A Project Report

on

Data Analysis using ML on Geolocational Data

Submitted in partial fulfillment of the

requirement for** the **award of** the **degree of

Bachelor of Technology in Computer Science andEngineering

Under the Supervision of

Dr. J.N. Singh

Assistant Professor

Department of Computer Science and Engineering

Submitted By :

Gourang Ajmera - 18SCSE1010669

Alok Singh - 18SCSE1010135

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING, DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING GALGOTIAS UNIVERSITY, GREATER NOIDA, INDIA DECEMBER - 2021

Report for Review-2

Project Title: DATA ANALYSIS USING ML ON GEOLOCATIONAL DATA

ABSTRACT:

Machine learning allows us to feed computer algorithms with large amounts of data and make computers analyze and make data-driven decisions and recommendations based solely on input data. This project will utilize ML to analyze geolocational data and user preferences to make smart recommendations to the user . In the fast-paced and busy environment that the average person lives in, it often happens that one is too tired to prepare a home-cooked meal. And of course, even if you get home cooked meals every day, it is not uncommon for you to want to have a good meal every now and then for social / recreational purposes. Now, imagine a scenario where someone has just moved to a new location. They already have certain preferences, certain tastes. It will save a lot of trouble for the student and food suppliers if the student lives near his favorite outlet. The convenience of the means better sales and time savings for customers. This project involves the utilization of K-Means Clustering to seek out the simplest accommodation for students in Bangalore (or the other city of your choice) by classifying accommodation for incoming students on the idea of their preferences on amenities, budget and proximity to the location.

Keywords:

. ML

. Unsupervised Learning

. Simulation

. Smart Recommendations

. Data-driven approach

Introduction:

In the fast-paced and busy environment where the average person lives, it is common for people to be too tired to prepare home-cooked meals. And of course, even if you eat homemade food every day, it's not unusual if you want to go out to eat once in a while for social/recreational purposes. However, it is a commonly understood idea that no matter where you live, the food you eat is an important aspect of the lifestyle you live. Now, imagine a scenario where someone has just moved to a new location. They already have certain preferences, certain tastes. This will save students a lot of trouble and food suppliers if the student lives near his or her favorite outlet. This project involves using K-Means Clustering to find the best accommodation for

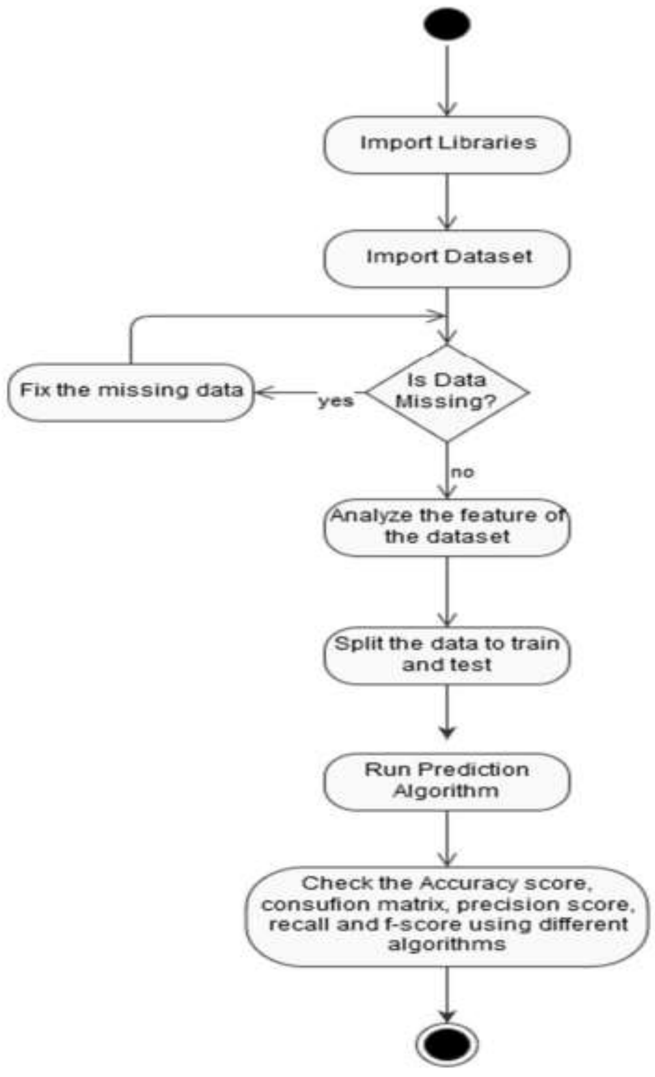
students in Bangalore (or another city of your choice) by rating accommodation for incoming students based on their preferences on facilities, budget and proximity to location.

Literature Review:

There are many algorithms derived above to determine k automatically. Most of these methods are wrappers around kmeans or some other clustering algorithm for fixed k. Use the wrapper method divide and combine the rules for centers to increase or decrease the value of k as the algorithm progresses.After calculating the BIC or Bayesian Information Criterion(BIC is a method for scoring and selecting a model) for each clustering model. Apart from BIC, other scoring functions are also available. Some researchers use the MDL method to find the best

k. The researchers also used the Minimum Description Length (MDL) framework, where the description length is the measurement value that tells us how well the data fit the model. This algorithm starts with a large value for k and removes the center (reduces k) each time that selection reduces the length of the description. Among the k reduction steps, they used the kmeans algorithm to optimize the fit of the model to the data.

Activity Diagram:


Import Libraries
Import Dataset
Is Data Missing?
yes
Fix the missing data
no
Analyze the feature of the dataset
Split the data to train and test
Run Prediction Algorithm
Check the Accuracy score, consufion matrix, precision score, recall and f-score using different algorithms

Required tools:

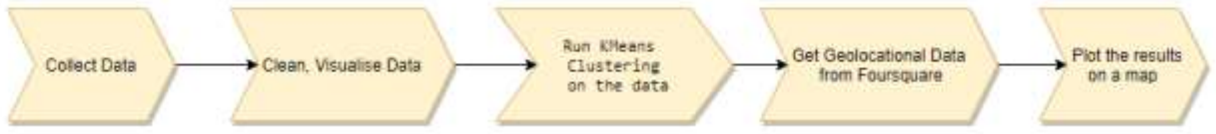
- Python
- For data - numpy and pandas package.
- For plotting - matplotlib package & seaborn packages
- For geospatial - geopy, folium.
- For machine learning - sklearn (preprocessing and cluster) scipy,
- For deep learning - minisom

<u>Feasibility Analysis:</u>

The project Data Analysis using ML on Geolocational Data is a simple software application which is supported on a personal computer , just like any native application. It is a python based project with the renowned python libraries to manage the application. For an application as complex as an recommendation system using machine learning and AI , this software has simplistic approach and does not have many complex features therefore, it can be used at the bare-bones level. Sklearn (preprocessing and cluster) , scipy , matplotlib & seaborn packages , pandas packages for python are the other packages that we will be using in this application. Python provides us with many different features and services to complete our project.The convenient design of the geopy package for the plotting on map and the simple user experience will provide smooth and good experience to the user.

<u>Complete work plan layout:</u>

The development of this application software will follow an iterative software development model in which the base product is first implemented as a small set of software requirements, then iterative enhancements and development releases are made until the entire system is implemented and ready for deployment. The first step would be collection of data set followed by visualizing the data. Then imposing K-Means clustering algorithm on the data. Get geolocational data from the resulting data-set and further plotting it on the map.


Collect Data
Clean, Visualise Data
Run KMeans Clustering on the data
Get Geolocational Data from Foursquare
Plot the results on a map

Modeling and Prediction with Machine Learning -

The main goal of the entire project is to predict the occurrence of heart disease with the highest accuracy. To achieve this, we will test several classification algorithms. This section covers all the results obtained from the study and introduces the best performer according to the accuracy metric. We have chosen a number of specific algorithms to solve supervised learning problems across classification methods.

First, let's arm ourselves with a handy tool that benefits from the coherence of the SciKit Learn library and creates a common task for training our models. The reason we display accuracy on both the train and test sets is to allow us to evaluate whether the model overfits or underfits the data.

Logistic regression –

Logistic regression is a classification algorithm used to assign observations to a discrete set of classes. Unlike linear regression, which outputs continuous number values, logistic regression uses the logistic sigmoid function to transform its output to return a probability value that can then be mapped to two or more discrete classes.

Types of Logical Regression:

- binary (pass/fail)
- multi (cats, dogs, sheep)

Process -

1. Split the problem into an n+1 binary classification problem (+1 because the index starts at 0).
2. For each class…
3. Estimate the probability that the observations will be in that single class.
4. prediction = <math>max (probability of classes)

Accuracy score of Logistic Regression is: 85.25%

Random Forest –

Random Forest is a supervised learning algorithm. Random forest can be used for both classification and regression problems, by using random forest regressor we can use random forest on regression problems. But we have used random forest on classification in this project so we will only consider the classification part.

Random Forest pseudocode –

- Randomly select “k” features from total “m” features.

  Where k << m

- Among the “k” features, calculate the node “d” using the best split point.
- Split the node into daughter nodes using the best split.
- Repeat 1 to 3 steps until “l” number of nodes has been reached.
- Build forest by repeating steps 1 to 4 for “n” number times to create “n” number of trees.

Random forest prediction pseudocode –

1. Takes the test features and use the rules of each randomly created decision tree to predict the outcome and stores the predicted outcome.

2. Calculate the votes for each predicted target.

3. Consider high voted predicted target as final prediction from random forest algo.

Accuracy score of Random Forest is 86.9%

Naïve Bayes -

Bayes’ Theorem is stated as:

P(h|d) = (P(d|h) * P(h)) / P(d)

- P(h|d) is the probability of hypothesis h given the data d. This is called the posterior probability.

- P(d|h) is the probability of data d given that the hypothesis h was true.

- P(h) is the probability of hypothesis h being true (regardless of the data). This is called the prior probability of h.

- P(d) is the probability of the data (regardless of the hypothesis).

Here, we are interested in calculating the posterior probability of P(h|d) from the prior probability p(h) with P(D) and P(d|h). After calculating the posterior probability for a number of different hypotheses, we will select the hypothesis with the highest probability. This is the maximum probable hypothesis and may formally be called the (MAP) hypothesis.

This can be written as:

MAP(h) = max(P(h|d))

Or

MAP(h) = max((P(d|h) * P(h)) / P(d))

Or

MAP(h) = max(P(d|h) * P(h))

The P(d) is a normalizing term which allows us to calculate the probability. We can drop it when we are interested in the most probable hypothesis as it is constant and only used to normalize. Back to classification, if we have an even number of instances in each class in our training data, then the probability of each class (e.g., P(h)) will be equal. Again, this would be a constant term in our equation, and we could drop it so that we end up with -

MAP(h) = max(P(d|h))

Naive Bayes is a classification algorithm for binary (two-class) and multi-class classification problems. The technique is easiest to understand when described using binary or categorical input values. It is called naive Bayes or idiot Bayes because the calculation of the probabilities for each hypothesis are simplified to make their calculation tractable. Rather than attempting to calculate the values of each attribute value P (d1, d2, d3|h), they are assumed to be conditionally independent given the target value and calculated as P(d1|h) * P(d2|H) and so on. This is a very strong assumption that is most unlikely in real data, i.e. that the attributes do not interact. Nevertheless, the approach performs surprisingly well on data where this assumption does not hold.

MAP(h) = max(P(d|h) * P(h))

Gaussian Naïve Bayes -

mean(x) = 1/n * sum(x)

Where n is the number of instances and x are the values for an input variable in your training data. We can calculate the standard deviation using the following equation -

standard deviation(x) = sqrt (1/n * sum(xi-mean(x)^2))

This is the square root of the average squared difference of each value of x from the mean value of x, where n is the number of instances, sqrt() is the square root function, sum() is the sum function, xi is a specific value of the x variable for the ith instance and mean(x) is described above, and ^2 is the square. Gaussian PDF with a new input for the variable, and in return the Gaussian PDF will provide an estimate of the probability of that new input value

for that class.

pdf (x, mean, sd) = (1 / (sqrt (2 * PI) * sd)) * exp (-((x-mean^2) / (2*sd^2)))

Where pdf(x) is the Gaussian Probability Density Function (PDF), sqrt () is the square root, mean and sd are the mean and standard deviation calculated above, Pi is the numerical constant, exp () is the numerical constant e or Euler’s number raised to power and x is the input value for the input variable.

K-Nearest Neighbor –

We can implement a KNN model by following the below steps:

1. Load the data

2. Initialize the value of k

3. For getting predicted class, iterate from 1 to total number of training data points.

- Calculate the distance between test data and each row of training data. Here we will use Euclidean distance as our distance metric since it’s the most popular method. The other metrics that can be used are Chebyshev, cosine, etc.
- Sort the calculated distances in ascending order based on distance values.
- Get top k rows from the sorted array.
- Get the most frequent class of these row.
- Return the predicted class.

Pseudocode -

- Place the best attribute of the dataset at the root of the tree.

- Split the training set into subsets. Subsets should be made in such a way that each subset contains data with the same value for an attribute.

- Repeat step 1 and step 2 on each subset until you find leaf nodes in all the branches of the tree.

Assumptions while creating Decision Tree –

- At the beginning, the whole training set is considered as the root.

- Feature values are preferred to be categorical. If the values are continuous then they are discretized prior to building the model.

- Records are distributed recursively on the basis of attribute values.

- Order to placing attributes as root or internal node of the tree is done by using some statistical approach.

- The popular attribute selection measures -

  1. Information gain
  2. Gini index

Minimum System Requirements –

Processors - Intel Atom® processor or Intel® Core™ i3 processor

Disk space - 3 GB or more

Operating systems - Windows 7 /8.1/ 10, OSX-10.8+

Python* versions - 2.7.X, 3.6.X

HDD: 3 GB free space / May vary for different data-sets

Source Code:

```
<a href="https://colab.research.google.com/github/AKG1301/Exploratory-Data-Analysis-on-Geolocational-Data/blob/main/Exploratory_Data_Analysis_on_Geolocational_Data.ipynb" target="_parent"><img src="https://colab.research.google.com/assets/colab-badge.svg" alt="Open In Colab"/></a>
```

#Exploratory Analysis of Geolocational Data

## Data Collection

```
import pandas as pd
data=pd.read_csv("/content/drive/MyDrive/  Exploratory Analysis of Geolocational Data/food_coded.csv")
data
```

#Data Cleaning

The process of Extracting the features, (and dealing with different kinds of values as well as NaN values) is known as Data Cleaning.

```
data.columns
column=['cook','eating_out','employment','ethnic_food',
'exercise','fruit_day','income','on_off_campus','pay_meal_out','sports','veggies_day'
]
```

```
d=data[column]

d

## Data Exploration and Visualisation




import seaborn as sns

sns.pairplot(d)

#Boxplot of Dataset

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

% matplotlib inline

ax=d.boxplot(figsize=(16,6))

ax.set_xticklabels(ax.get_xticklabels(),rotation=30)

d.shape

s=d.dropna()

## Run KMeans Clustering on the data

## for data

import numpy as np
import pandas as pd

## for plotting

import matplotlib.pyplot as plt

import seaborn as sns

## for geospatial

import folium

import geopy

## for machine learning

from sklearn import preprocessing, cluster

import scipy

## for deep learning

import minisom

f=['cook','income']

X = s[f]

max_k = 10

## iterations

distortions = []

for i in range(1, max_k+1):

    if len(X) >= i:
```

```
    model = cluster.KMeans(n_clusters=i, init='k-means++', max_iter=300,
n_init=10, random_state=0)
    model.fit(X)
    distortions.append(model.inertia_)
## best k: the lowest derivative
k = [i*100 for i in np.diff(distortions,2)].index(min([i*100 for i
   in np.diff(distortions,2)]))
## plot
fig, ax = plt.subplots()
ax.plot(range(1, len(distortions)+1), distortions)
ax.axvline(k, ls='--', color="red", label="k = "+str(k))
ax.set(title='The Elbow Method', xlabel='Number of clusters',
    ylabel="Distortion")
ax.legend()
ax.grid(True)
plt.show()


## Get Geolocational Data
from pandas.io.json import json_normalize
import folium
```

```
from geopy.geocoders import Nominatim
import requests
CLIENT_ID =
"KTCJJ2YZ2143QHEZ2JAQS4FJIO5DLSDO0YN4YBXPMI5NKTEF" # your
Foursquare ID
CLIENT_SECRET =
"KNG2LO22BPLHN1E3OAHWLYQ5PQBN14XYZMEMAS0CPJEJKOTR" #
your Foursquare Secret
VERSION = '20200316'
LIMIT = 10000
url =
'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v
={}&ll={},{}&radius={}&limit={}'.format(
    CLIENT_ID,
    CLIENT_SECRET,
    VERSION,
    17.448372, 78.526957,
    30000,
    LIMIT)
results = requests.get(url).json()
```

```
results

venues = results['response']['groups'][0]['items']

nearby_venues = json_normalize(venues)
```

## Adding two more Columns Restaurant and Others

1. Restaurant: Number of Restaurant in the radius of 20 km
2. others:Number of Gyms, Parks,etc in the radius of 20 km

```
resta=[]

oth=[]

for lat,long in

zip(nearby_venues['venue.location.lat'],nearby_venues['venue.location.lng']):

   url =

'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v

={}&ll={},{}&radius={}&limit={}'.format(

       CLIENT_ID,

       CLIENT_SECRET,

       VERSION,

       lat,long,

       1000,

       100)
```

```
    res = requests.get(url).json()

    venue = res['response']['groups'][0]['items']

    nearby_venue = json_normalize(venue)

    df=nearby_venue['venue.categories']


    g=[]

    for i in range(0,df.size):

      g.append(df[i][0]['icon']['prefix'].find('food'))

    co=0

    for i in g:

      if i>1:

        co+=1

    resta.append(co)

    oth.append(len(g)-co)


nearby_venues['restaurant']=resta

nearby_venues['others']=oth

nearby_venues

## Changing the Column Name

lat=nearby_venues['venue.location.lat']
```

```
long=nearby_venues['venue.location.lng']
```

## Install the minisom library using pip

MiniSom is a minimalistic and Numpy based implementation of the Self Organizing Maps (SOM). SOM is a type of Artificial Neural Network able to convert complex, nonlinear statistical relationships between high-dimensional data items into simple geometric relationships on a low-dimensional display. Minisom is designed to allow researchers to easily build on top of it and to give students the ability to quickly grasp its details.

```
pip install minisom
```

## Run K Means clustering on the dataset, with the optimal K value using Elbow Method

A fundamental step for any unsupervised algorithm is to determine the optimal number of clusters into which the data may be clustered. The Elbow Method is one of the most popular methods to determine this optimal value of k.

```
f=['venue.location.lat','venue.location.lng']
X = nearby_venues[f]
max_k = 10
```

## iterations

```
distortions = []
for i in range(1, max_k+1):
   if len(X) >= i:
      model = cluster.KMeans(n_clusters=i, init='k-means++', max_iter=300,
n_init=10, random_state=0)
      model.fit(X)
      distortions.append(model.inertia_)
## best k: the lowest derivative
k = [i*100 for i in np.diff(distortions,2)].index(min([i*100 for i
    in np.diff(distortions,2)]))
## plot
fig, ax = plt.subplots()
ax.plot(range(1, len(distortions)+1), distortions)
ax.axvline(k, ls='--', color="red", label="k = "+str(k))
ax.set(title='The Elbow Method', xlabel='Number of clusters',
      ylabel="Distortion")
ax.legend()
ax.grid(True)
plt.show()
city = "Hyderabad"
```

```
## get location

locator = geopy.geocoders.Nominatim(user_agent="MyCoder")

location = locator.geocode(city)

print(location)

## keep latitude and longitude only

location = [location.latitude, location.longitude]

print("[lat, long]:", location)

nearby_venues.head()

nearby_venues.columns

##Data Cleaning Process for Extracting Necessary Columns in the Dataset

n=nearby_venues.drop(['referralId', 'reasons.count', 'reasons.items', 'venue.id',

      'venue.name',

      'venue.location.labeledLatLngs', 'venue.location.distance',

      'venue.location.cc',

      'venue.categories', 'venue.photos.count', 'venue.photos.groups',

      'venue.location.crossStreet', 'venue.location.address','venue.location.city',

      'venue.location.state', 'venue.location.crossStreet',

      'venue.location.neighborhood',     'venue.venuePage.id',

      'venue.location.postalCode','venue.location.country'],axis=1)

n.columns
```

```
## Dropping Nan Values from Dataset

n=n.dropna()

n = n.rename(columns={'venue.location.lat': 'lat', 'venue.location.lng': 'long'})

n

###Convert Every Row of Column ***'venue.location.formattedAddress'*** from

List to String

n['venue.location.formattedAddress']

spec_chars = ["[","]"]

for char in spec_chars:

  n['venue.location.formattedAddress'] =

n['venue.location.formattedAddress'].astype(str).str.replace(char, ' ')

#Plot the clustered locations on a map

x, y = "lat", "long"

color = "restaurant"

size = "others"

popup = "venue.location.formattedAddress"

data = n.copy()


## create color column
```

```
lst_colors=["red","green","orange"]

lst_elements = sorted(list(n[color].unique()))


## create size column (scaled)

scaler = preprocessing.MinMaxScaler(feature_range=(3,15))

data["size"] = scaler.fit_transform(

        data[size].values.reshape(-1,1)).reshape(-1)


## initialize the map with the starting location

map_ = folium.Map(location=location, tiles="cartodbpositron",

         zoom_start=11)

## add points

data.apply(lambda row: folium.CircleMarker(

      location=[row[x],row[y]],popup=row[popup],

      radius=row["size"]).add_to(map_), axis=1)

X = n[["lat","long"]]

max_k = 10

## iterations

distortions = []

for i in range(1, max_k+1):
    if len(X) >= i:
        model = cluster.KMeans(n_clusters=i, init='k-means++', max_iter=300,
n_init=10, random_state=0)
        model.fit(X)
        distortions.append(model.inertia_)
## best k: the lowest derivative
k = [i*100 for i in np.diff(distortions,2)].index(min([i*100 for i in
np.diff(distortions,2)]))
## plot
fig, ax = plt.subplots()
ax.plot(range(1, len(distortions)+1), distortions)
ax.axvline(k, ls='--', color="red", label="k = "+str(k))
ax.set(title='The Elbow Method', xlabel='Number of clusters',
        ylabel="Distortion")
ax.legend()
ax.grid(True)
plt.show()


k = 6
model = cluster.KMeans(n_clusters=k, init='k-means++')
```

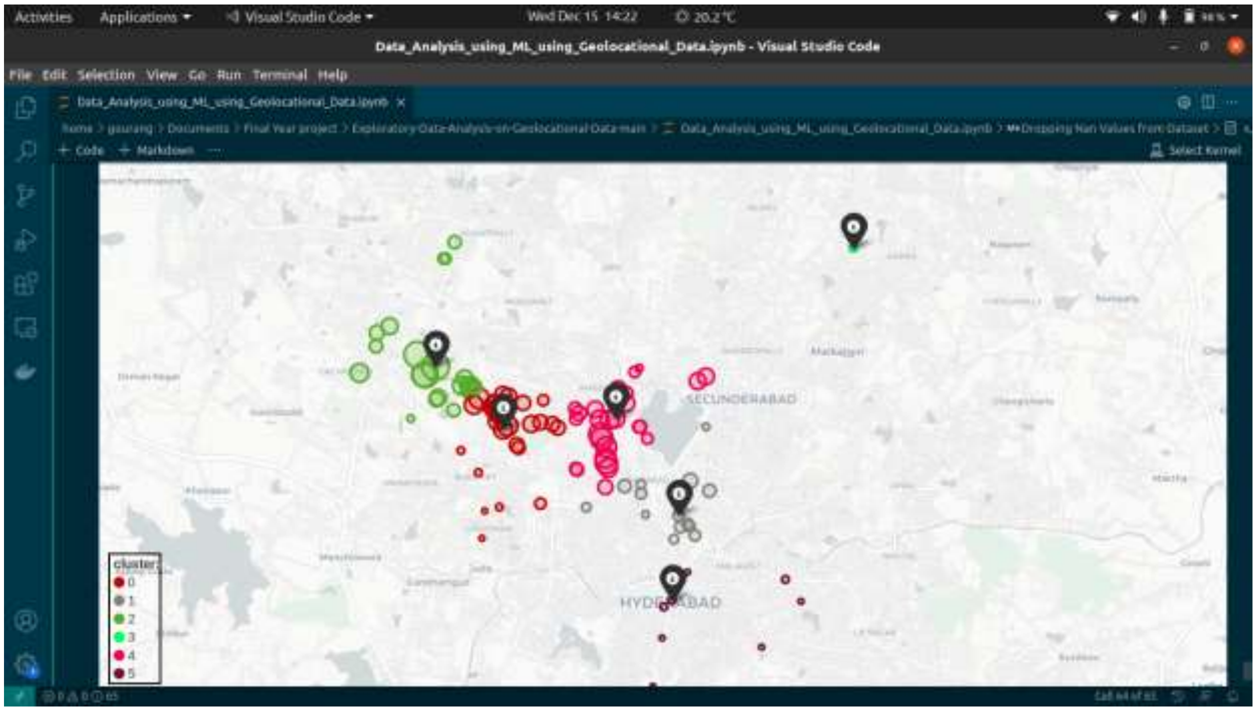
```
X = n[["lat","long"]]

## clustering

dtf_X = X.copy()

dtf_X["cluster"] = model.fit_predict(X)

## find real centroids

closest, distances = scipy.cluster.vq.vq(model.cluster_centers_,

                     dtf_X.drop("cluster", axis=1).values)

dtf_X["centroids"] = 0

for i in closest:

    dtf_X["centroids"].iloc[i] = 1

## add clustering info to the original dataset

n[["cluster","centroids"]] = dtf_X[["cluster","centroids"]]

n

## plot

fig, ax = plt.subplots()

sns.scatterplot(x="lat", y="long", data=n,

                palette=sns.color_palette("bright",k),

                hue='cluster', size="centroids", size_order=[1,0],

                legend="brief", ax=ax).set_title('Clustering (k='+str(k)+')')

th_centroids = model.cluster_centers_
```

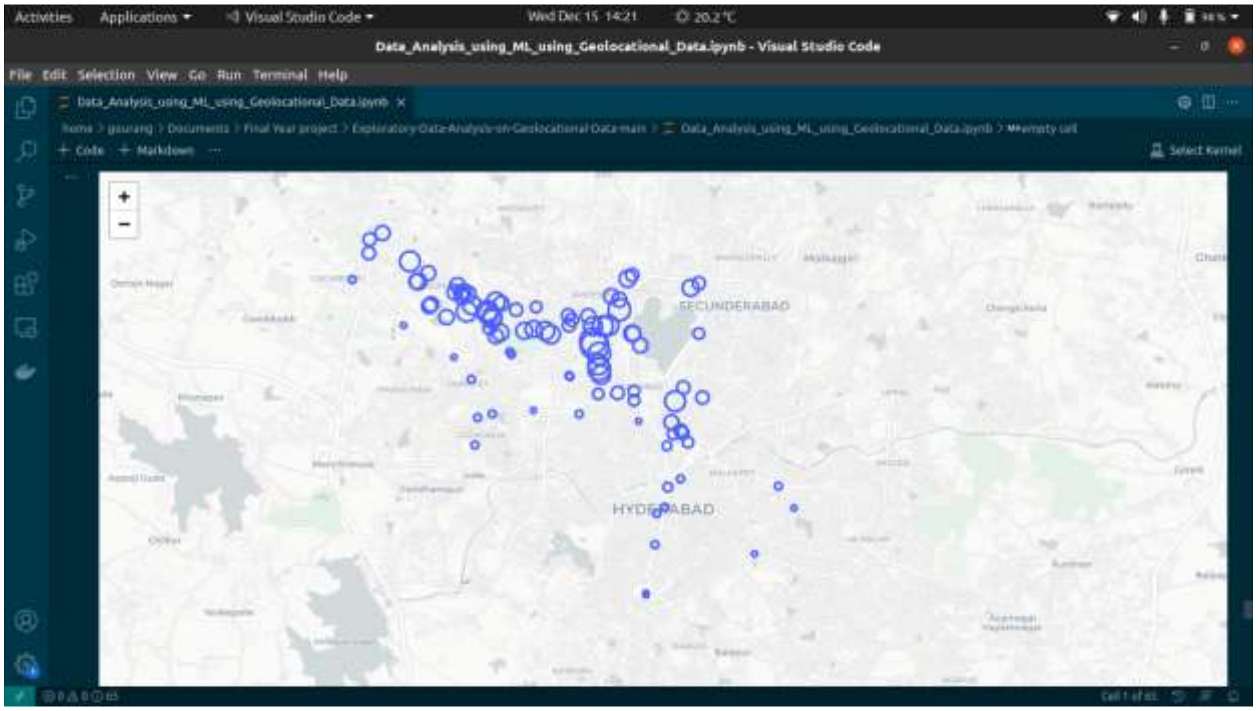
```
ax.scatter(th_centroids[:,0], th_centroids[:,1], s=50, c='black',
        marker="x")
model = cluster.AffinityPropagation()


k = n["cluster"].nunique()
sns.scatterplot(x="lat", y="long", data=n,
            palette=sns.color_palette("bright",k),
            hue='cluster', size="centroids", size_order=[1,0],
            legend="brief").set_title('Clustering (k='+str(k)+')')
x, y = "lat", "long"
color = "cluster"
size = "restaurant"
popup = "venue.location.formattedAddress"
marker = "centroids"
data = n.copy()
## create color column
lst_elements = sorted(list(n[color].unique()))
lst_colors = ['#%06X' % np.random.randint(0, 0xFFFFFF) for i in
          range(len(lst_elements))]
data["color"] = data[color].apply(lambda x:
```

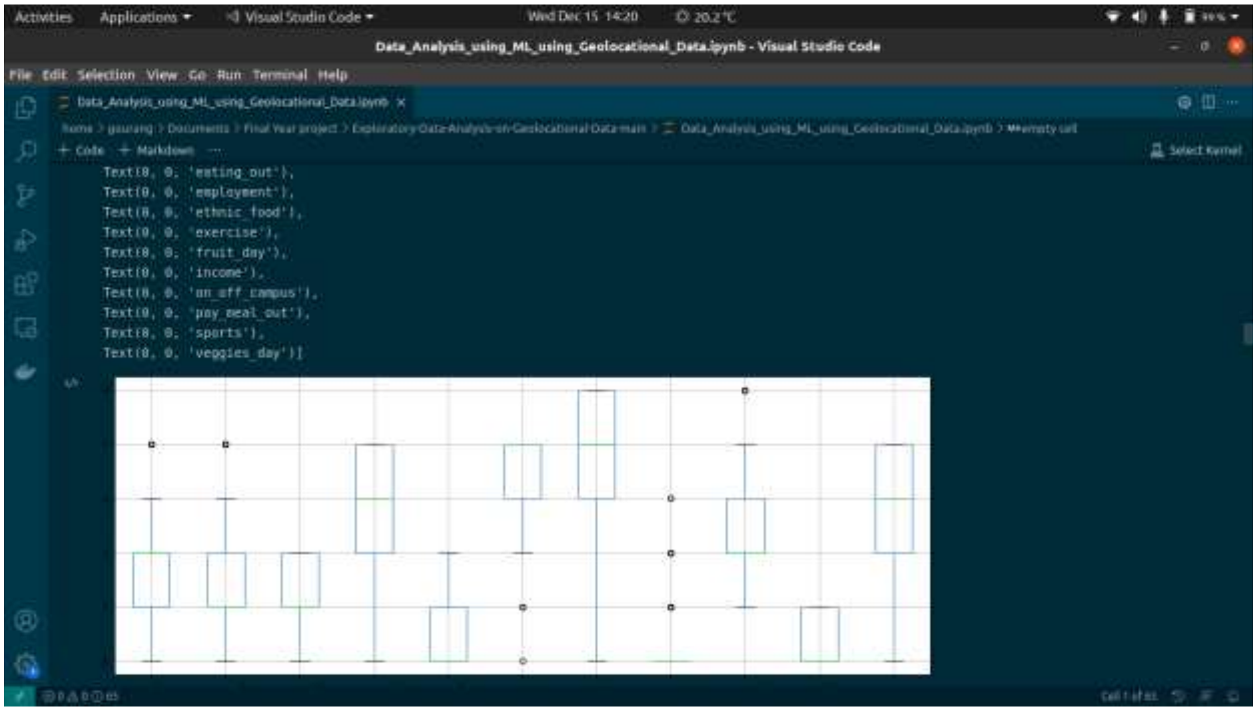
```
                lst_colors[lst_elements.index(x)])
## create size column (scaled)
scaler = preprocessing.MinMaxScaler(feature_range=(3,15))
data["size"] = scaler.fit_transform(
               data[size].values.reshape(-1,1)).reshape(-1)
## initialize the map with the starting location
map_ = folium.Map(location=location, tiles="cartodbpositron",
                  zoom_start=11)
## add points
data.apply(lambda row: folium.CircleMarker(
           location=[row[x],row[y]],
           color=row["color"], fill=True,popup=row[popup],
           radius=row["size"]).add_to(map_), axis=1)
## add html legend
legend_html = """<div style="position:fixed; bottom:10px; left:10px; border:2px
solid black; z-index:9999; font-size:14px;">&nbsp;<b>"""+color+""":</b><br>"""
for i in lst_elements:
    legend_html = legend_html+"""&nbsp;<i class="fa fa-circle
    fa-1x" style="color:"""+lst_colors[lst_elements.index(i)]+"""">
    </i>&nbsp;"""+str(i)+"""<br>"""
```

```
legend_html = legend_html+"""</div>"""

map_.get_root().html.add_child(folium.Element(legend_html))

## add centroids marker

lst_elements = sorted(list(n[marker].unique()))

data[data[marker]==1].apply(lambda row:

        folium.Marker(location=[row[x],row[y]],

        draggable=False,  popup=row[popup] ,

        icon=folium.Icon(color="black")).add_to(map_), axis=1)
```

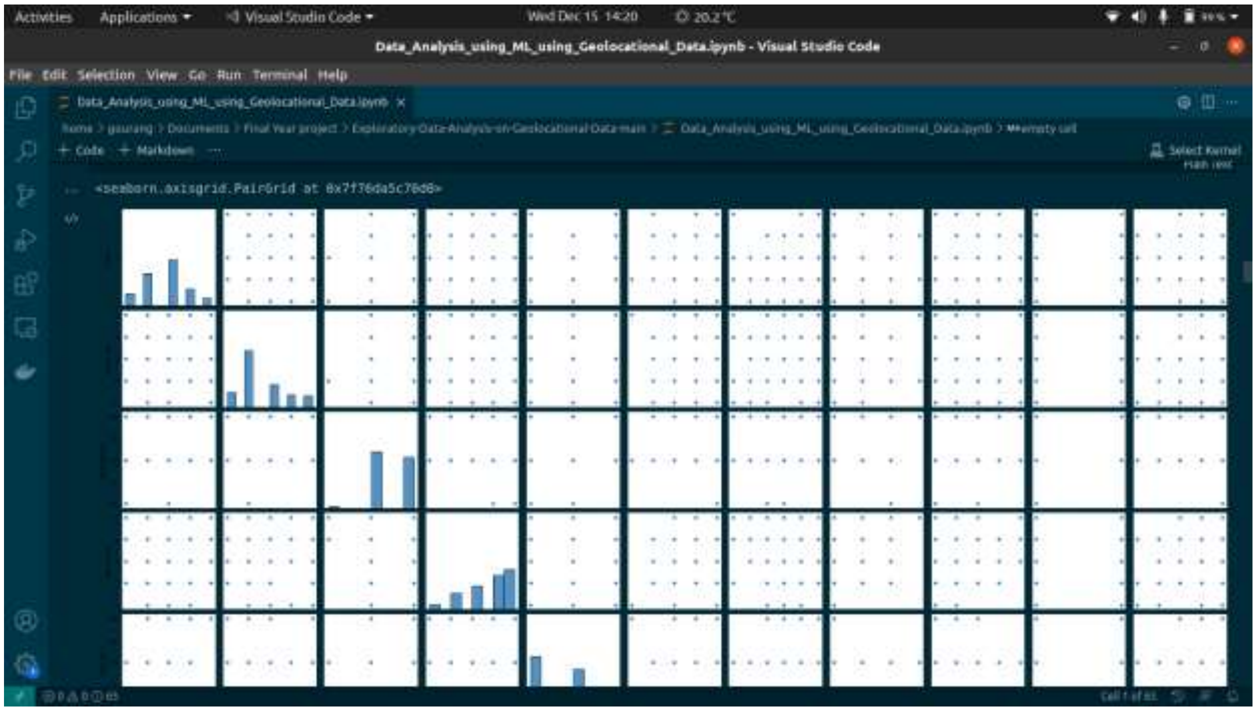
Results:


Activities
Applications
Visual Studio Code
Wed Dec 15 14:22
Data_Analysis_using_ML_using_Geolocational_Data.ipynb - Visual Studio Code
File Edit Selection View Go Run Terminal Help
Code
Markdown
Select Kernel
SECUNDERABAD
HYDERABAD
cluster
0
1
2
3
4
5


Activities
Applications
Visual Studio Code
Wed Dec 15 14:21
Data_Analysis_using_ML_using_Geolocational_Data.ipynb - Visual Studio Code
File Edit Selection View Go Run Terminal Help
Code
Markdown
Select Kernel
SECUNDERABAD
HYDERABAD


Activities
Applications
Visual Studio Code
Wed Dec 15 14:20
20.2 °C
Data_Analysis_using_ML_using_Geolocational_Data.ipynb - Visual Studio Code
File Edit Selection View Go Run Terminal Help
Data_Analysis_using_ML_using_Geolocational_Data.ipynb
+ Code + Markdown
Select Kernel

```
    Text(0, 0, 'eating_out'),
    Text(0, 0, 'employment'),
    Text(0, 0, 'ethnic_food'),
    Text(0, 0, 'exercise'),
    Text(0, 0, 'fruit_day'),
    Text(0, 0, 'income'),
    Text(0, 0, 'on_off_campus'),
    Text(0, 0, 'pay_meal_out'),
    Text(0, 0, 'sports'),
    Text(0, 0, 'veggies_day')]
```


Activities
Applications
Visual Studio Code
Wed Dec 15 14:20
20.2 °C
Data_Analysis_using_ML_using_Geolocational_Data.ipynb - Visual Studio Code
File Edit Selection View Go Run Terminal Help
Data_Analysis_using_ML_using_Geolocational_Data.ipynb
+ Code + Markdown
Select Kernel

```
<seaborn.axisgrid.PairGrid at 0x7f76da5c70d0>
```

GROUP ID- BT4122

Team Members:

GOURANG AJMERA

(18SCSE1010669, 18021011893, gourang_ajmera.scsebtech@galgotiasuniversity.edu.in)

ALOK SINGH

(18SCSE1010135,18021011382, alok_singh.scsebtech@galgotiasuniversity.edu.in)

Citations and References

[1] Springer research Paper by Jie Bao,Yu Zheng

https://link.springer.com/referenceworkentry/10.1007%2F978-3-319

[2]Data quest article on numpy,

https://www.dataquest.io/blog/numpy-tutorial-python/

[3] Matplotlib documentaions, https://matplotlib.org/

[4]Reposhub a hub of opensource github repositories,

https://reposhub.com/python/deep-learning/JustGlowing-minisom.html/>

[5] Scikit learn article on tutorial point,

[6]Seaborn :Statistical data visualization official documentations,

1. Saravanakumar Pichumani, T. V. P. Sundararajan, Rajesh Kumar Dhanaraj, Yunyoung Nam, Seifedine Kadry, "Ruzicka Indexed Regressive Homomorphic Ephemeral Key Benaloh Cryptography for Secure Data Aggregation in WSN," Journal of Internet Technology, vol. 22, no. 6 , pp. 1287-1297, Nov. 2021.

2. Kumar, D. R., Krishna, T. A., & Wahi, A. (2018). Health Monitoring Framework for in Time Recognition of Pulmonary Embolism Using Internet of Things. Journal of Computational and Theoretical Nanoscience, 15(5), 1598–1602. https://doi.org/10.1166/jctn.2018.7347

3. Krishnasamy, L., Dhanaraj, R. K., Ganesh Gopal, D., Reddy Gadekallu, T., Aboudaif, M. K., & Abouel Nasr, E. (2020). A Heuristic Angular Clustering Framework for Secured Statistical Data Aggregation in Sensor Networks. Sensors, 20(17), 4937. https://doi.org/10.3390/s20174937

4. Dhiviya, S., Malathy, S., & Kumar, D. R. (2018). Internet of Things (IoT) Elements, Trends and Applications. Journal of Computational and Theoretical Nanoscience,
15(5), 1639–1643. https://doi.org/10.1166/jctn.2018.7354

5. Rajesh Kumar, D., & Shanmugam, A. (2017). A Hyper Heuristic Localization Based Cloned Node Detection Technique Using GSA Based Simulated Annealing in Sensor Networks. In Cognitive Computing for Big Data Systems Over IoT (pp. 307–335). Springer International Publishing. https://doi.org/10.1007/978-3-319-70688-7_13

6. Prasanth, T., Gunasekaran, M., & Kumar, D. R. (2018, December). Big data Applications on Health Care. 2018 4th International Conference on Computing Communication and Automation (ICCCA). 2018 4th International Conference on Computing Communication and Automation (ICCCA). https://doi.org/10.1109/ccaa.2018.8777586

7. Sathish, R., & Kumar, D. R. (2013, April). Dynamic Detection of Clone Attack in Wireless Sensor Networks. 2013 International Conference on Communication Systems and

Network Technologies. 2013 International Conference on Communication Systems and Network Technologies (CSNT 2013). https://doi.org/10.1109/csnt.2013.110

8. Rajesh Kumar D, & ManjupPriya S. (2013, December). Cloud based M-Healthcare emergency using SPOC. 2013 Fifth International Conference on Advanced Computing (ICoAC). 2013 Fifth International Conference on Advanced Computing (ICoAC). https://doi.org/10.1109/icoac.2013.6921965

9. Sathish, R., & Kumar, D. R. (2013, March). Proficient algorithms for replication attack detection in Wireless Sensor Networks &#x2014; A survey. 2013 IEEE International Conference ON Emerging Trends in Computing, Communication and Nanotechnology (ICECCN). 2013 International Conference on Emerging Trends in Computing, Communication and Nanotechnology (ICE-CCN). https://doi.org/10.1109/ice-ccn.2013.6528465

10. Soumya Ranjan Jena, Raju Shanmugam, Rajesh Kumar Dhanaraj, Kavita Saini Recent Advances and Future Research Directions in Edge Cloud Framework. (2019). International Journal of Engineering and Advanced Technology, 9(2), 439–444. https://doi.org/10.35940/ijeat.b3090.129219

11. Kumar, R. N., Chandran, V., Valarmathi, R. S., & Kumar, D. R. (2018). Bitstream Compression for High Speed Embedded Systems Using Separated Split Look Up Tables (LUTs). Journal of Computational and Theoretical Nanoscience, 15(5), 1719–1727. https://doi.org/10.1166/jctn.2018.7367

12. Lalitha, K., Kumar, D. R., Poongodi, C., & Arumugam, J. (2021). Healthcare Internet of Things – The Role of Communication Tools and Technologies. In Blockchain, Internet of Things, and Artificial Intelligence (pp. 331–348). Chapman and Hall/CRC. https://doi.org/10.1201/9780429352898-17

13. Rajesh Kumar, D., Rajkumar, K., Lalitha, K., & Dhanakoti, V. (2020). Bigdata in the Management of Diabetes Mellitus Treatment. In Studies in Big Data (pp. 293–324). Springer Singapore. https://doi.org/10.1007/978-981-15-4112-4_14

14. Chandraprabha, M., & Dhanaraj, R. K. (2020, November 5). Machine learning based Pedantic Analysis of Predictive Algorithms in Crop Yield Management. 2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA). 2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA). https://doi.org/10.1109/iceca49313.2020.9297544

15. Dhanaraj, R. K., Rajkumar, K., & Hariharan, U. (2020). Enterprise IoT Modeling: Supervised, Unsupervised, and Reinforcement Learning. In Business Intelligence for Enterprise Internet of Things (pp. 55–79). Springer International Publishing. https://doi.org/10.1007/978-3-030-44407-5_3

16. Cynthia, J., Sankari, M., Suguna, M., & kumar, D. R. (2018, December). Survey on Disaster Management using VANET. 2018 4th International Conference on Computing Communication and Automation (ICCCA). 2018 4th International Conference on Computing Communication and Automation (ICCCA). https://doi.org/10.1109/ccaa.2018.8777331

17. Dhanaraj, R. K., Shanmugam, A., Palanisamy, C., & Natarajan, A. (2016). Optimal Clone Attack Detection Model using an Energy-Efficient GSA based Simulated Annealing in Wireless Sensor Networks. Asian Journal of Research in Social Sciences and Humanities, 6(11), 201. https://doi.org/10.5958/2249-7315.2016.01186.2

18. Sathya, K., & Kumar, D. R. (2012, February). Energy efficient clustering in sensor networks using Cluster Manager. 2012 International Conference on

Computing, Communication and Applications. 2012 International Conference on Computing, Communication and Applications (ICCCA). https://doi.org/10.1109/iccca.2012.6179177

19. Lalitha, K., Varadhaganapathy, S., Santhoshi, S., & Kumar, D. R. (2018, December). A Review on Possibilities of Hearing Loss and Implantable Hearing Devices for Teenagers. 2018 4th International Conference on Computing Communication and Automation (ICCCA). 2018 4th International Conference on Computing Communication and Automation (ICCCA). https://doi.org/10.1109/ccaa.2018.8777336

20. Krishnamoorthi, S., Jayapaul, P., Dhanaraj, R.K. et al. Design of pseudo-random number generator from turbulence padded chaotic map. Nonlinear Dyn (2021). https://doi.org/10.1007/s11071-021-06346-x

21. R. K. Dhanaraj, L. Krishnasamy, O. Geman and D. R. Izdrui, "Black hole and sink hole attack detection in wireless body area networks," Computers, Materials & Continua, vol. 68, no.2, pp. 1949–1965, 2021. doi:10.32604/cmc.2021.015363

22. Rajesh Kumar Dhanaraj, Lalitha, K., Anitha, S., Khaitan, S., Gupta, P., & Goyal, M. K. (2021). Hybrid and dynamic clustering-based data aggregation and routing for wireless sensor networks [JB]. Journal of Intelligent & Fuzzy Systems, 1–15.

23. M. D. Ramasamy, K. Periasamy, L. Krishnasamy, R. K. Dhanaraj, S. Kadry and Y. Nam, "Multi-Disease Classification Model using Strassen’s Half of Threshold (SHoT) Training Algorithm in Healthcare Sector," in IEEE Access, doi: 10.1109/ACCESS.2021.3103746.

24. Ramakrishnan, V., Chenniappan, P., Dhanaraj, R. K., Hsu, C. H., Xiao, Y., & Al-Turjman, F. (2021). Bootstrap aggregative mean shift clustering for big data anti-pattern detection analytics in 5G/6G communication networks. Computers & Electrical Engineering, 95, 107380.

25. KRISHNASAMY, L., RAMASAMY, T., DHANARAJ, R., & CHINNASAMY, P. (2021). A geodesic deployment and radial shaped clustering (RSC) algorithm with