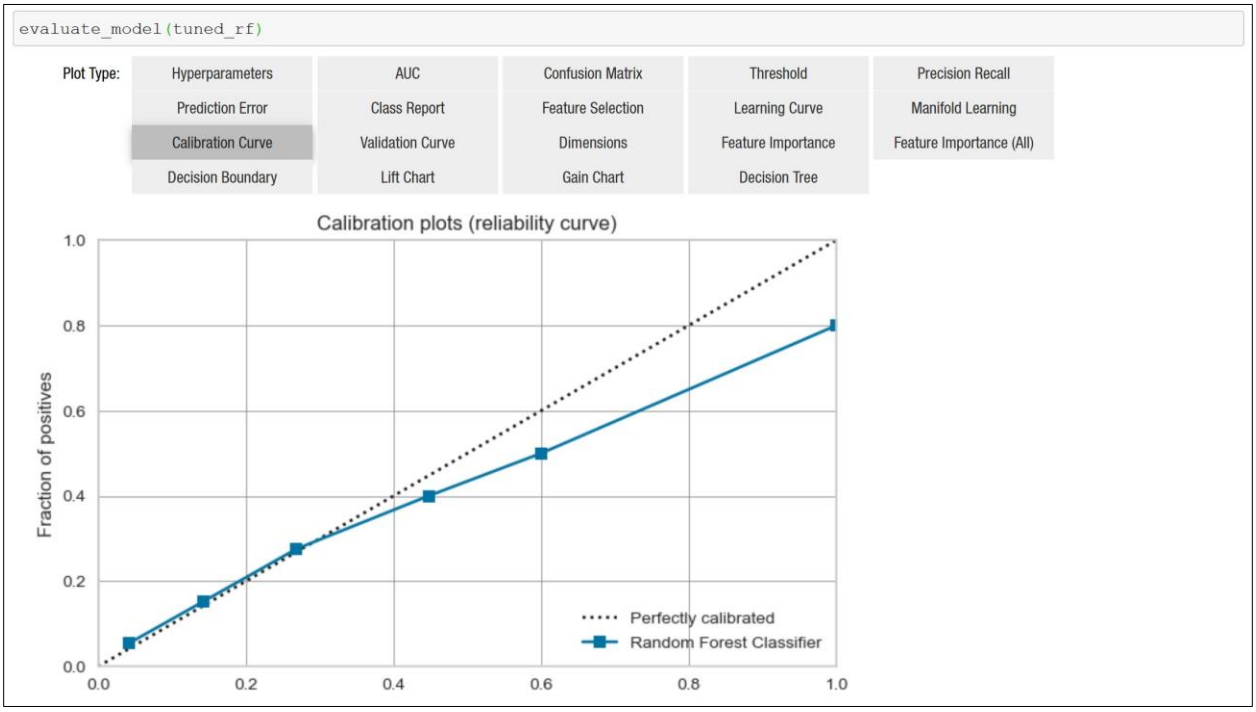


Figure 178: Calibration curves plot for tuned Random Forest model - All_Clusters.CSV when plant layer is as the origin point

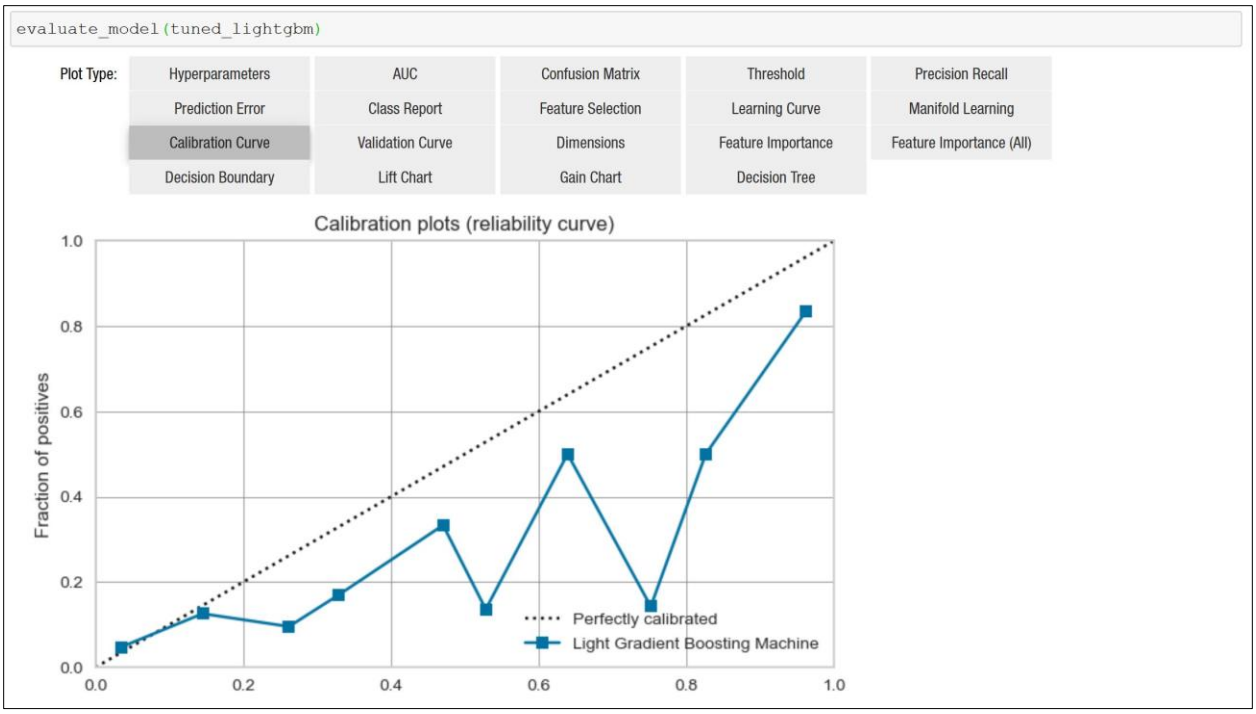


Figure 179: Calibration curves plot for tuned Lightgbm model - All_Clusters.CSV when plant layer is as the origin point

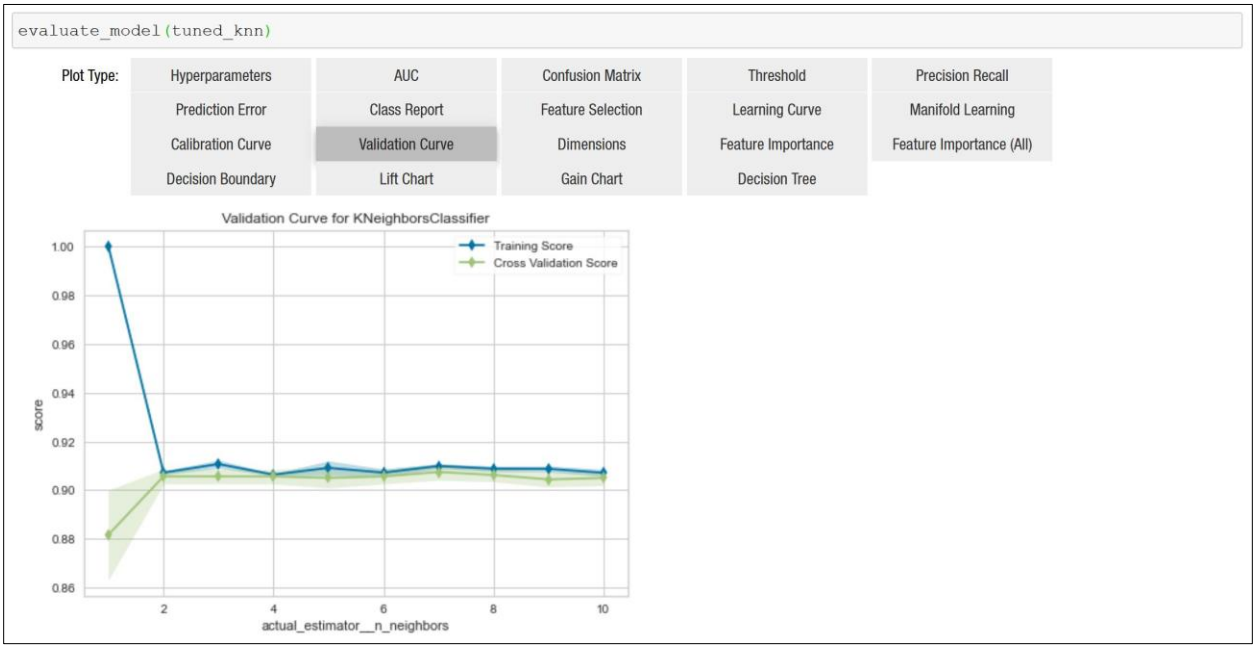


Figure 180: Validation curve plot for tuned K Neighbors model - All_Clusters.CSV when plant layer is as the origin point

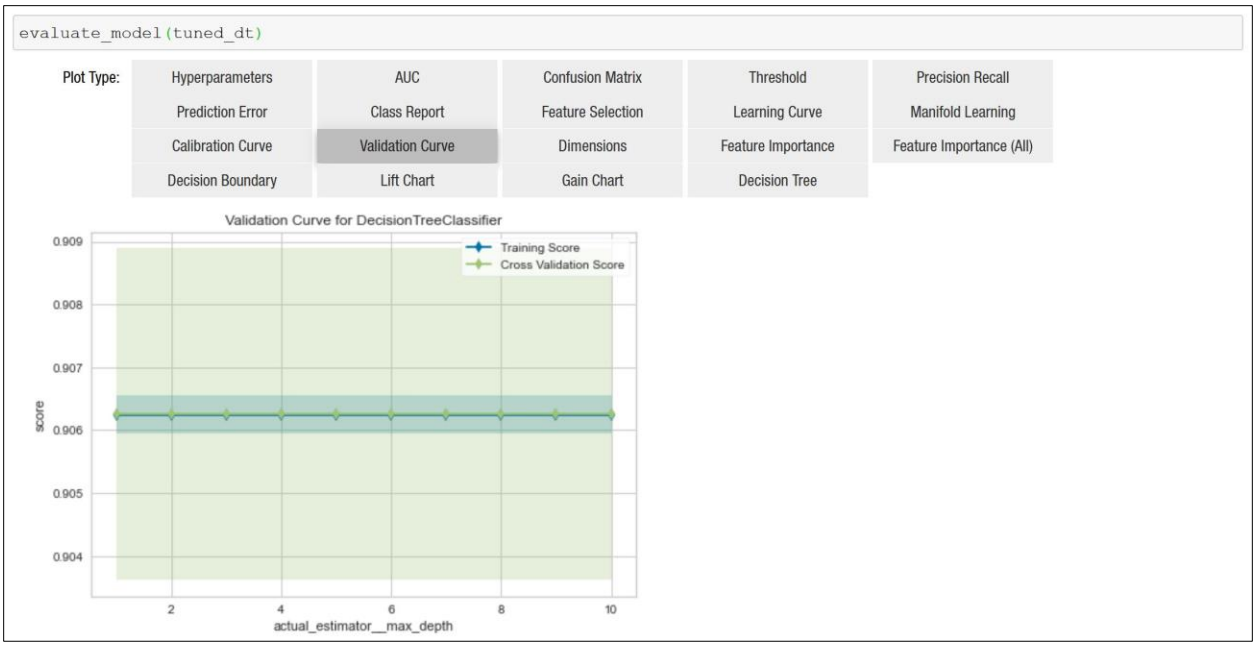


Figure 181: Validation curve plot for decision tree model - All_Clusters.CSV when plant layer is as the origin point

Figure 182: Validation curve plot for Random Forest model - All_Clusters.CSV when plant layer is as the origin point

Figure 183: Validation curve plot for Lightgbm model - All_Clusters.CSV when plant layer is as the origin point

10. Predict on unseen data

The predict_model() function is also used to predict on the unseen dataset. The only difference from section 9 above is that this time we will pass the data_unseen parameter. data_unseen is the variable created at the beginning of this test and contains 5% (123 samples) of the original dataset which was never exposed to PyCaret. (see section 5 for explanation).

```
unseen_predictions = predict_model(final_knn, data=data_unseen)
unseen_predictions.head()
```

	Unnamed: 0	0	1	2	3	Label	Score
0	283	0.0	Power_Plant994	34.083539	-117.234824	0.0	0.9375
1	291	0.0	Power_Plant987	34.082326	-117.242921	0.0	0.9375
2	295	0.0	Power_Plant982	34.085366	-117.267032	0.0	0.9375
3	335	0.0	Power_Plant94	37.946381	-120.530078	0.0	0.9375
4	361	0.0	Power_Plant916	38.166267	-121.800618	0.0	1.0000

Figure 184: Prediction on unseen data - All_Clusters.CSV when plant layer is as the origin point

The reason for returning the 0 labels for two tries is that most of the data have the 0 labels so test data are closer to the range of non-important point or cluster 0.

With investigating all the results and after comparison between the results of two tries (When the school layer is as an origin point and when plants are as an origin point) we can confirm that the KNN algorithm is the selected model among all other models for deploying it in the plugin since it has a better performance compared to the other models.

And it is time to map the final result to the QGIS for checking which cell can have the potential for establishing the new factory.

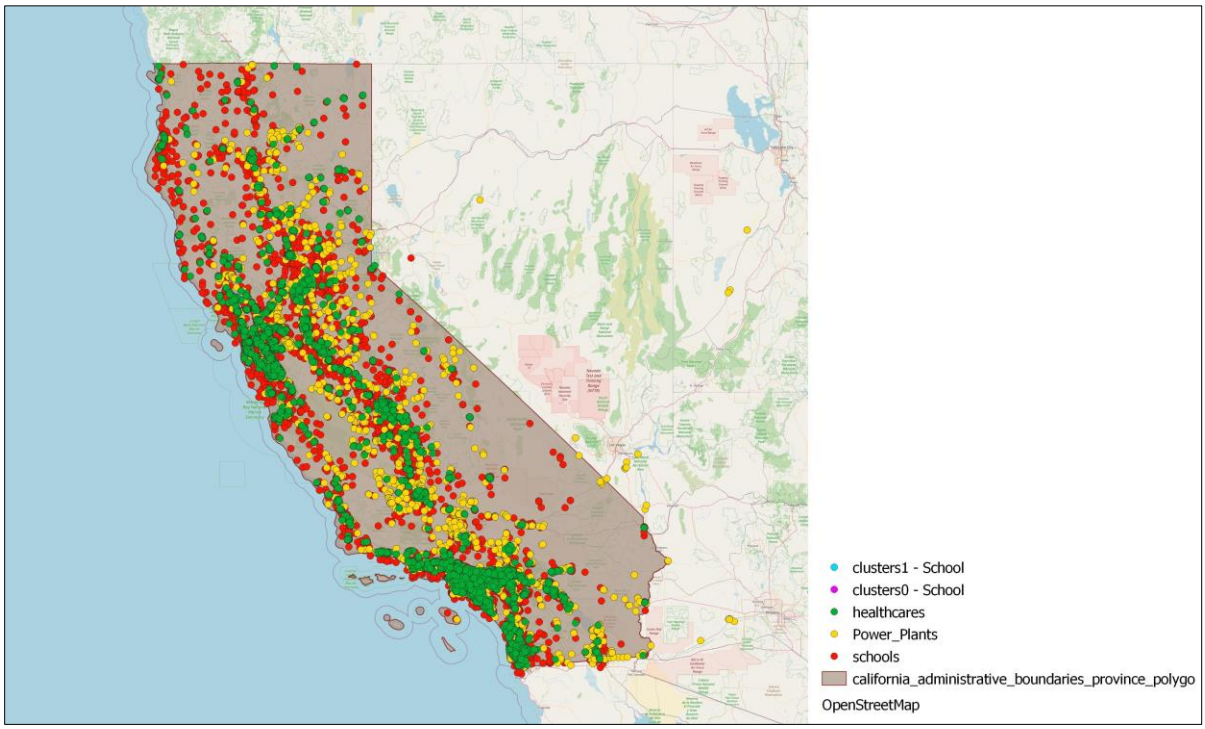


Figure 185: Distribution of Healthcare, plants, and schools' points in California State when School is as the origin point

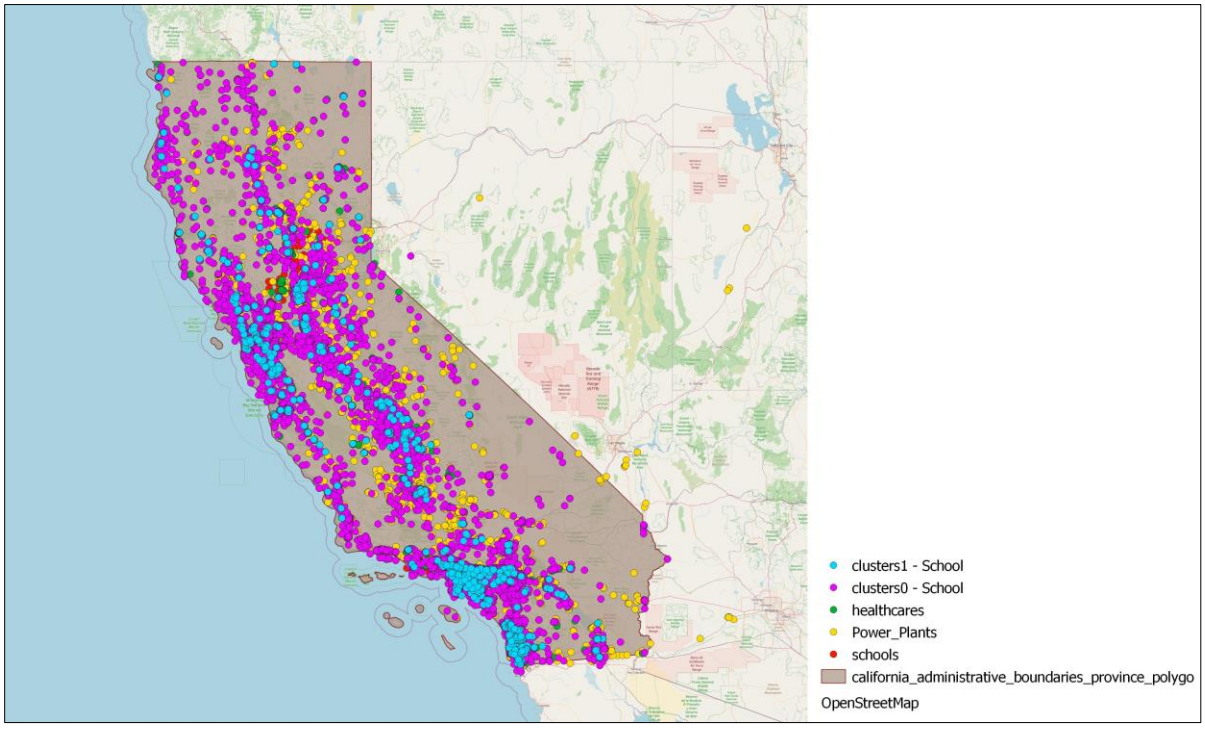


Figure 186: Distribution of cluster 0 and 1 in California State when School is as the origin point

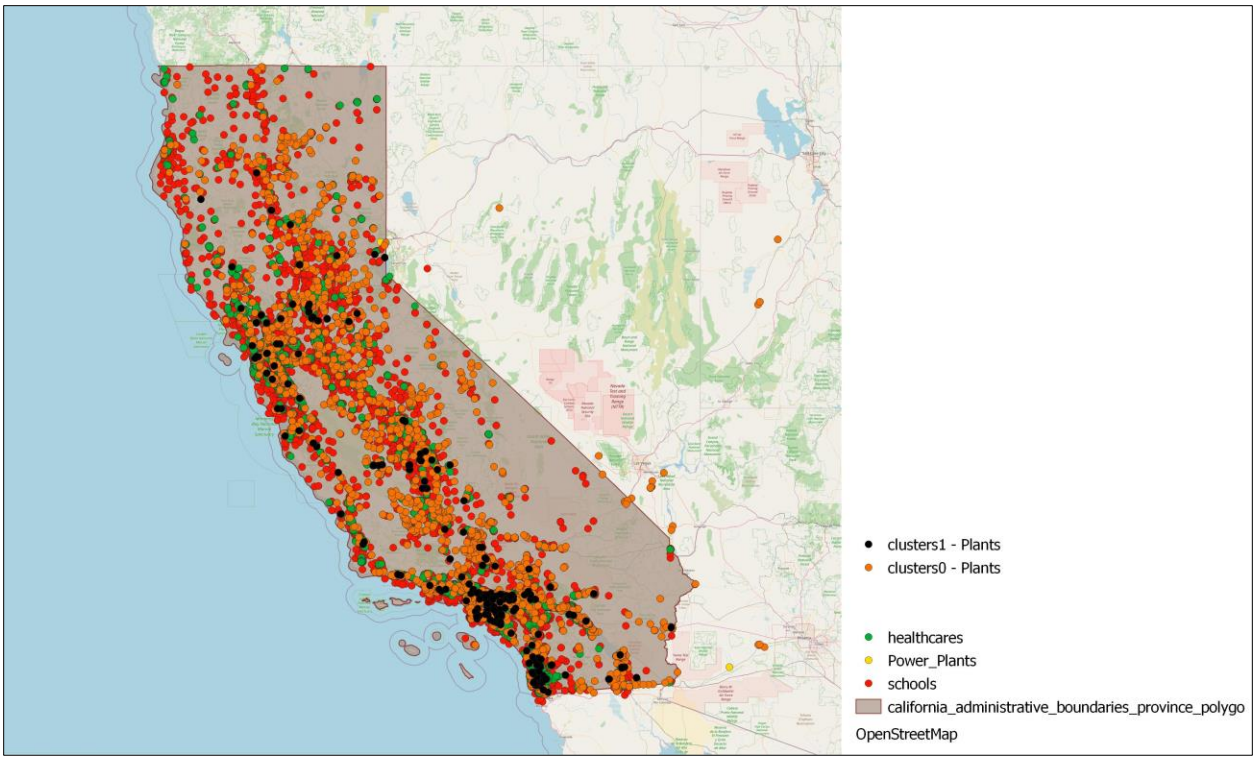


Figure 187: Distribution of cluster 0 and 1 in California State when the plant is as the origin point

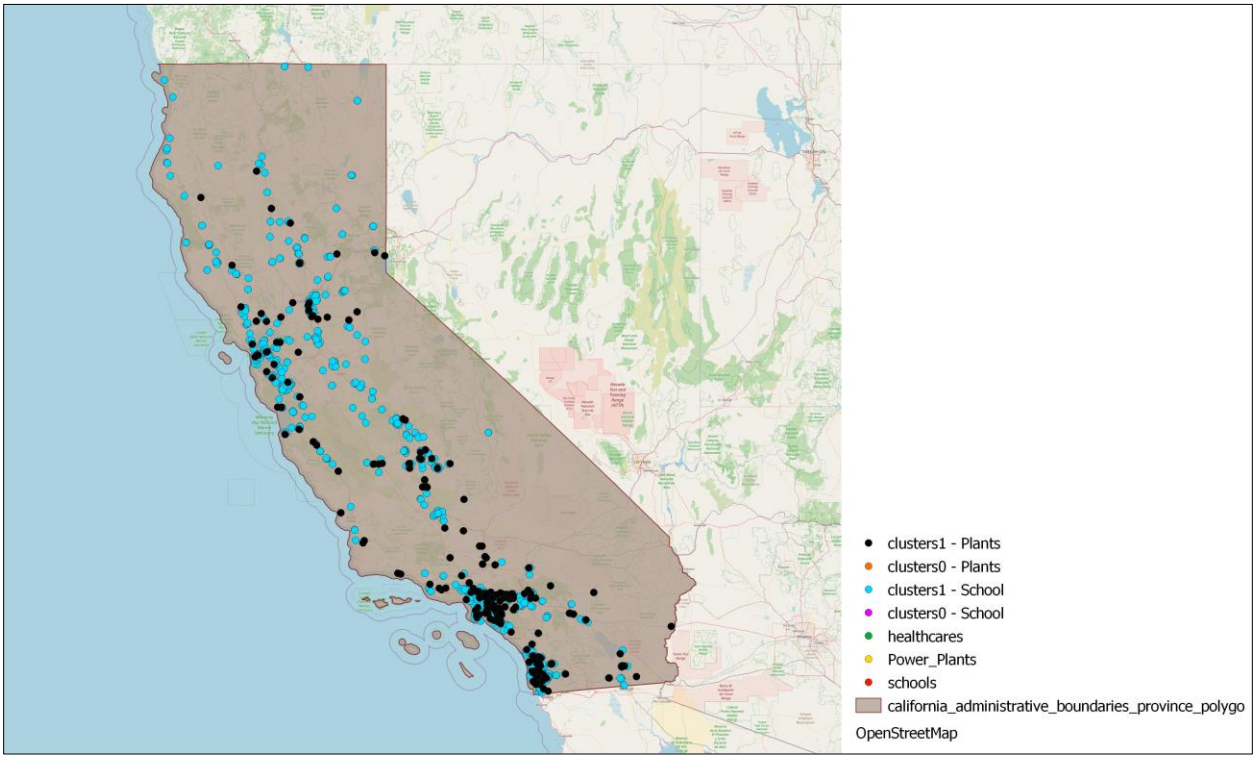


Figure 188: Distribution of cluster 1 in California State once when school is as origin point and once when the plant is as the origin point

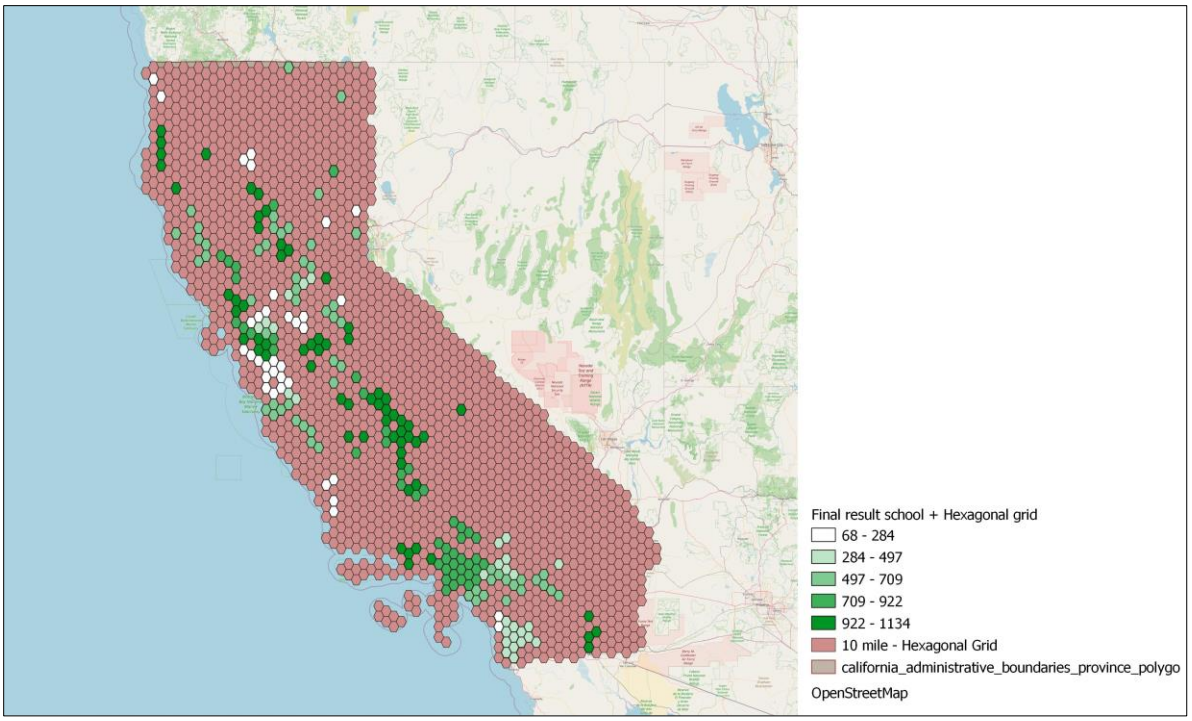


Figure 189: Indexing the California state (Coloring the parcels (Hexagonal grid))- when school is as the origin point

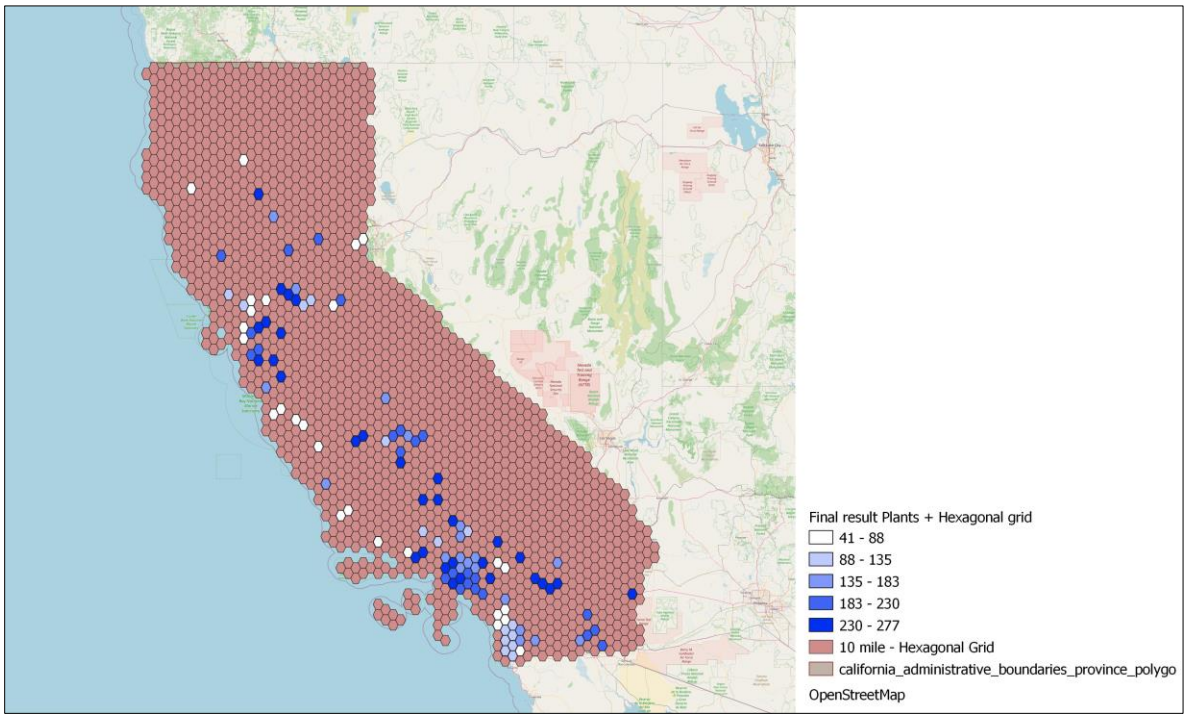


Figure 190: Indexing the California state (Coloring the parcels (Hexagonal grid))- when the plant is as the origin point

We can see that the importance of the parcel would be changed based on our priority in selecting the origin layer for analyzing and finding the location.

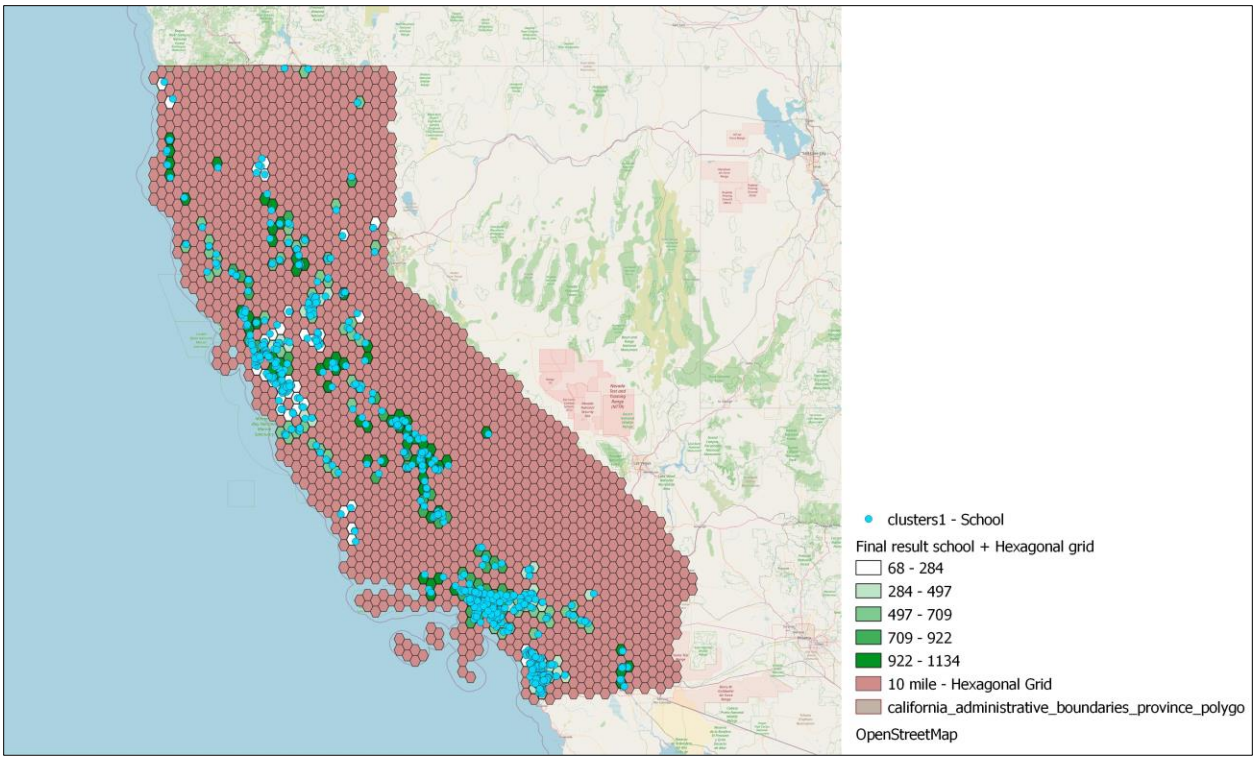


Figure 191: Indexing the California state (Coloring the parcels (Hexagonal grid))- when school is as origin point – joined layer

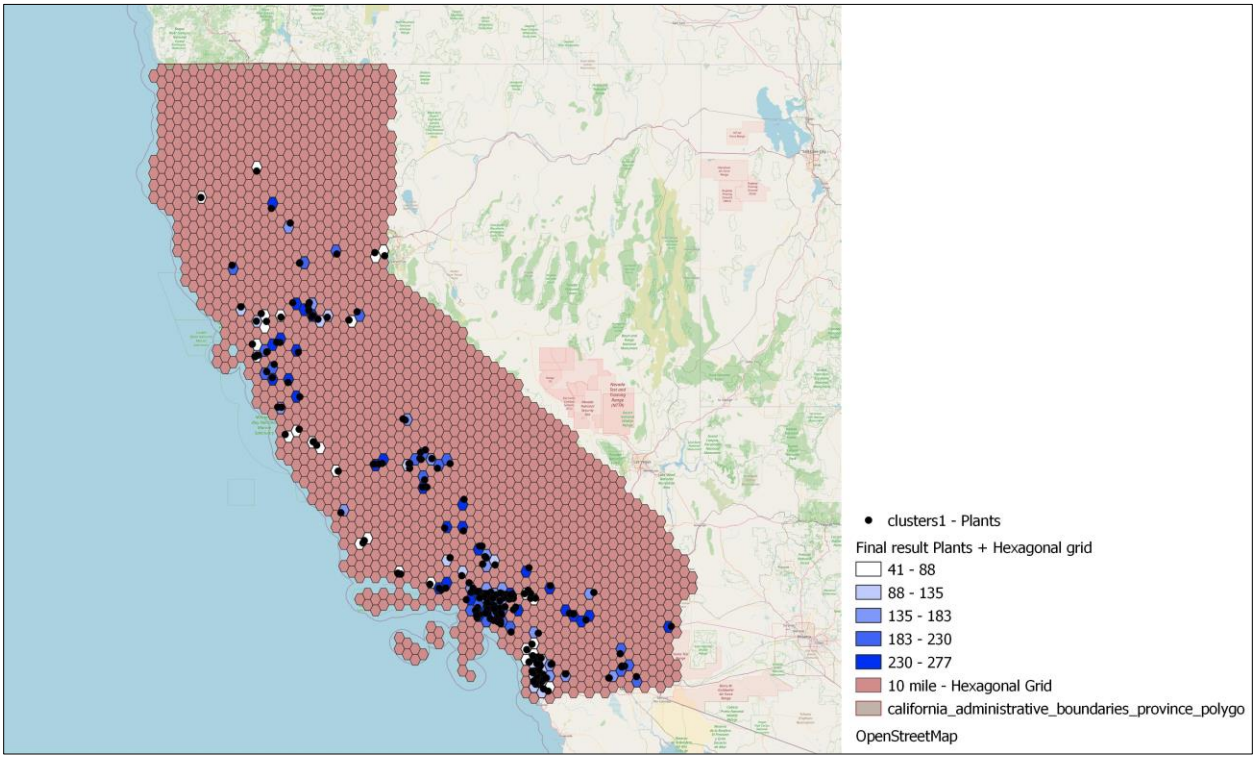


Figure 192: Indexing the California state (Coloring the parcels (Hexagonal grid))- when the plant is as origin point – joined layer

5.10 – UI DESIGN

A plugin with its user interface was built in QGIS software as a workspace of this computational system. To create this user interface, the GT designer was used. We can see the following images for creating and designing the plugin and its user interface.

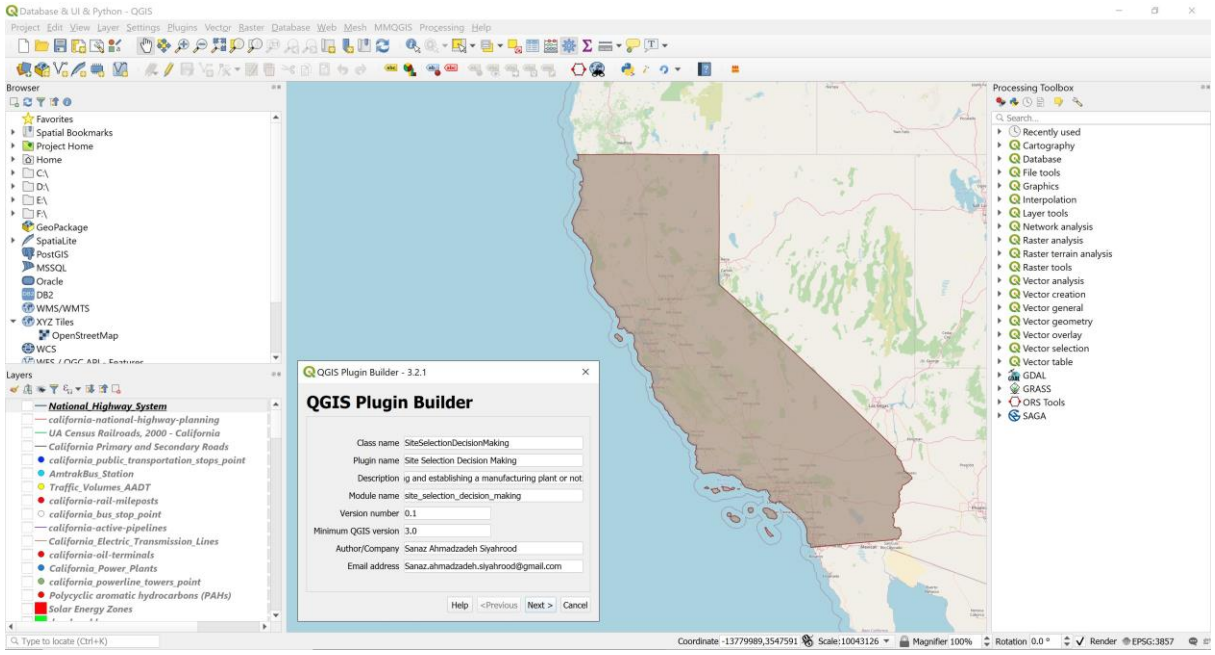


Figure 193: Building a plugin in QGIS using plugin builder – step 1

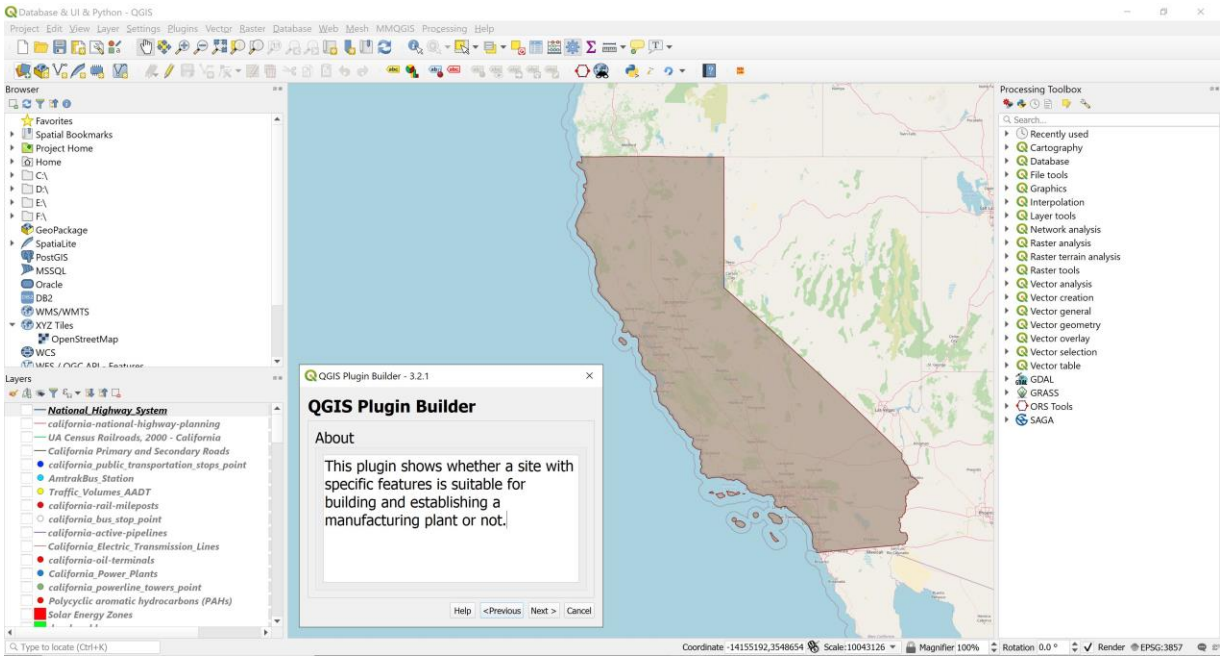


Figure 194: Building a plugin in QGIS using plugin builder – step 2

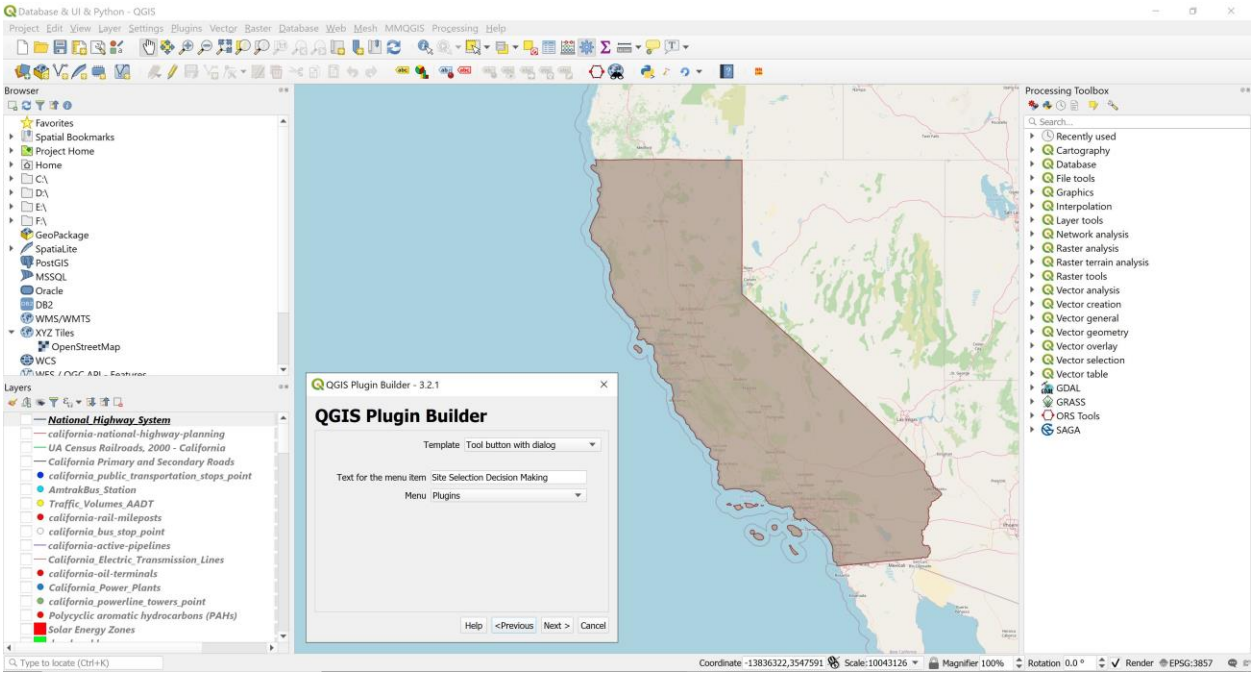


Figure 195: Building a plugin in QGIS using plugin builder – step 3

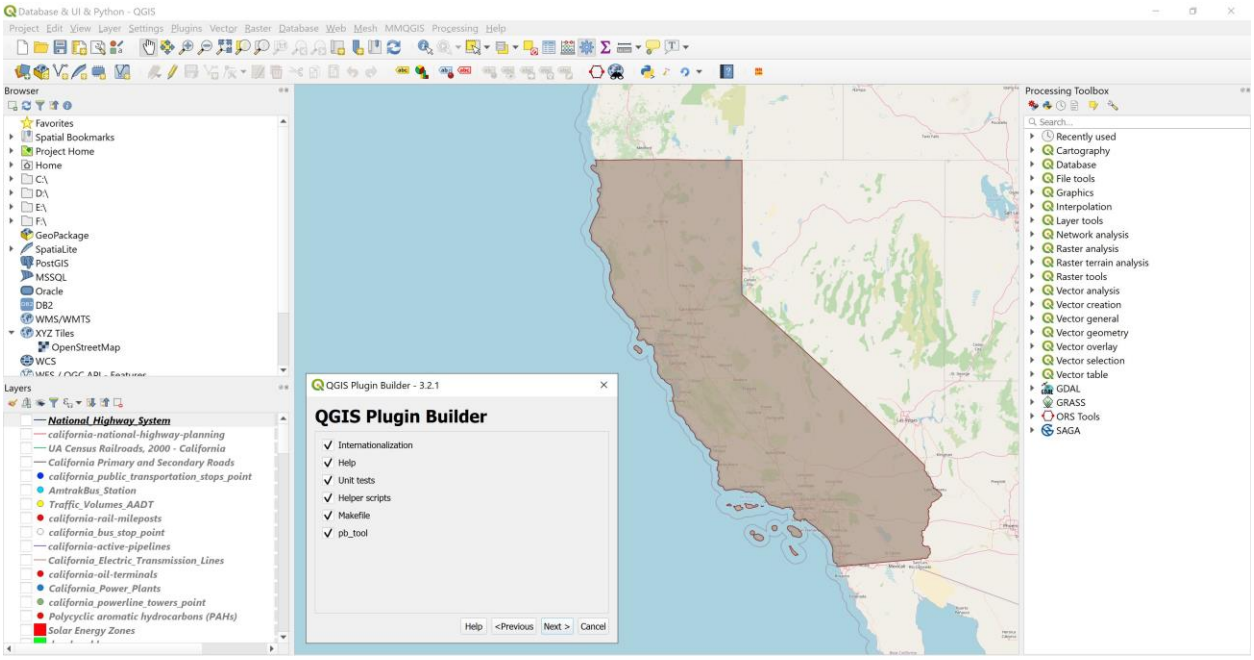


Figure 196: Building a plugin in QGIS using plugin builder – step 4

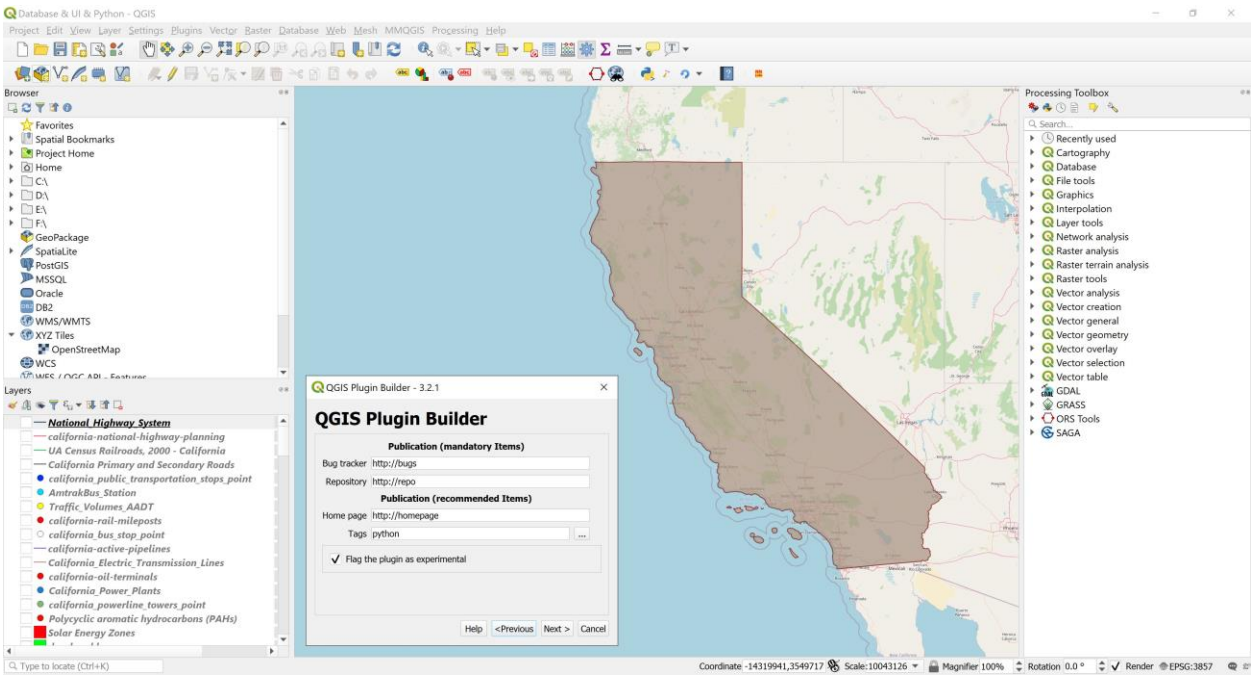


Figure 197: Building a plugin in QGIS using plugin builder – step 5

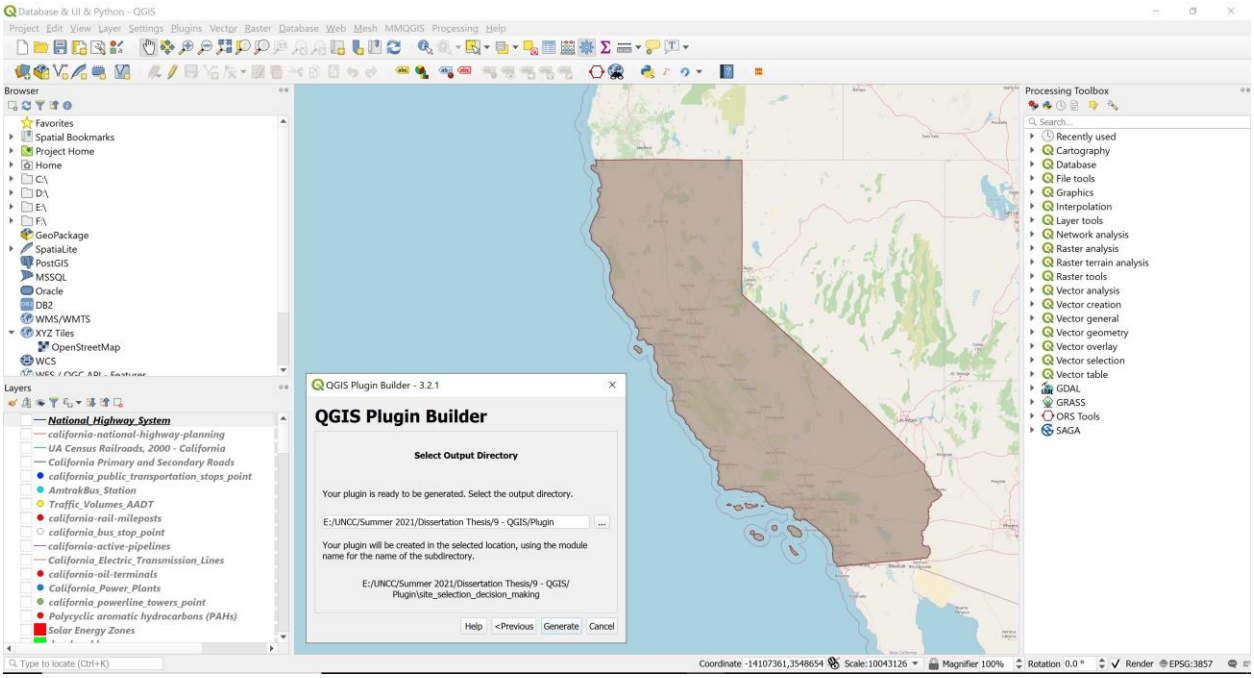


Figure 198: Building a plugin in QGIS using plugin builder – step 6

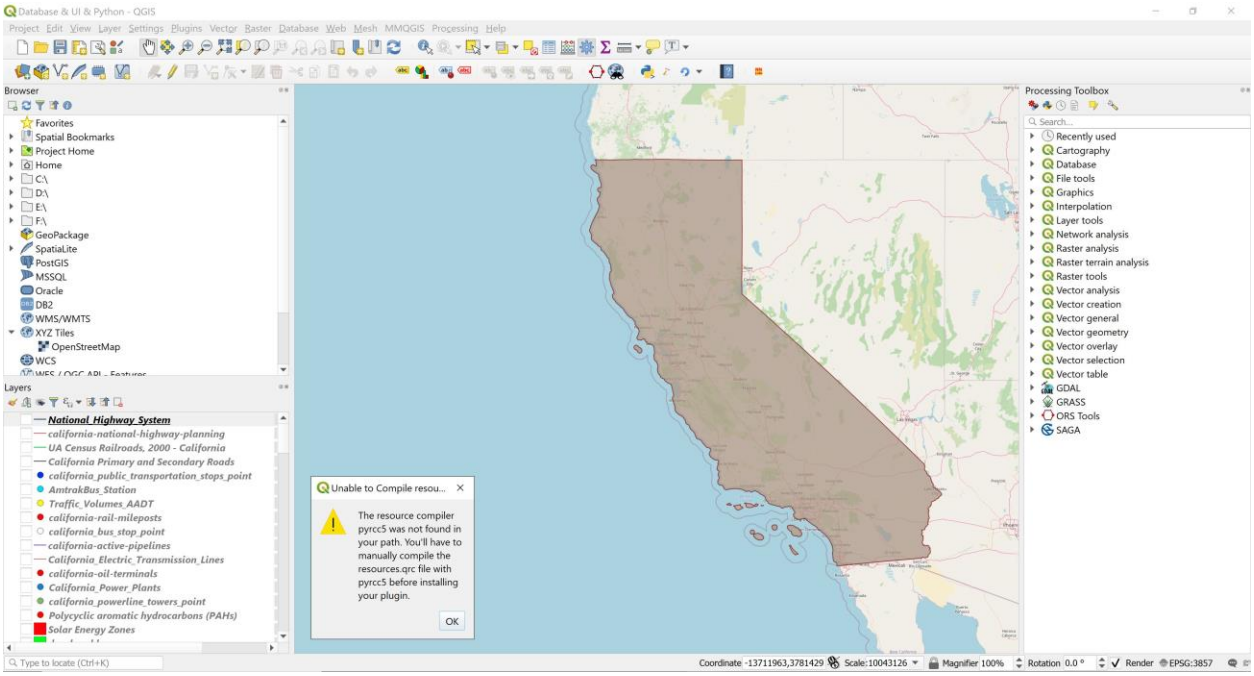


Figure 199: Building a plugin in QGIS using plugin builder – step 7

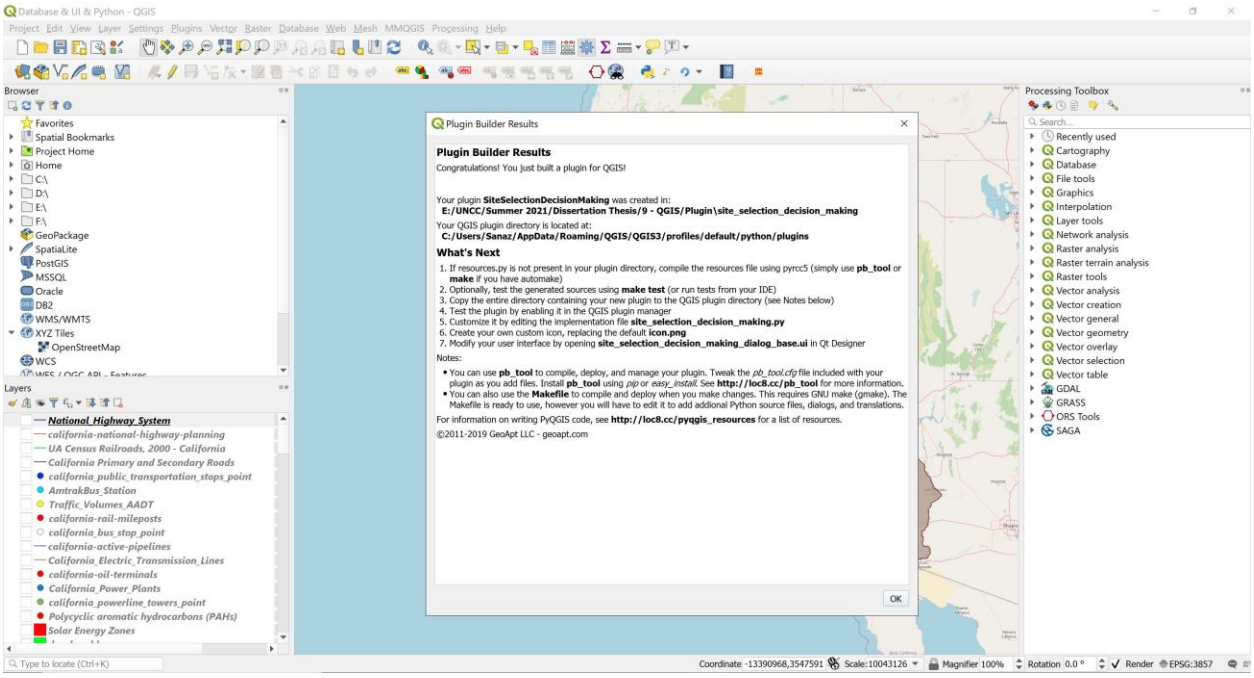


Figure 200: Building a plugin in QGIS using plugin builder – step 8

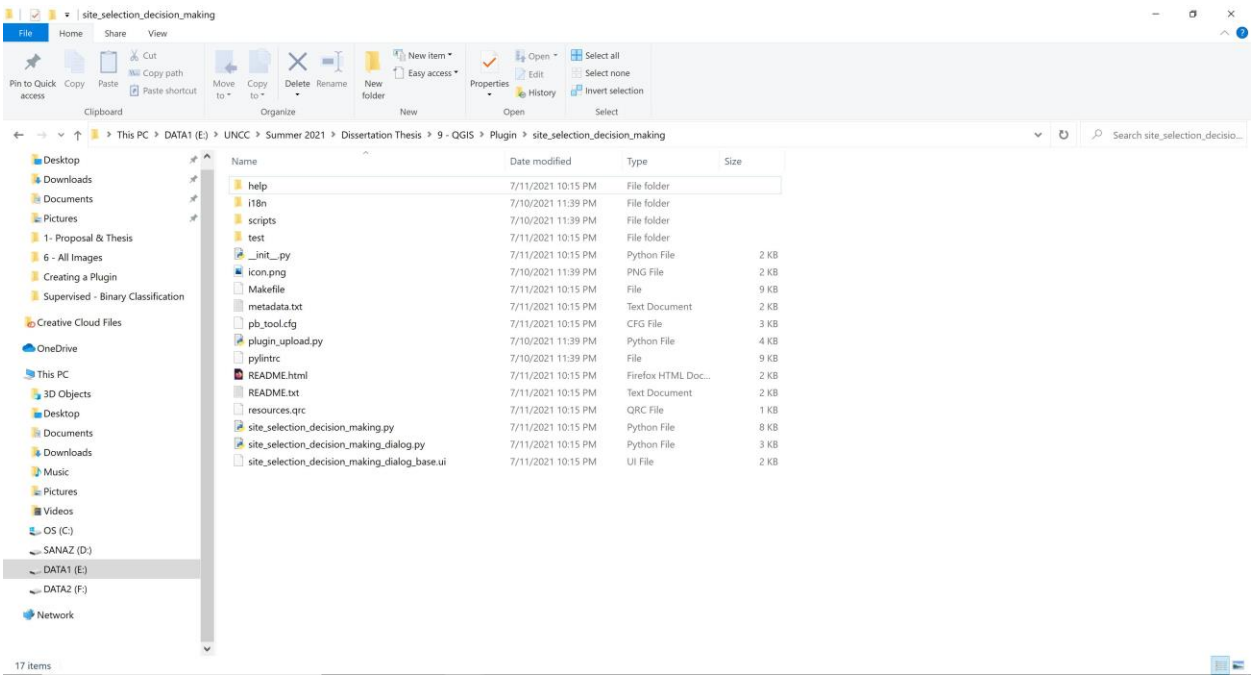


Figure 201: Building a plugin in QGIS using plugin builder – step 9

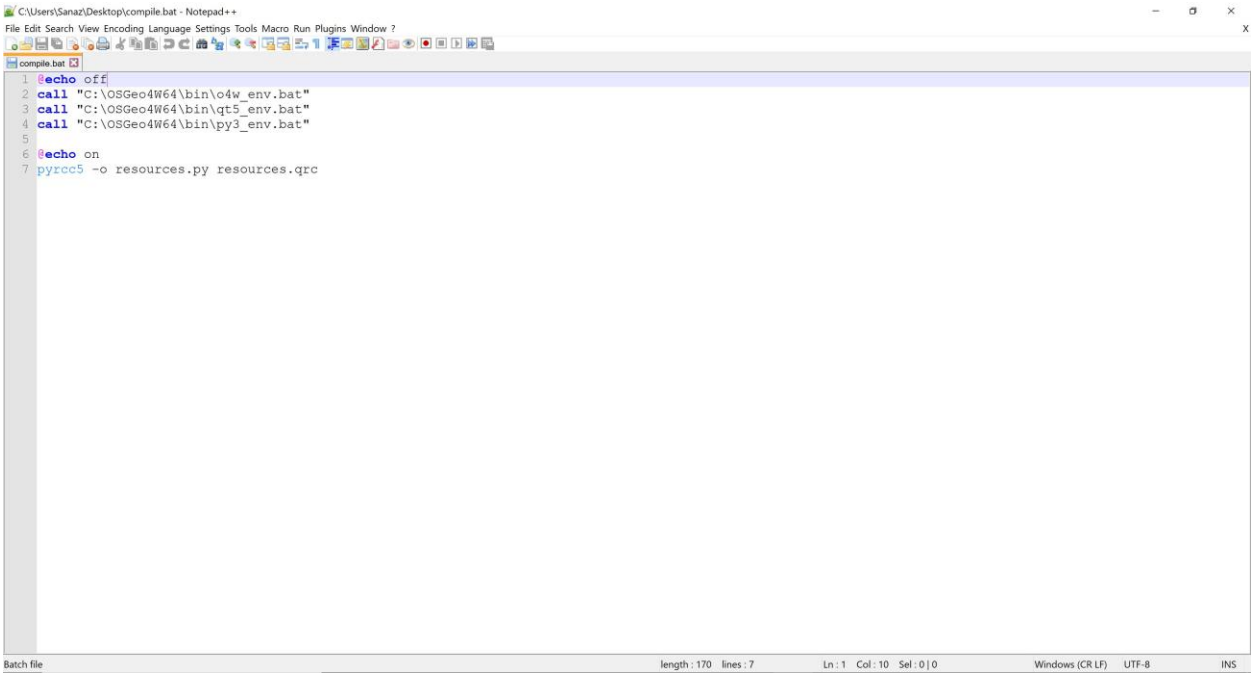


Figure 202: Building a plugin in QGIS using plugin builder – step 10 – compiling the plugin

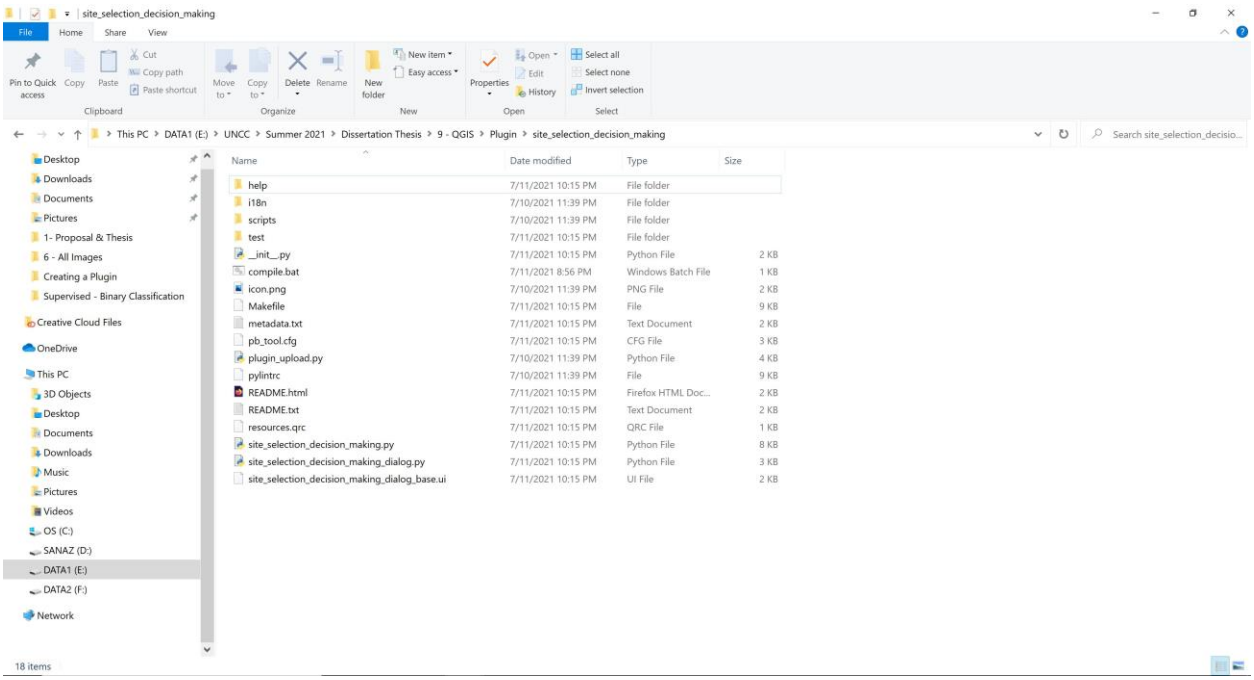


Figure 203: Building a plugin in QGIS using plugin builder – step 11

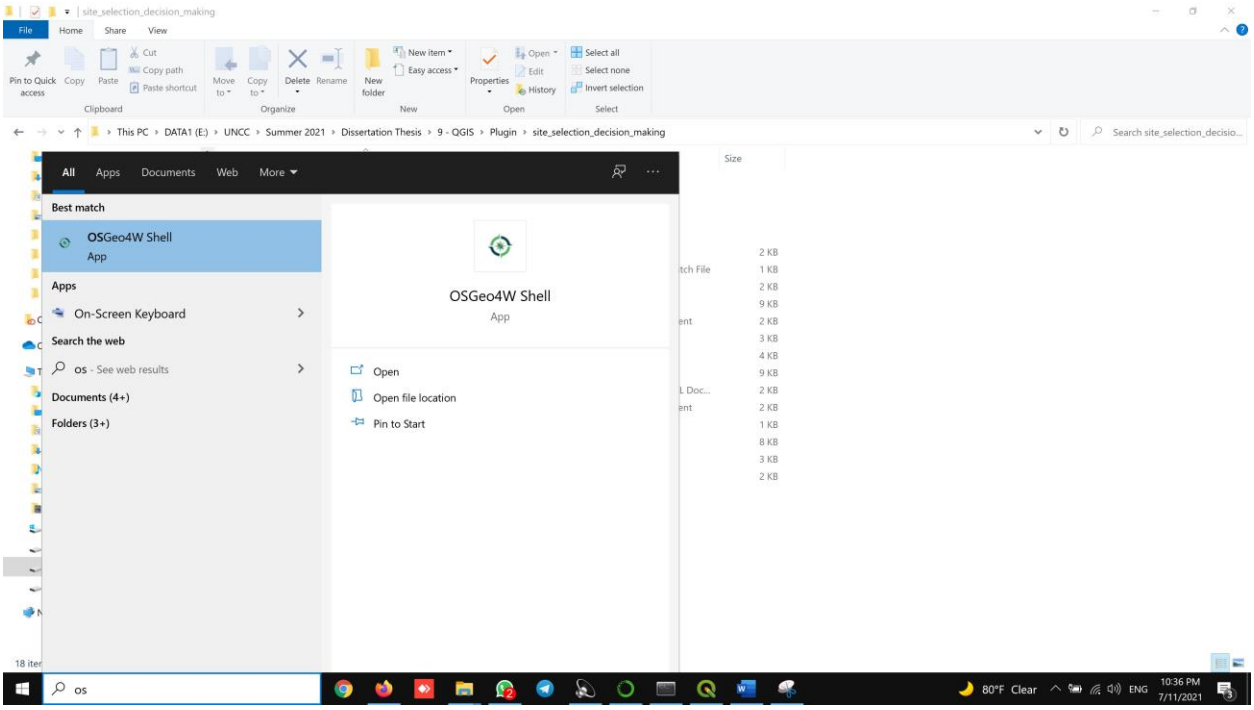


Figure 204: Building a plugin in QGIS using plugin builder – step 12

In this step with help of the QSGEO4W shell, the plugin compiled to the directory of QGIS.

The following steps were passed to completing the compiling.

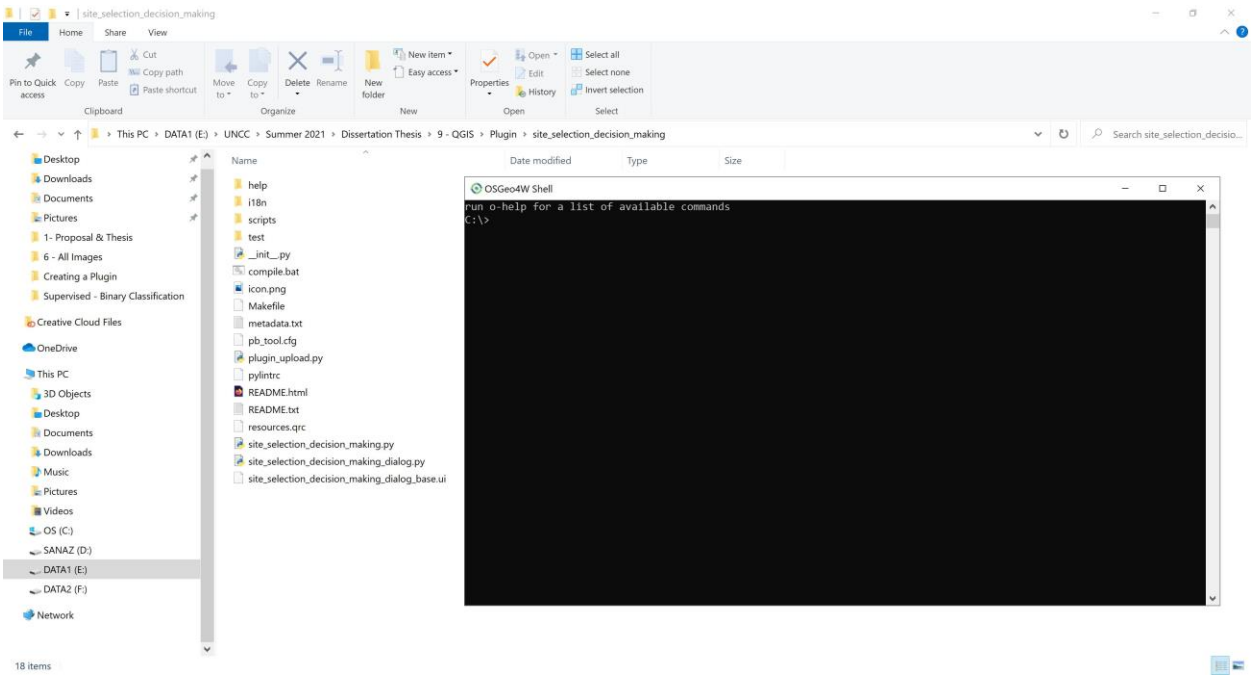


Figure 205: Building a plugin in QGIS using plugin builder – step 13

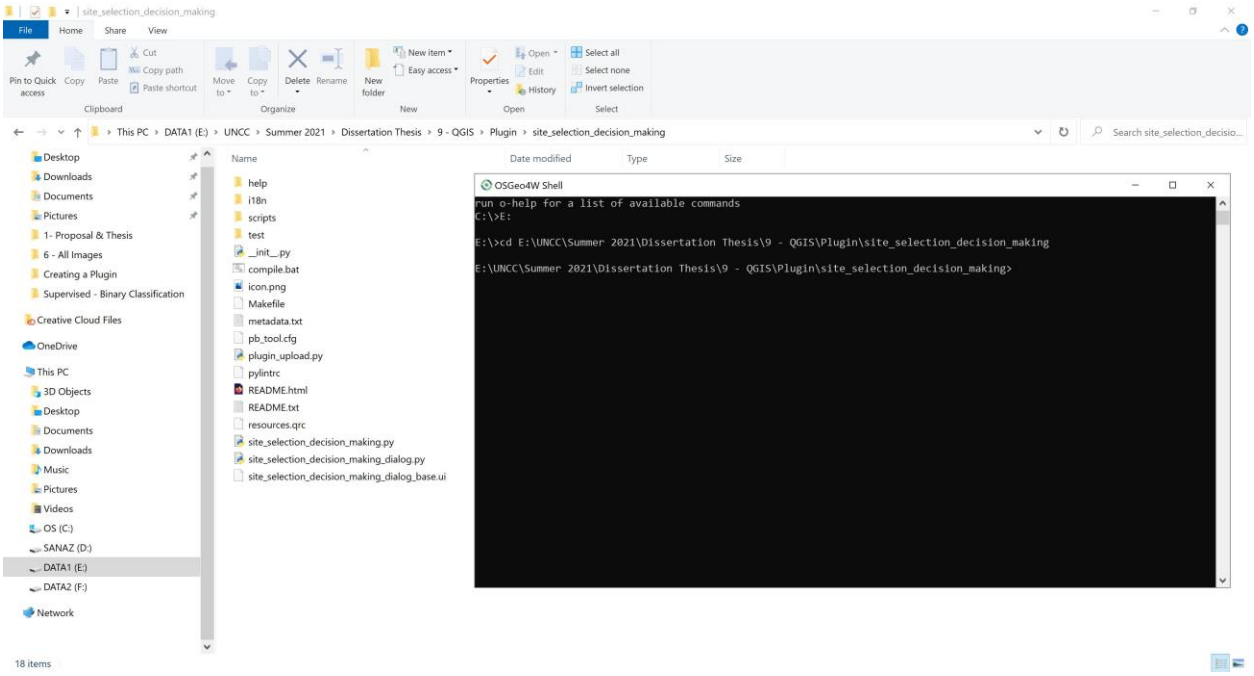


Figure 206: Building a plugin in QGIS using plugin builder – step 14

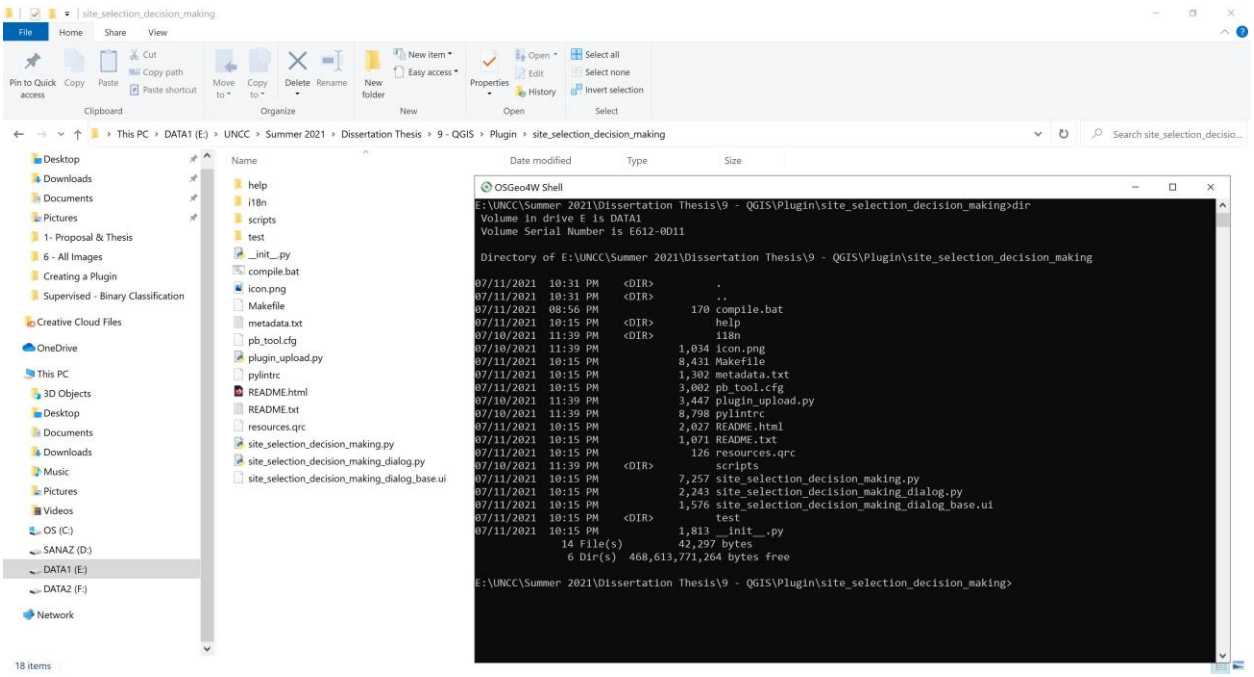


Figure 207: Building a plugin in QGIS using plugin builder – step 15

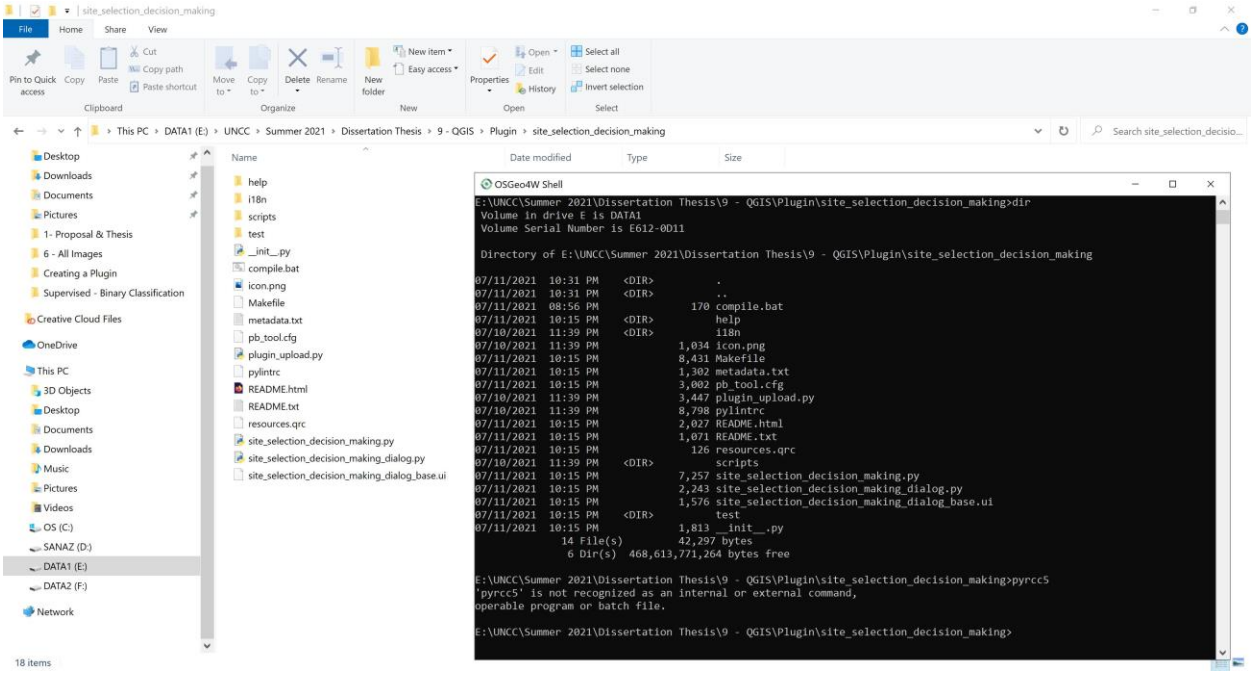


Figure 208: Building a plugin in QGIS using plugin builder – step 16

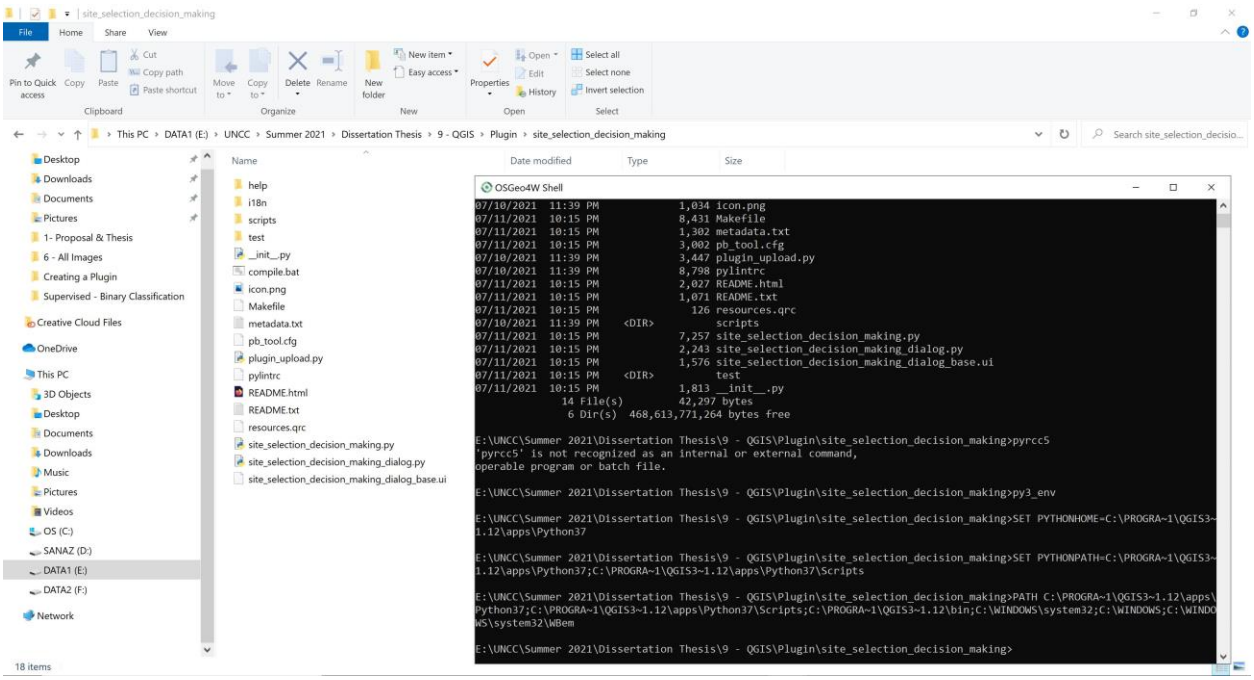


Figure 209: Building a plugin in QGIS using plugin builder – step 17

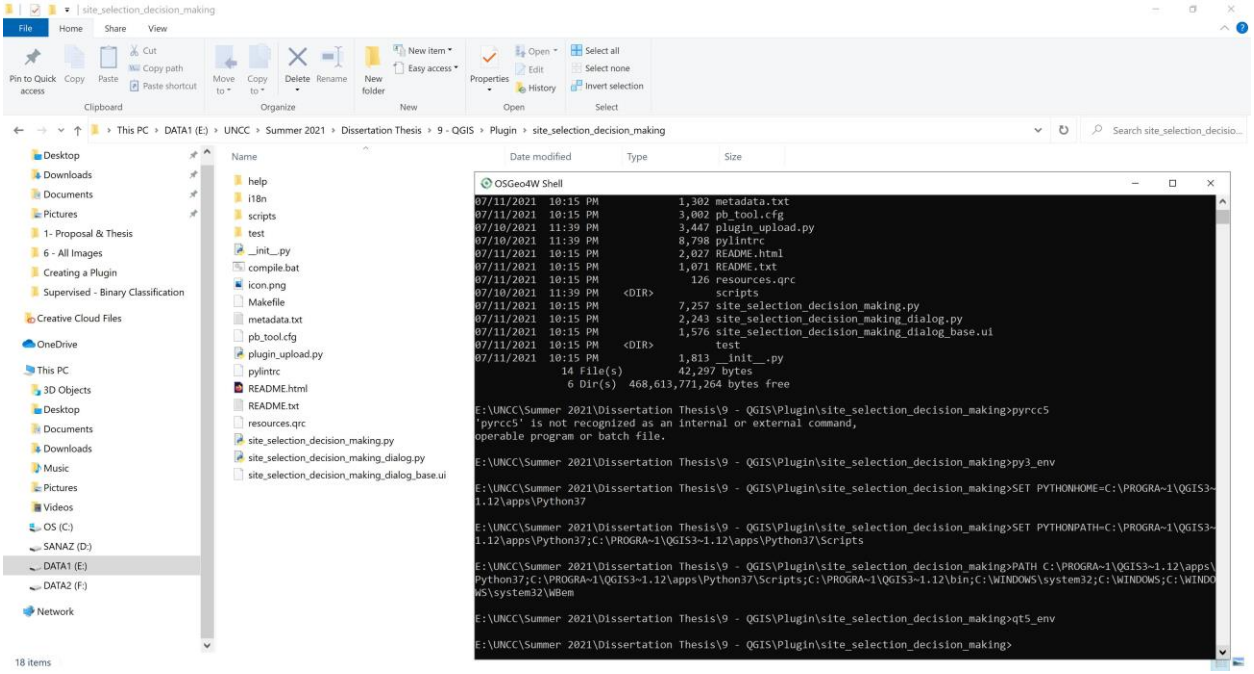


Figure 210: Building a plugin in QGIS using plugin builder – step 18

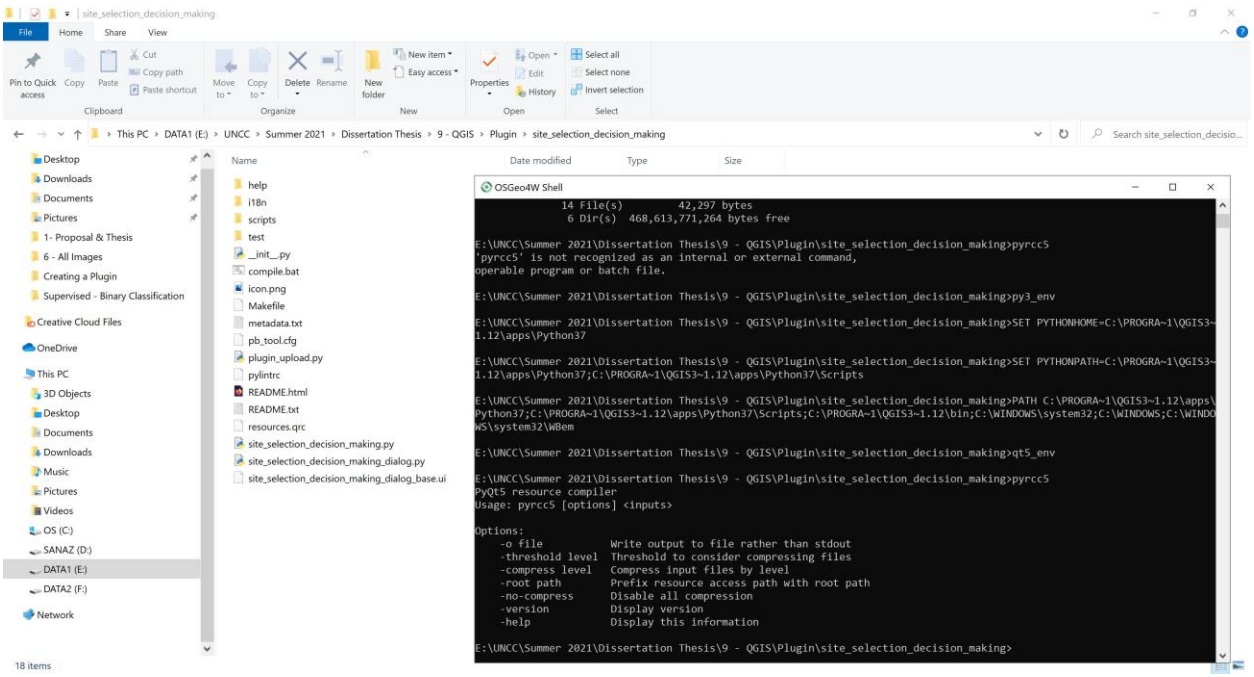


Figure 211: Building a plugin in QGIS using plugin builder – step 19

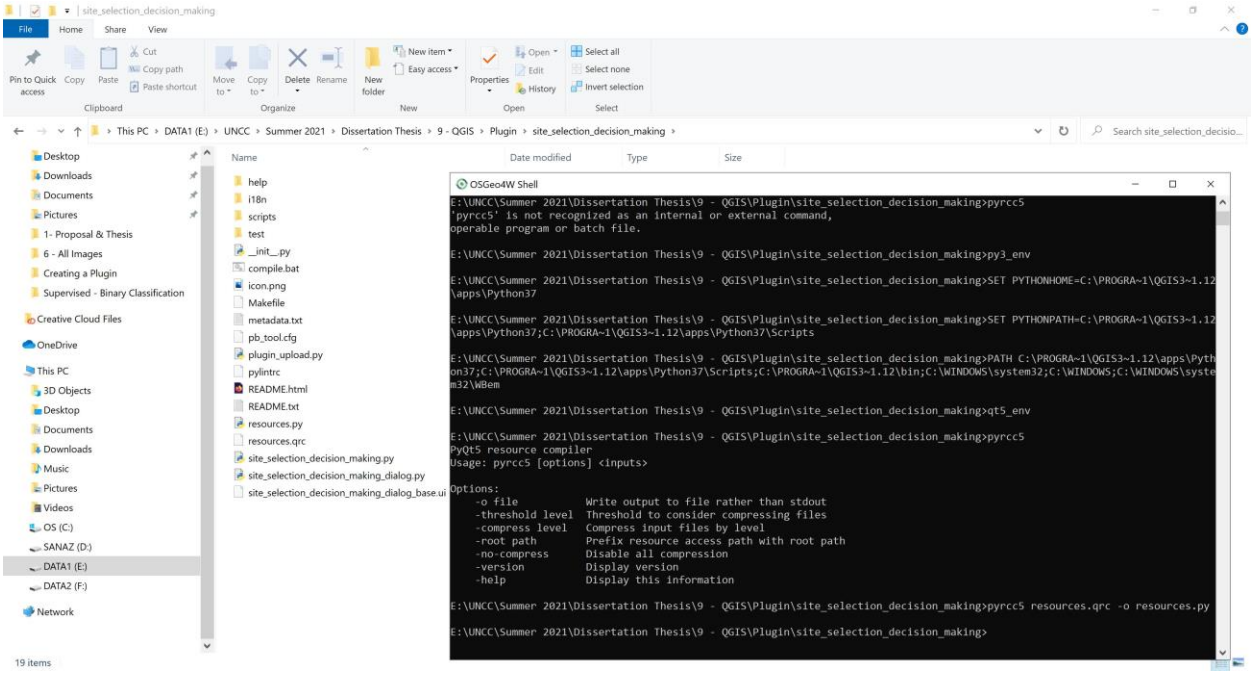


Figure 212: Building a plugin in QGIS using plugin builder – step 20

In image 212 we can see that resources.py has been created as a result of the successful compiling.

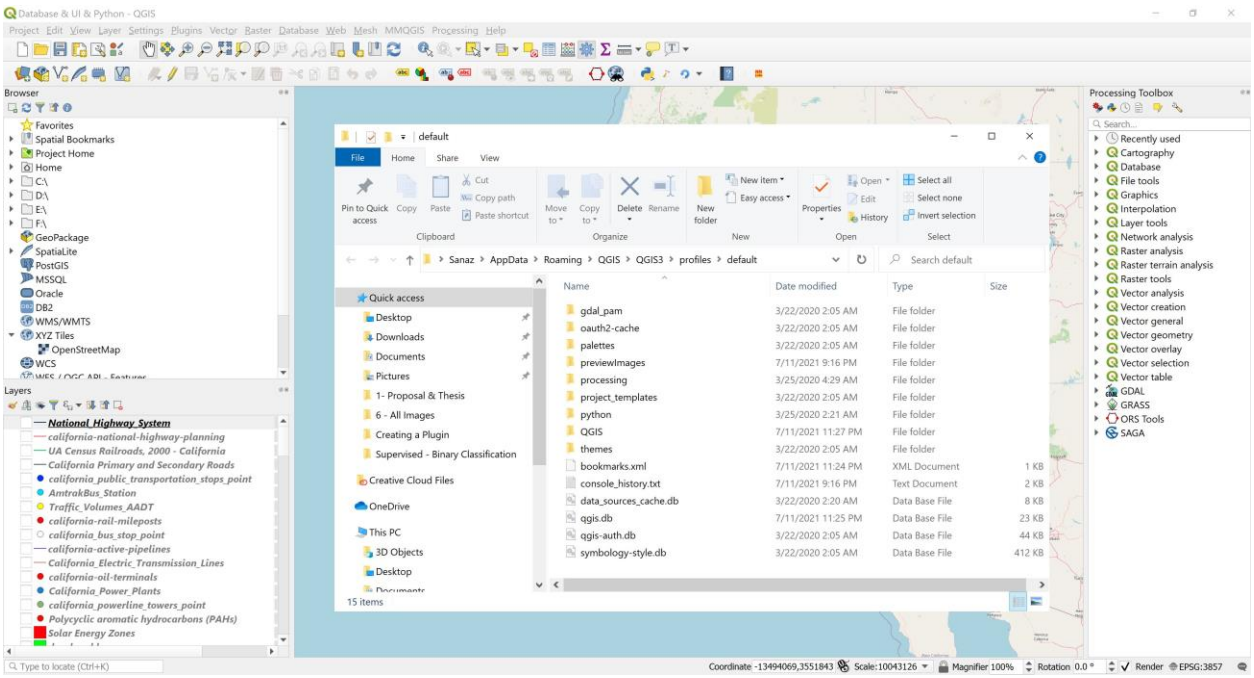


Figure 213: Building a plugin in QGIS using plugin builder – step 20 – Putting the plugin under the folder of all QGIS' plugins

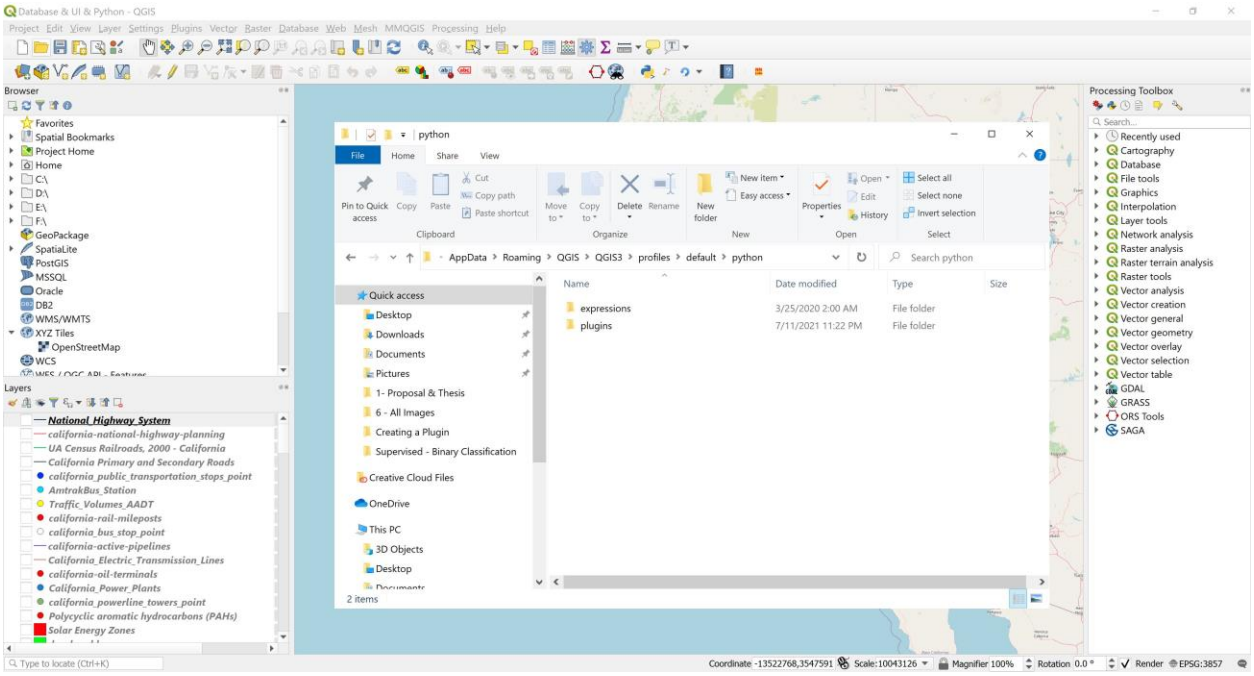


Figure 214: Building a plugin in QGIS using plugin builder – step 21

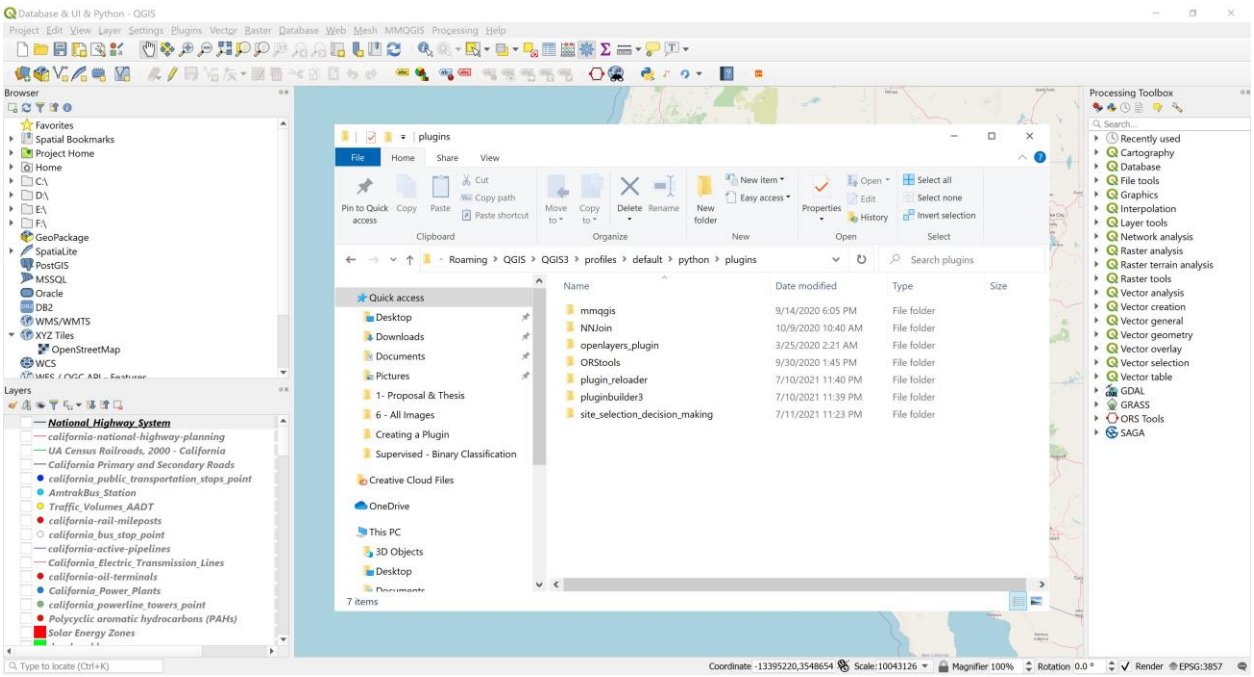


Figure 215: Building a plugin in QGIS using plugin builder – step 22 – site selection decision making

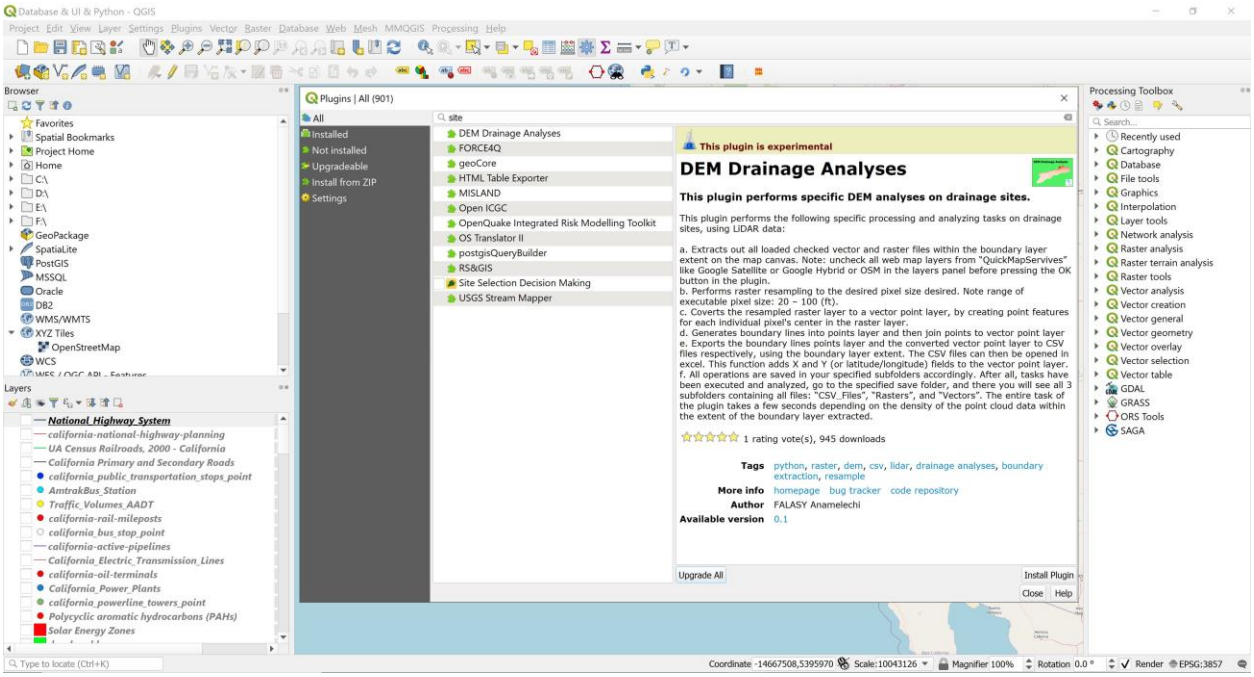


Figure 216: Building a plugin in QGIS using plugin builder – step 23 – Installing the site selection decision making in QGIS

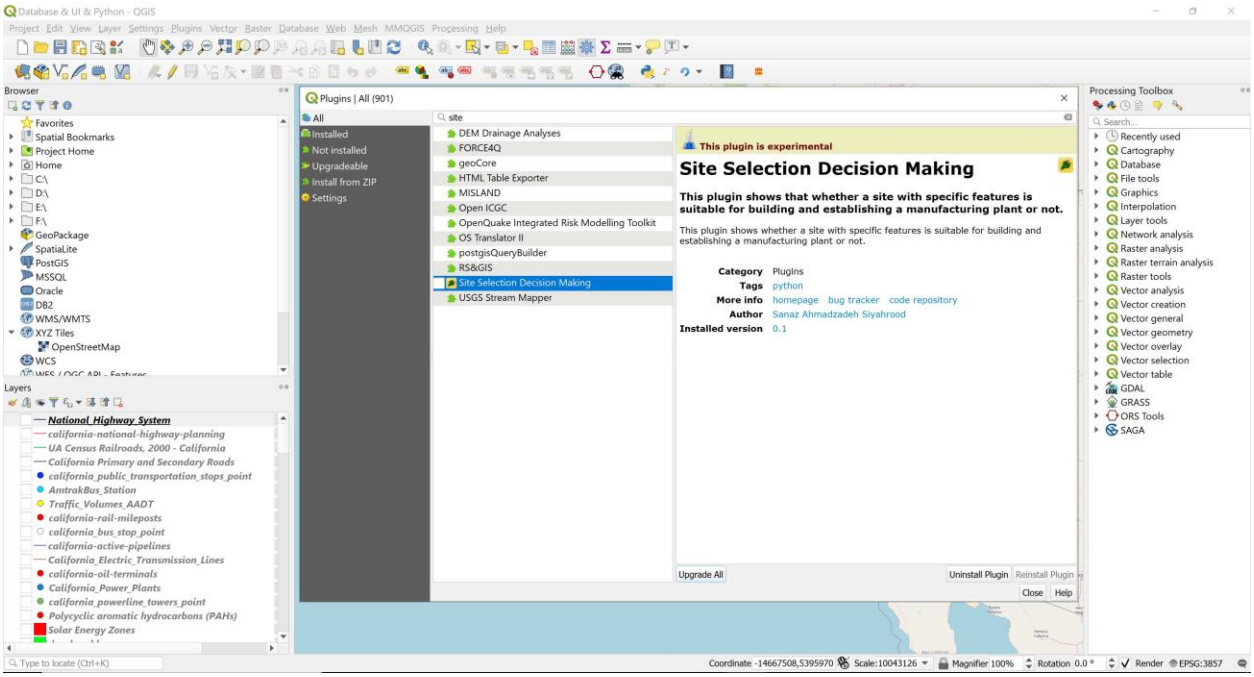


Figure 217: Building a plugin in QGIS using plugin builder – step 24 – Popping up the site selection decision making in QGIS

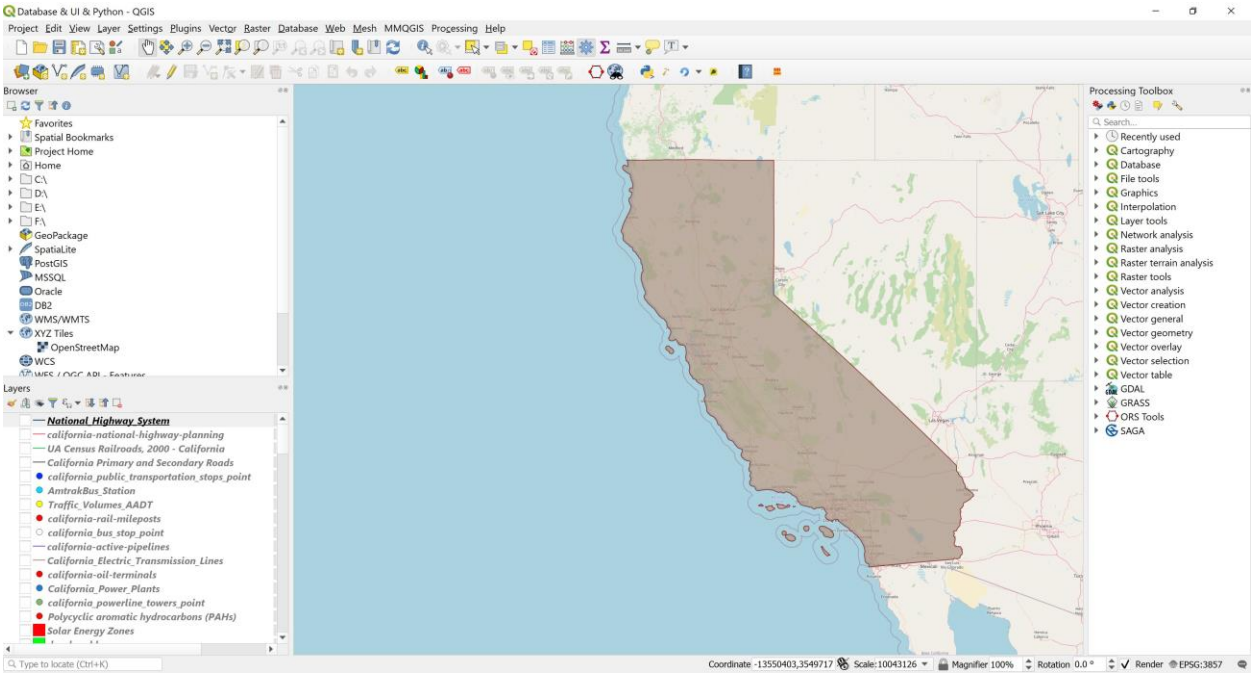


Figure 218: Building a plugin in QGIS using plugin builder – step 25 – Popping up the site selection decision making in QGIS

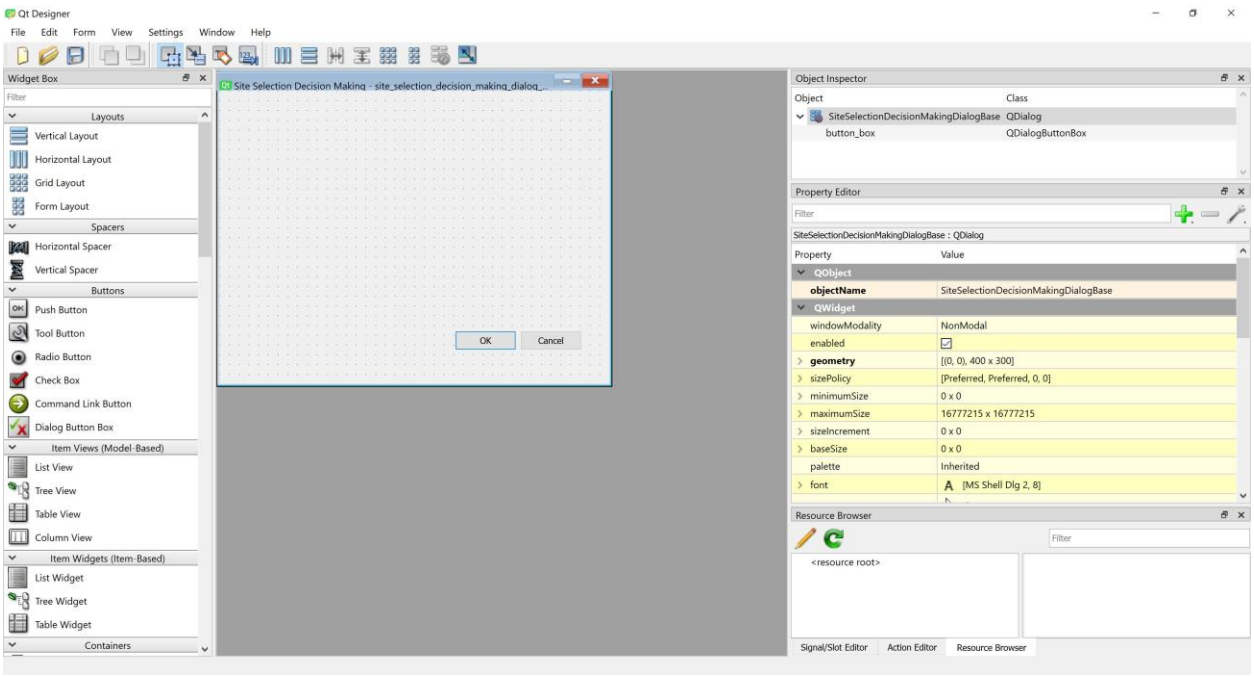


Figure 219: Building a plugin's user interface in GT designer– step 26 – Adding the features to the user interface

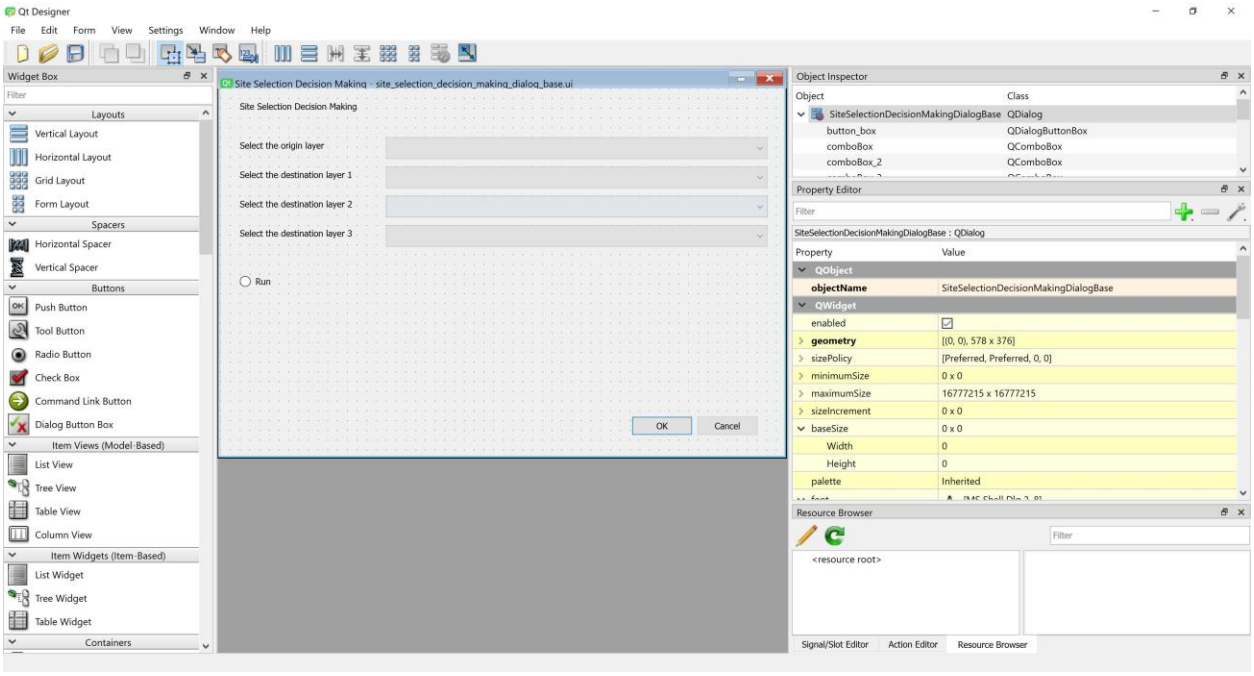


Figure 220: Building a plugin's user interface in GT designer– step 27 – Adding the features to the user interface

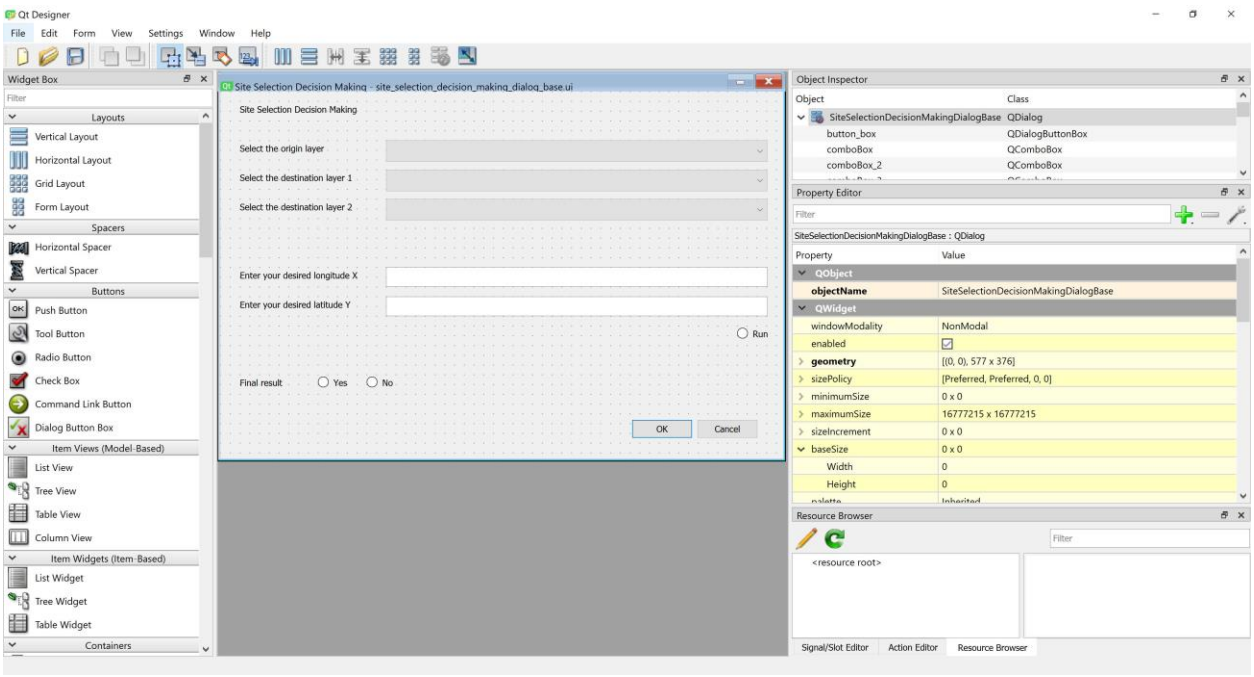


Figure 221: Building a plugin's user interface in GT designer– step 28 – Finalizing the user interface

After completing the design of the user interface, we started to gather all of the codes into the python code of the plugin. First, the required code for populating the fields of the plugin have been finalized, then task 1 and 2 and lastly machine learning algorithm were deployed.

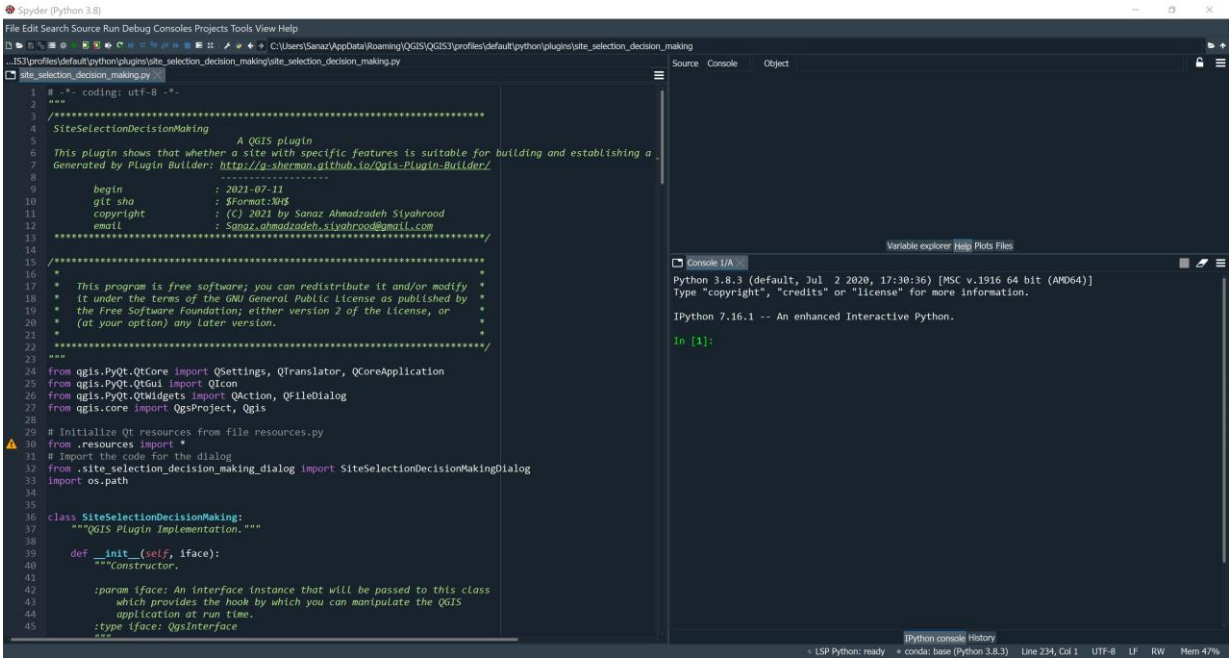


Figure 222:Python code development of the plugin

5.11 – BACKEND WORKFLOW OF ENTIRE SYSTEM

Based on all of the sections that were explained in detail before, the backend workflow of the entire system is as following:

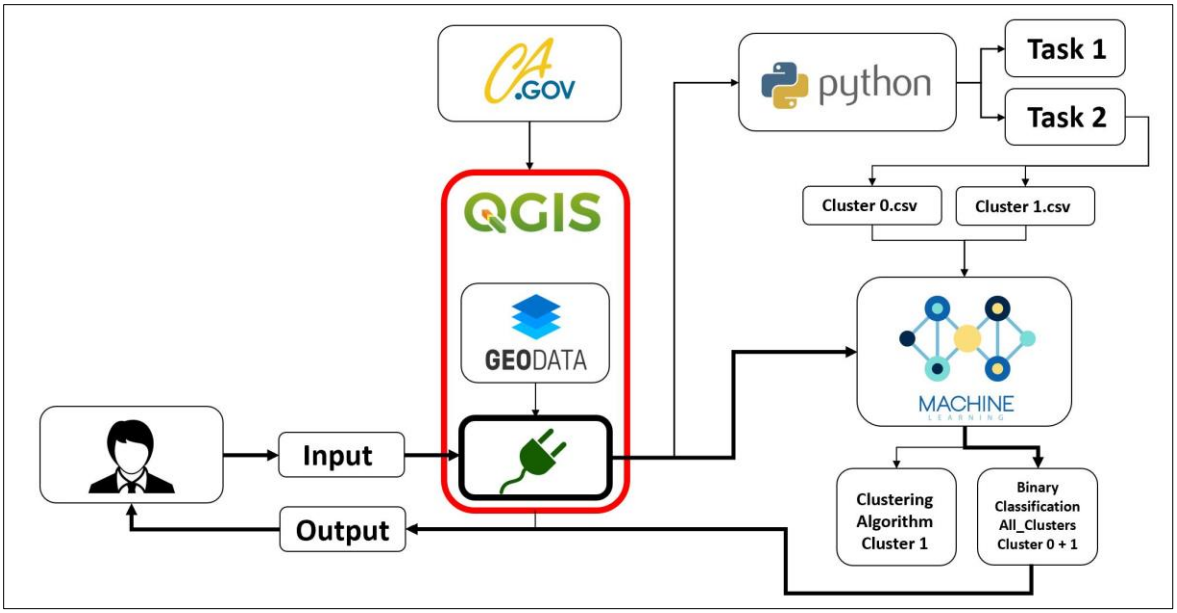


Figure 223: The backend workflow of the entire system

Also, the mechanism of the calculating entire system has been shown in the image below:

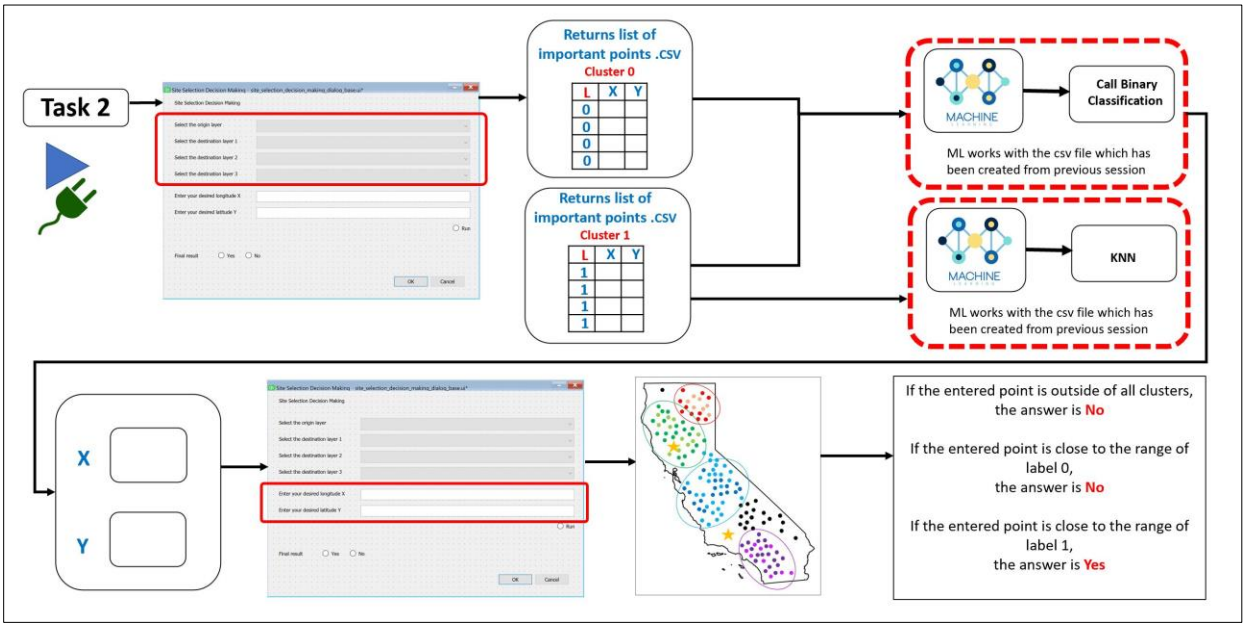


Figure 224: Mechanism of the calculating of the entire system for returning the final result

SECTION 6 : DISCUSSIONS AND FUTURE WORK

According to all analysis in previous sections especially in implementing task 1 and task 2 in Python programming and also machine learning algorithm, it can be concluded that with help of comparing all models of machine learning that can be applied for each specific dataset and also different types of plotting that can show several details of each model, we can select the optimum model as a finalized model to predict on test and unseen data. By changing the origin layer, all results change in each step however, as all layers are in the same format and they show the distribution of several points across the California State, so selected model would be the same whether the school layer is as an origin point or plant is as an origin point. In both tries, KNN due to covering several aspects of the analysis can be one of the logical algorithms for predicting unseen data. It is not a good decision to finalizing the model just after comparing all models since as we saw in all analysis, a model became final when we were able to check different types of plots for it. And it means that we should check several parameters for selecting the best model. This is important because a model may have good accuracy, but other parameters in it may not show the right results. Therefore, considering all the factors and after comparing different types of plots, we choose the final model. Hence, KNN had the best performance among the other 3 models (Decision tree, Random Forest, gradient boosting machine) because its results in different plots such as AUC plot, precision-recall, feature importance, confusion matrix, Calibration curves, validation curve are more precise than other models.

Based on the discussion in my final defense and also the potential of this work to continue and the comments which I received, for the future work of this study, I have the plan to add the following sections for expanding the different parts of this plugin; site selection decision making.

1- I will show the results of each analysis which change based on the selection of different layers as an origin point, this option gives the user the chance of comparison to figure out which result is more in line with their wishes and plans and can cover most of their requirements and criteria.

2- As we mentioned before, in this thesis only the point format of the GIS information has been used for analysis and calculations. To get closer to reality in calculations, Also, I have the plan to define a function that can automatically transfer one type of data to other types so that we can use all the other types of data in our dataset. For example, transferring polyline or polygon format to point format and vice versa.

3- All parameters and factors (layouts) which are involved in finding the potential location are categorized under 3 main sections including.

    1- Human factor
    2- Physical & Environmental Factor
    3- Economic factor

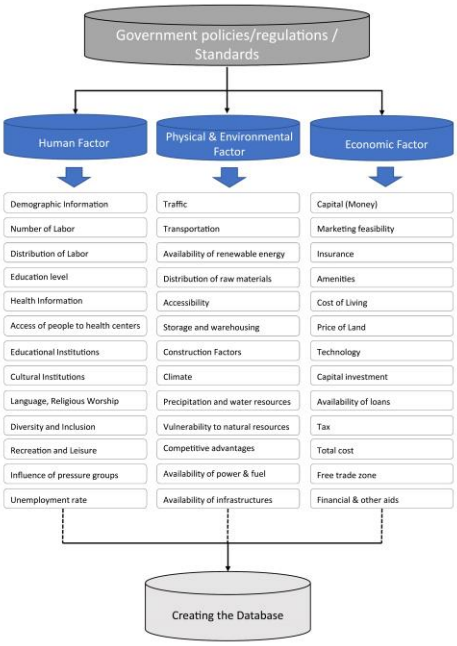


Figure 225: Schematic diagram of California's database

In the future development of this system, 4 levels can be considered to analyze the parameters. At each level factors and parameters are weighted according to their importance in finding the best solution.

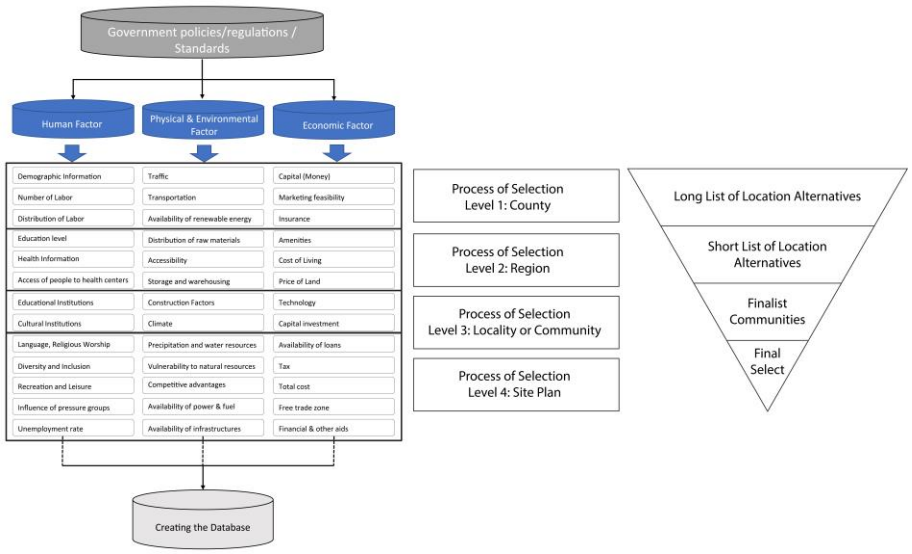


Figure 226: Internal analysis which will be done based on the different layouts of the database.

To reach an accurate result or correct answer, it is recommended that the smart selection first selects a country, a region out of the country, locality out of the region, and the exact site plan out of the chosen locality. Selecting a location out of a locality, a region, or a country should be based on a thorough review of relevant factors. Hence, selection depends upon the following factors. Image 226 can be explored through the explanation below:

1. Process of Selection – Level 1: County
2. Process of Selection – Level 2: Region
3. Process of Selection – Level 3: Locality or Community
4. Process of Selection – Level 4: Site

Considering the 3 main categories which were mentioned before, we can promote the economic impact (less cost), functional impact (higher productivity), public acceptance (better community relations), and quality of life (better living). For this study, only one or two layers were selected from the comprehensive database for the initial analysis.

4- After running the plugin, all the required methods and machine learning algorithms are recalled for returning the final results. By hitting the Ok button, and after completing the calculation with help of machine learning algorithms the user can see the final results in two ways; 0 or 1. If the plugin returns number 0 it means that the selected X and Y (selected location) cannot be a good candidate for establishing a new factory. On the other hand, if the plugin returns number 1 it means that the selected X and Y (selected location) can be a good candidate for establishing a new factory. And this issue is because 0 is referring to non-important points and 1 is referring to the important point. However, as we can measure the amount of each cluster in the distribution plot (see images 97 and 108), so we have a plan to add more sections in the final results indicating the percentage of the correct answer which is related to the amount of each cluster that has been created through KNN algorithm. Also, another factor can be used to measure the correctness of the answer and it is showing the score factor (see images 148, 184) when the final result returns. The model can give a score to each prediction on unseen data. This score is between 0 and 1. The closer the result is to 1, the more accurate the answer. So, we can show both these parameters when the final result return in UI (Plugin interface), therefore the user can better understand the answer that the plugin will display. The following image can show these goals.

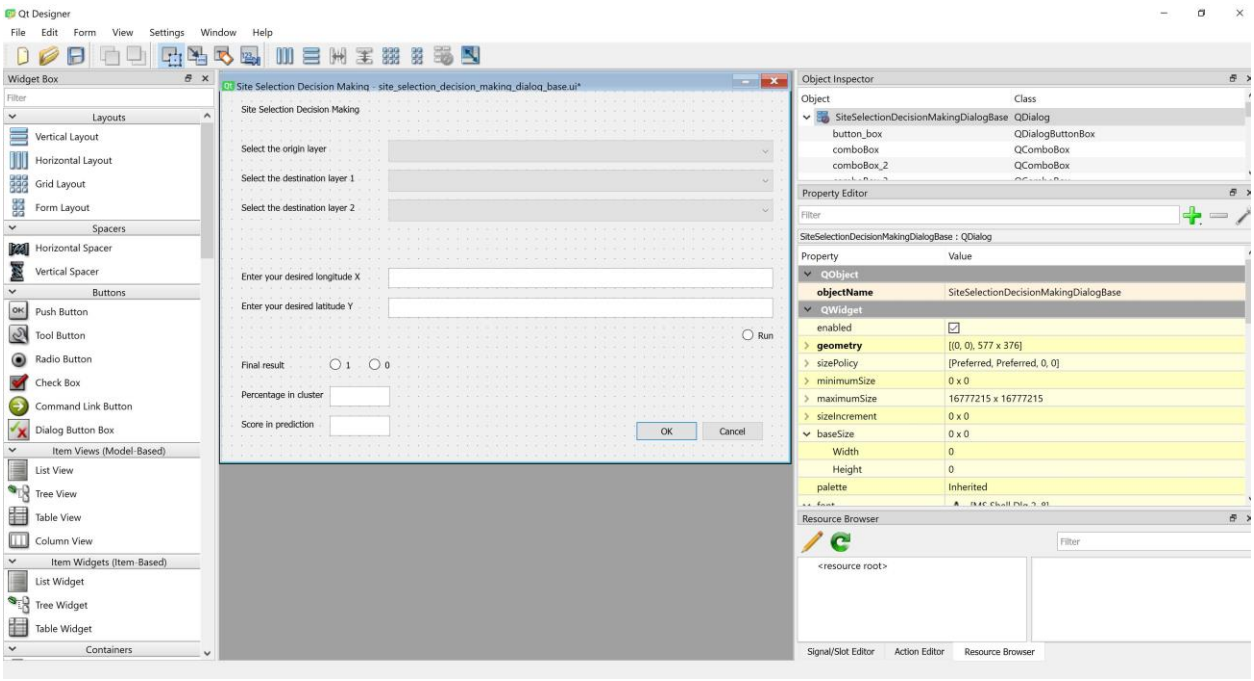


Figure 227: Displaying more information in the UI of the plugin when it returns the final result.

5- In this study, we considered a hexagonal mesh with a radius of 5 miles as a network on which we can show the final results. In this study, each of the hexagons is published as an urban parcel to be able to compare parcels in the final results. However, for the future development of this study, we have a plan to decrease the size of the calculation from all over California state to just a city or county in this state. In this way, we can better control the parameters and we can reach accurate and reliable results.

6- As the last discussion, we should mention that since the ultimate goal of this study is to be able to provide this plugin as an auxiliary tool for firms, companies, and institutions, so arrangements should be made in which all the conditions are provided for the practical use of a plugin. In this regard, the speed of calculations should be improved by methods, so that more factors can be considered to find the optimal answer.

SECTION 7 : CONCLUSIONS

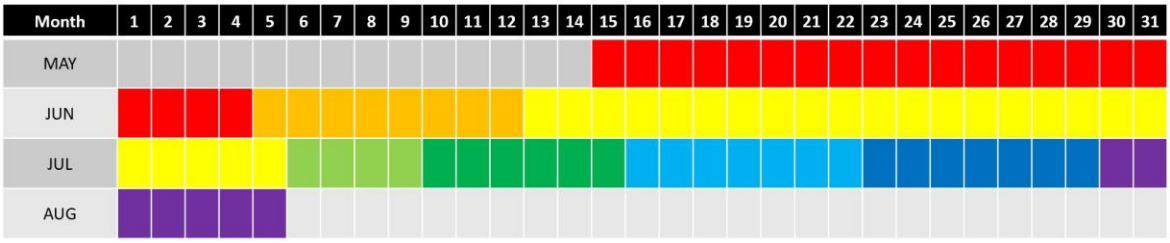
The main purpose of this study was to find a location or several candidates that have the potential for establishing a new factory by prioritizing the sites plans that are close to the places which have renewable energies. As the plant layer is including hundreds of the factories in California State that are using renewable energies, so we can say that they are some points that are close to our concept. Therefore, in two tries we did the whole of the process once manufacturing plants were as the origin point, and once the school layer was an origin point. This process showed that changing the origin layer can change the final result since by changing the origin layer we are changing the resources for designing the hubs. For doing this analysis two different tasks in the format of python codes were used. Also, the PyCaret library was implemented as a multifunctional library for covering the purpose of the machine learning parts. It is important to mention that in advance level of working with Pycaret library we can pass more details inside each method to reach the exact answer to our question. This study needs more details in the future to go to the depth of calculation and all these tries are because of the boosting the accuracy of the result. Although there are lots of solutions for finding the optimum location for the specific purpose in this thesis, we tried to get a little closer to the answer with the help of the important logic (access to essential urban services and renewable energies), as well as machine learning algorithms. This is despite the fact that in previous studies, close to this subject, more classical methods have been used to find the answer to the question. Moreover, the California State can be one of the target site plans for testing our data due to the importance of this state for using renewable energies for a different purpose.

SECTION 8 : POINTS OF NOTE

8.1 – WORK PLAN AND TIMETABLE

Using the previous experiences, I had already gained, I started to work on my thesis in May 2021 based on the following timeline. The details of this timetable were as following:

Table 2: Timetable of the study

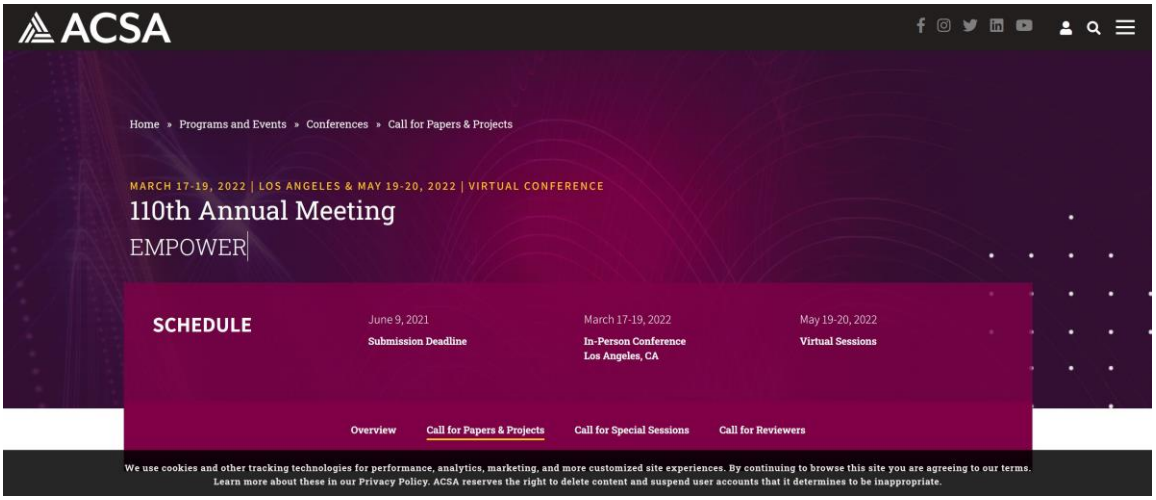


	• Topic selection • Doing some search around the topic • Research for literature review • Doing topic modeling • Collecting data • Writing the proposal
	• Finalizing data and creating the database • Building workflow
	• Starting coding
	• Starting machine learning section • Initial testing
	• Feedback and correction
	• Finalizing the workflow and output
	• Thesis and defense
	• Submission milestone

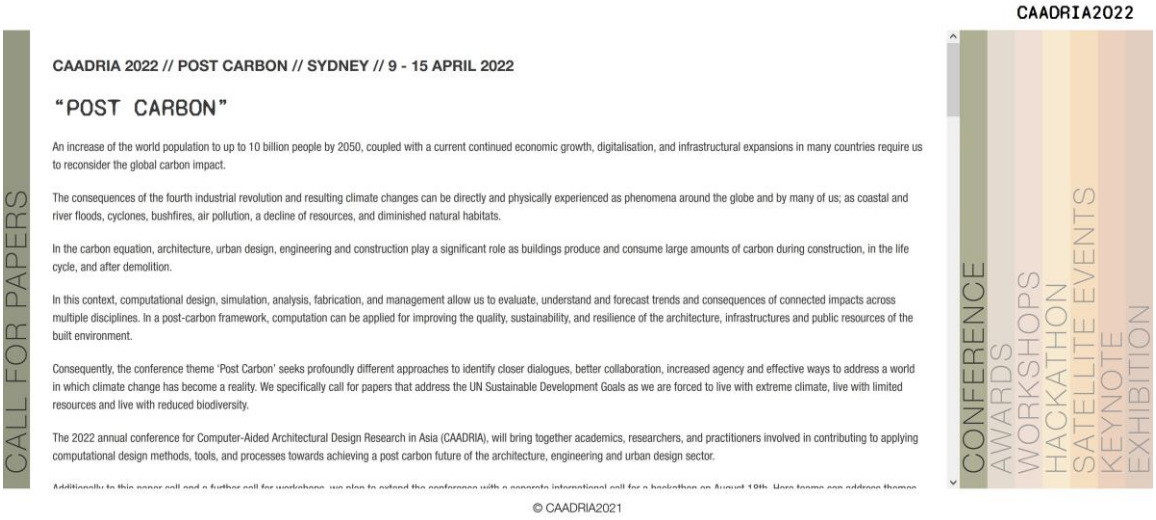
8.2 – OUTPUTS OF THIS STUDY – PUBLICATION

The abstract of the initial concept of this thesis was submitted at ACSA Conference which will hold MARCH 17-19, 2022, | in Los Angeles. For more information regarding the ACSA conference please refer to the following link. We are working on its full paper for submission.

https://www.acsa-arch.org/conference/110th-annual-meeting/call-for-abstracts-projects/



The abstract and full paper of this thesis is preparing to submit at the CAADRIA 2022 // POST CARBON // which will hold SYDNEY // 9 - 15 APRIL 2022. For more information regarding the CAADRIA 2022 conference please refer to the following link. https://caadria2022.org/



REFERENCES

1. Aboolian, R., O. Berman, and D. Krass, *Competitive facility location and design problem.* European Journal of Operational Research, 2007. **182**(1): p. 40-62.
2. Algharib, S.M., *Distance and coverage: an assessment of location-allocation models for fire stations in Kuwait City, Kuwait*. 2011, Kent State University.
3. Armour, G.C. and E.S.J.M.S. Buffa, *A heuristic algorithm and simulation approach to relative location of facilities.* 1963. **9**(2): p. 294-309.
4. Auty, R.M.J.E.G., *Scale economies and plant vintage: toward a factory classification.* 1975. **51**(2): p. 150-162.
5. Beck, R.L. and J.D.J.S.J.o.A.E. Goodin, *Optimum number and location of manufacturing milk plants to minimize marketing costs.* 1980. **12**(1378-2016-111144): p. 103-108.
6. Buckley, P.J. and R.J.A.o.M.P. Strange, *The governance of the global factory: Location and control of world economic activity.* 2015. **29**(2): p. 237-249.
7. Byun, D.-h. and E.J.J.o.M.S. Suh, *AHP model for selecting an automobile factory site'.* 1998. **7**: p. 15-30.
8. Cabot, A.V., R.L. Francis, and M.A.J.A.T. Stary, *A network flow solution to a rectilinear distance facility location problem.* 1970. **2**(2): p. 132-141.
9. Carsjens, G.J., A.J.L. Ligtenberg, and u. planning, *A GIS-based support tool for sustainable spatial planning in metropolitan areas.* 2007. **80**(1-2): p. 72-83.
10. Chang, P.-Y., H.-Y.J.J.o.I.E. Lin, and Management, *Manufacturing plant location selection in logistics network using Analytic Hierarchy Process.* 2015. **8**(5): p. 1547-1575.
11. Chou, T.-Y., C.-L. Hsu, and M.-C.J.I.j.o.h.m. Chen, *A fuzzy multi-criteria decision model for international tourist hotels location selection.* 2008. **27**(2): p. 293-301.
12. Church, R.L.J.G.i.s., *Location modelling and GIS.* 1999. **1**: p. 293-303.
13. Cradden, L., et al., *Multi-criteria site selection for offshore renewable energy platforms.* Renewable Energy, 2016. **87**: p. 791-806.
14. Deich, M. *State Taxes and Manufacturing Plant Location*. in *Proceedings of the Annual Conference on Taxation Held under the Auspices of the National Tax Association-Tax Institute of America*. 1989. JSTOR.
15. Deveci, M., I.Z. Akyurt, and S.J.J.o.E.I.M. Yavuz, *A GIS-based interval type-2 fuzzy set for public bread factory site selection.* 2018.
16. Efroymson, M.A. and T.L. Ray, *A Branch-Bound Algorithm for Plant Location.* Operations Research, 1966. **14**(3): p. 361-368.
17. Farahani, R.Z., et al., *Hub location problems: A review of models, classification, solution techniques, and applications.* Computers & Industrial Engineering, 2013. **64**(4): p. 1096-1109.
18. Feizizadeh, B., T.J.J.o.E.P. Blaschke, and Management, *Land suitability analysis for Tabriz County, Iran: a multi-criteria evaluation approach using GIS.* 2013. **56**(1): p. 1-23.
19. Florence, P.S. and W. Baldamus, *Investment, Location, and Size of Plant: A Realistic Inquiry Into the Structure of British and American Industries*. 1948: University Press.
20. Galvao, R.D., L.G.A. Espejo, and B.J.E.J.o.O.R. Boffey, *A hierarchical model for the location of perinatal facilities in the municipality of Rio de Janeiro.* 2002. **138**(3): p. 495-517.
21. Forsey, W.B., *Using location-allocation models to aid in the locating of preventive health care facilities for Newfoundland & Labrador*. 2014, Memorial University of Newfoundland.
22. Garetti, M., M.J.P.p. Taisch, and control, *Sustainable manufacturing: trends and research challenges*. 2012. **23**(2-3): p. 83-104.
23. Gothwal, S. and R. Saha, *Plant location selection of a manufacturing industry using analytic hierarchy process approach.* International Journal of Services and Operations Management, 2015. **22**(2): p. 235-255.
24. Hillsman, E.L.J.E. and P. A, *The p-median structure as a unified linear model for location—allocation analysis.* 1984. **16**(3): p. 305-318.
25. Holmes, T.J.J.J.o.p.E., *The effect of state policies on the location of manufacturing: Evidence from state borders.* 1998. **106**(4): p. 667-705.
26. Jagtap, R.S. and S.S. Mohanty, *Sustainable Manufacturing: Green Factory: A case study of a tool manufacturing company*. 2020.
27. Jelokhani-Niaraki, M. and J. Malczewski, *A group multicriteria spatial decision support system for parking site selection problem: A case study.* Land Use Policy, 2015. **42**: p. 492-508.

28. Jeppesen, T., J.A. List, and H.J.J.o.r.s. Folmer, *Environmental regulations and new plant location decisions: Evidence from a meta-analysis.* 2002. **42**(1): p. 19-49.
29. Kim, B.-J. and J.-H. Yom, *Legal information WPS for factory site location analysis.*
30. Lee, J.M. and Y.H. Lee, *Facility location and scale decision problem with customer preference.* Computers & Industrial Engineering, 2012. **63**(1): p. 184-191.
31. Levinson, A., *Environmental regulations and manufacturers' location choices: Evidence from the Census of Manufactures.* Journal of Public Economics, 1996. **62**(1): p. 5-29.
32. Lukoko, P., C.J.I.J.o.S.B. Mundia, and A. Research, *GIS based site suitability analysis for location of a sugar factory in trans Mara district.* 2016. **25**(3): p. 324-339.
33. Manatkar, R., et al., *An integrated inventory optimization model for facility location-allocation problem.* 2016. **54**(12): p. 3640-3658.
34. Matt, D.T., et al., *Urban production–A socially sustainable factory concept to overcome shortcomings of qualified workers in smart SMEs.* 2020. **139**: p. 105384.
35. Mcnamara, K. and W.P.J.T.R.o.R.S. Kriesel, *Manufacturing Location: the Impact of Human Capital Stocks and Flows.* 1988. **18**: p. 42-48.
36. McPherson, E.M., *Plant location selection techniques*. 1995: Elsevier.
37. Mitsuishi, M., K. Ueda, and F. Kimura, *Manufacturing systems and technologies for the new frontier*. 2008: Springer.
38. Moellmann, J. and V.M.J.S.-E.P.S. Thomas, *Social enterprise factory location and allocation model: Small scale manufacturing for East Africa.* 2019. **68**: p. 100694.
39. Muhsin, N., T. Ahamed, and R.J.A.-P.J.o.R.S. Noguchi, *GIS-based multi-criteria analysis modeling used to locate suitable sites for industries in suburban areas in Bangladesh to ensure the sustainability of agricultural lands.* 2018. **2**(1): p. 35-64.
40. Pasandideh, S.H.R. and S.T.A.J.J.o.I.M. Niaki, *Genetic application in a facility location problem with random demand within queuing framework.* 2012. **23**(3): p. 651-659.
41. Pellenbarg, P.H.J.J.o.E.P. and Management, *Sustainable business sites in the Netherlands: a survey of policies and experiences.* 2002. **45**(1): p. 59-84.
42. Rabbani, M., et al., *A hybrid robust possibilistic approach for a sustainable supply chain location-allocation network design.* 2020. **7**(1): p. 60-75.
43. Narula, S.C. and U.I.J.O. Ogbu, *An hierarchal location—allocation problem.* 1979. **7**(2): p. 137-143.
44. Revelle, C.S. and G. Laporte, *The Plant Location Problem: New Models and Research Prospects.* Operations Research, 1996. **44**(6): p. 864-874.
45. Schmenner, R.W., J.C. Huber, and R.L.J.J.o.U.E. Cook, *Geographic differences and the location of new manufacturing facilities.* 1987. **21**(1): p. 83-104.
46. Scott, P. and P.J.T.E.H.R. Walsh, *Patterns and determinants of manufacturing plant location in interwar London.* 2004. **57**(1): p. 109-141.
47. Shafiee-Gol, S., et al., *A mathematical model to design dynamic cellular manufacturing systems in multiple plants with production planning and location–allocation decisions.* 2021. **25**(5): p. 3931-3954.
48. Sheppard, E.S.J.E. and P. a, *A conceptual framework for dynamic location—allocation analysis.* 1974. **6**(5): p. 547-564.
49. Sherali, A.D. and C.J.A.T. Shetty, *The rectilinear distance location-allocation problem.* 1977. **9**(2): p. 136-143.
50. Sihag, N., et al., *The influence of manufacturing plant site selection on environmental impact of machining processes.* 2019. **80**: p. 186-191.
51. Smith, E.D., B.J. Deaton, and D.R. Kelch, *Location determinants of manufacturing industry in rural areas.* 1978.
52. Stanley, J.A., C.A. Radford, and A.G.J.P.o.t.R.S.B.B.S. Jeffs, *Location, location, location: finding a suitable home among the noise.* 2012. **279**(1742): p. 3622-3631.
53. Wu, B., B.R. Sarker, and K.P.J.A.E. Paudel, *Sustainable energy from biomass: Biomethane manufacturing plant location and distribution problem.* 2015. **158**: p. 597-608.
54. Wesolowsky, G.O.J.M.S., *Dynamic facility location.* 1973. **19**(11): p. 1241-1248.
55. Kumari, R. and S.K.J.I.J.o.C.A. Srivastava, *Machine learning: A review on binary classification.* 2017. **160**(7).
56. Gain, U. and V. Hotti. *Low-code AutoML-augmented Data Pipeline–A Review and Experiments*. in *Journal of Physics: Conference Series*. 2021. IOP Publishing.

57. Yoon, K. and C.-L.J.I.J.o.P.R. Hwang, *Manufacturing plant location analysis by multiple attribute decision making: Part I—single-plant strategy.* 1985. **23**(2): p. 345-359.
58. Walker, R., R. Walker, and F.J.R.S. Calzonetti, *Searching for new manufacturing plant locations: a study of location decisions in Central Appalachia.* 1990. **24**(1): p. 15-30.
59. Shi, C., et al., *A quantitative discriminant method of elbow point for the optimal number of clusters in clustering algorithm.* 2021. **2021**(1): p. 1-16.
60. Gentleman, R. and V.J. Carey, *Unsupervised machine learning*, in *Bioconductor case studies*. 2008, Springer. p. 137-157.
61. McGregor, A., et al. *Flow clustering using machine learning techniques*. in *International workshop on passive and active network measurement*. 2004. Springer.
62. Visa, S., et al., *Confusion matrix-based feature selection.* 2011. **710**: p. 120-127.
63. Davis, J. and M. Goadrich. *The relationship between Precision-Recall and ROC curves*. in *Proceedings of the 23rd international conference on Machine learning*. 2006.
64. Boyd, K., et al. *Unachievable region in precision-recall space and its effect on empirical evaluation*. in *Proceedings of the... International Conference on Machine Learning. International Conference on Machine Learning*. 2012. NIH Public Access.

65. Websites for collecting data – Open Sources Data

https://data.cnra.ca.gov/
https://data.edd.ca.gov/
https://data.ca.gov/
https://data.chhs.ca.gov/
http://census.ire.org/data/bulkdata.html
https://datausa.io/profile/geo/california
https://geodata.lib.berkeley.edu/?f%5Bdc_format_s%5D%5B%5D=Shapefile&f%5Bdct_spatial_sm%5D%5B%5D=California&page=10
https://data.cnra.ca.gov/dataset?groups=energy
https://solareis.anl.gov/maps/gis/index.cfm
https://blmsolar.anl.gov/maps/shapefiles/
https://www.energy.ca.gov/data-reports/energy-almanac/california-electricity-data
https://databasin.org/galleries/07acc8fec40a41f586a421cdcef98e96/#expand=103254%2C103255%2C103222%2C103220

https://healthyplacesindex.org/data-reports/
https://healthyplacesindex.org/data-reports/
https://data.cnra.ca.gov/group/land-management
https://data.cnra.ca.gov/dataset/school-lands
https://data.cnra.ca.gov/dataset/high-water-line
https://data.cnra.ca.gov/dataset/low-water-line
https://data.cnra.ca.gov/dataset/calfire-forestdistricts1
https://data.cnra.ca.gov/dataset/significant-lands-water-lines
https://data.cnra.ca.gov/dataset/california-power-plants
https://data.cnra.ca.gov/dataset/california-building-climate-zones
https://data.cnra.ca.gov/dataset/california-wind-resource-area
https://data.cnra.ca.gov/dataset/2019-utility-scale-solar-capacity-by-county
https://data.cnra.ca.gov/dataset/2019-utility-scale-solar-electrical-generation-gwh-and-capacity-mw
https://www.energy.ca.gov/data-reports/energy-almanac/california-electricity-data/california-electrical-energy-generation
https://data.cnra.ca.gov/dataset/california-electricity-demand-forecast-zones
https://data.cnra.ca.gov/dataset/california-electric-utility-service-areas
https://data.cnra.ca.gov/dataset/california-natural-gas-service-area
https://data.cnra.ca.gov/dataset/california-electric-balancing-authority
https://gis.data.ca.gov/search?collection=Dataset
https://data.cnra.ca.gov/dataset/biosds714-fmu
https://data.cnra.ca.gov/dataset/vegetation-survey-points-cdfw-ds1020
https://data.cnra.ca.gov/dataset/suction-dredge-special-regulations-2020-ds788
https://data.cnra.ca.gov/dataset/conservation-plan-boundaries-hcp-and-nccp-ds760
https://data.cnra.ca.gov/dataset/wildlife-conservation-board-wcb-approved-projects-ds672
https://data.cnra.ca.gov/dataset/seeps-and-springs-ace-ds27311
https://data.cnra.ca.gov/dataset/lakes-by-watershed-ace-ds2762
https://data.cnra.ca.gov/dataset/species-biodiversity-ace-ds2769
https://data.cnra.ca.gov/dataset/communityvulnerability2020
https://ucsd.libguides.com/gis/data-geographic-region#CALIFORNIA
https://data.ca.gov/dataset/surface-water-water-quality-regulated-facility-information
https://data.ca.gov/dataset/water-quality-data
https://data.ca.gov/dataset/surface-water-sampling-location-information
https://data.ca.gov/dataset/surface-water-habitat-results
https://data.cnra.ca.gov/dataset/calgw_update2020/resource/ed76914e-e8e2-4731-aecc-6417e90ff9b1
https://data.cnra.ca.gov/dataset/calgw_update2020/resource/4cee2c97-2a5f-433d-b52c-99cbf31d57cf
https://data.ca.gov/dataset/quarterly-census-of-employment-and-wages-qcew
https://www.edd.ca.gov/newsroom/unemployment-april-2021.htm
https://www.labormarketinfo.edd.ca.gov/data/Top-Statistics.html#UR
https://www.bls.gov/bls/geography.htm

https://data.ers.usda.gov/reports.aspx?ID=17828
https://www.labormarketinfo.edd.ca.gov/data/Top-Statistics.html#UR
https://www.labormarketinfo.edd.ca.gov/data/interactive-labor-market-data-tools.html
https://www.labormarketinfo.edd.ca.gov/geography/supply-and-demand-tool.html
https://data.edd.ca.gov/d/yimz-gm8e/visualization
https://data.ca.gov/dataset/taxablesalesbycitysmall1
https://data.ca.gov/dataset/taxablesalesbycitylarge1
https://data.ca.gov/dataset/taxsaleslargecounties1
https://data.ca.gov/dataset/taxsalessmallcounties1
https://data.ca.gov/dataset/taxsalesbycounty1
https://data.ca.gov/dataset/sutdrevdistcountiescountytranstax
https://data.ca.gov/dataset/cdtfa-administrative-areas1
https://data.ca.gov/dataset/wioa-regional-planning-units-boundary-map
https://data.ca.gov/dataset/california-counties
https://data.ca.gov/dataset/california-counties
https://data.ca.gov/dataset/regional-economic-markets-boundary-map
https://data.ca.gov/dataset/california-metropolitan-statistical-areas-msa-and-metropolitan-divisions-md
https://data.ca.gov/dataset/facility-profile-attributes
https://data.ca.gov/dataset/death-profiles-by-zip-code
https://data.ca.gov/dataset/managed-care-provider-network
https://data.chhs.ca.gov/dataset/electronic-health-record-ehr-incentive-program-payments-for-eligible-providers3/resource/d59097d4-dfa2-40be-b95f-a025c52f82d0?inner_span=True
https://data.chhs.ca.gov/dataset/electronic-health-record-ehr-incentive-program-payments-to-eligible-hospitals3/resource/26c4dac8-93e0-4465-8881-49bd5d11b811?inner_span=True
https://data.ca.gov/dataset/licensed-and-certified-healthcare-facility-listing-may-20211
https://data.ca.gov/dataset/enrolled-medi-cal-fee-for-service-provider-file2
https://data.ca.gov/dataset/family-pact-providers-file1
https://apps.gis.ucla.edu/geodata/dataset/california-public-schools
https://apps.gis.ucla.edu/geodata/dataset/california-primary-and-secondary-roads
https://apps.gis.ucla.edu/geodata/dataset/california-places-cities-and-census-designated-places
https://apps.gis.ucla.edu/geodata/dataset/california-census-tracts-with-census-2010-dp1
https://catalog.data.gov/dataset/tiger-line-shapefile-2017-state-california-current-county-subdivision-state-based
https://www.arcgis.com/home/item.html?id=2f227372477d4cddadc0cd0b002ec657
https://geodata.lib.berkeley.edu/catalog/stanford-zp226vq1688
https://geodata.lib.berkeley.edu/catalog/stanford-jt346pj7452
https://geodata.lib.berkeley.edu/catalog/stanford-cj067sr0133
https://data.cnra.ca.gov/dataset/wind-project-size-in-california-counties-and-the-united-states
https://apps.gis.ucla.edu/geodata/dataset/north-america-roads-and-highways
https://geodata.lib.berkeley.edu/catalog/stanford-wh394qy4535
https://geodata.lib.berkeley.edu/catalog/stanford-ws423tx2448
https://geodata.lib.berkeley.edu/catalog/stanford-dw718fh3417
https://geodata.lib.berkeley.edu/catalog/stanford-mj355yr6542
https://cecgis-caenergy.opendata.arcgis.com/documents/CAEnergy::utility-scale-renewable-and-non-renewable-electrical-generation-by-county/explore
https://data.cnra.ca.gov/dataset/utility-scale-renewable-and-non-renewable-electrical-generation-by-county
https://data.cnra.ca.gov/dataset/electric-utility-service-area
https://data.cnra.ca.gov/dataset/utility-scale-renewable-and-non-renewable-energy
https://solareis.anl.gov/maps/gis/index.cfm
https://www.cnu.org/publicsquare/2021/02/08/defining-15-minute-city
https://gisgeography.com/rasterization-vectorization/