Enhancing Urban Mobility and leveraging Urban Site Selection

Key to creating sustainable, efficient, and vibrant cities for the future."

Training 20 Slides Completed

time taken
Sem-4

GROUP–5

TEAMMATES

NAME	ROLL NUMBER
NIKHILESH KRISHNA	AM.EN.U4AIE22015
P SARVAN SRI SAI	AM.EN.U4AIE22039
K MUKESH	AM.EN.U4AIE22029
SARIGA SAJI	AM.EN.U4AIE22046

Problem statement

In urban planning and commercial development, selecting the optimal location for new commercial spaces such as shops, hotels, hospitals, offices, and railway stations is crucial for success. Traditional methods of site selection often result in costly mistakes due to poor accessibility, low foot traffic, and suboptimal placement relative to amenities. There is a need for a systematic and data-driven approach to identify the best locations for commercial establishments.

Use Case

Our proposed system addresses this need by leveraging datasets scraped from Google Maps and analyzing various factors such as proximity to amenities and accessibility via busy roads. This system aims to provide optimal location recommendations for businesses, city planners, and real estate developers. By integrating these features, we can enhance city planning, improve transit routing directions, and create a more efficient urban environment. The model continuously improves with more data, offering increasingly accurate recommendations over time, thereby aiding in informed decision-making for zoning, development, and commercial viability.

This project not only helps businesses and city planners but also benefits real estate developers by suggesting the best land development projects and identifying suitable locations for purchasing land based on specific requirements.

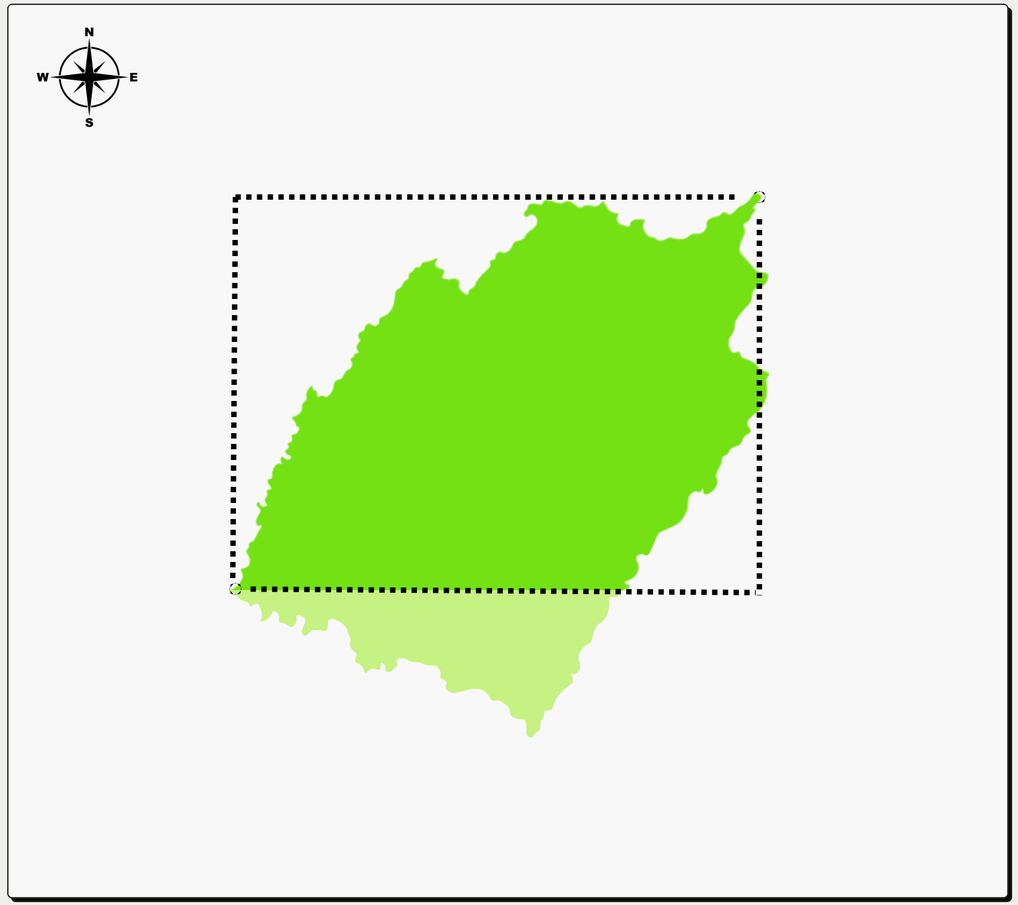
Group-5

Scraping the Data Sets from Google Maps

Exploring the Scraping Models

Selenium

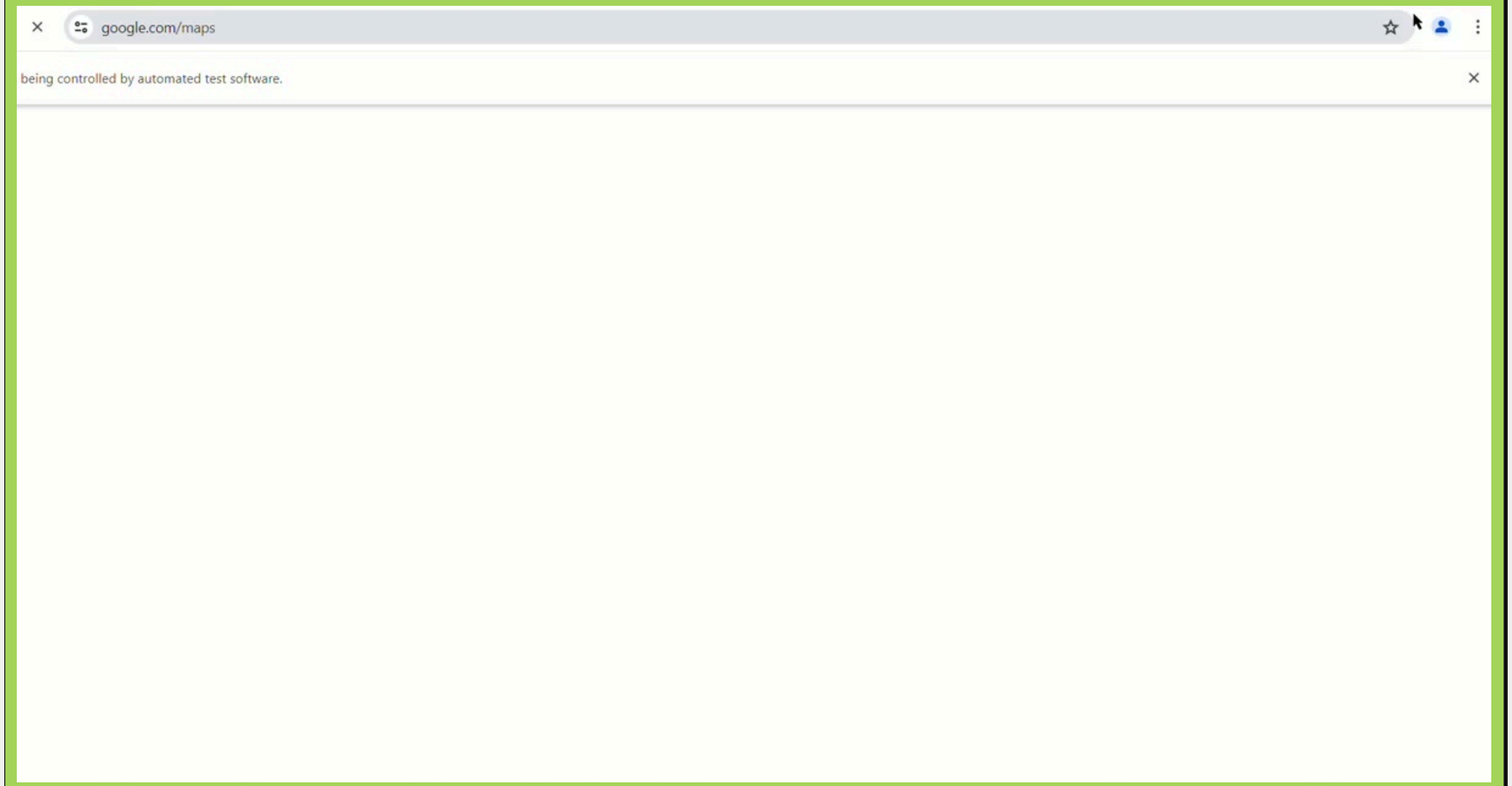
Selenium is a powerful tool for controlling web browsers through programs and performing browser automation. It is functional for all browsers, works on all major operating systems, and its scripts can be written in various programming languages, including Python. Selenium is primarily used for testing web applications but can be used for scraping data from web pages and automating repetitive web-based tasks.



BeautifulSoup:

BeautifulSoup is a Python library for parsing HTML and XML documents. It provides tools to navigate, search, and modify the parse tree, making it easier to extract data from web pages. BeautifulSoup is particularly useful for web scraping, as it handles common parsing issues such as poorly formatted HTML.

Exploring the Scraping Models



Web Scraping Process

Calculate Latitude and Longitude Differences:

- Calculate the change in latitude and longitude for a given distance using the moving_metre function..
- Store the calculated changes in d_latitude and d_longitude.

Initialize Data Storage:

- Create a list named data to store information about each location, including 'Name', 'Rating', 'Number of Reviews', 'Latitude', 'Longitude', 'Category', and 'Sub-Category'

Calculate Iterations:

- Compute the number of steps needed to move across the given latitude and longitude ranges.

Iterate Over the Grid:

- Use nested loops to iterate over the grid defined by the latitude and longitude ranges.

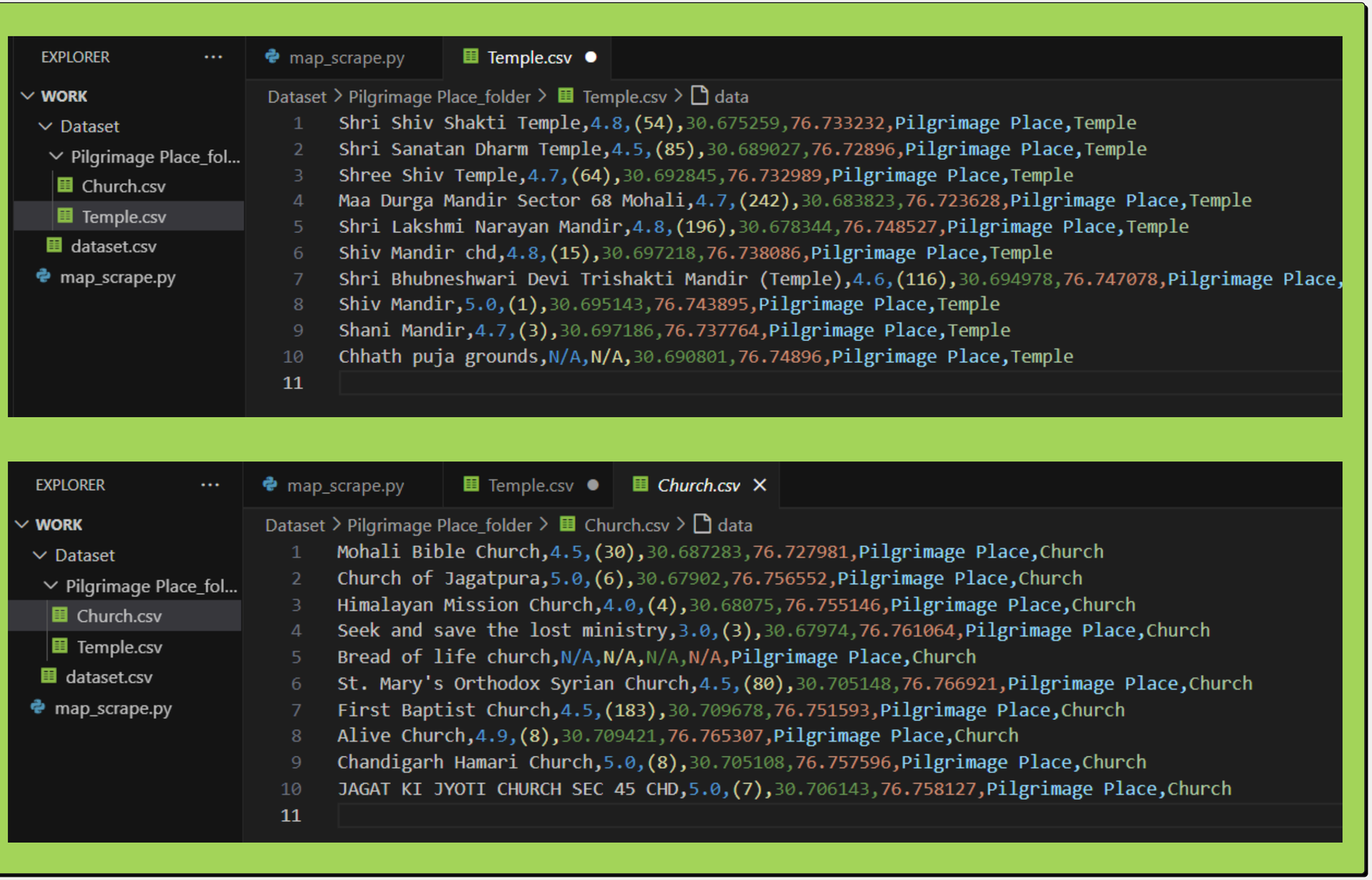
Set Starting Location:

- Set the starting location in the driver/browser using the set_starting_location function with the current latitude and longitude.

Exploring the Scraping Models

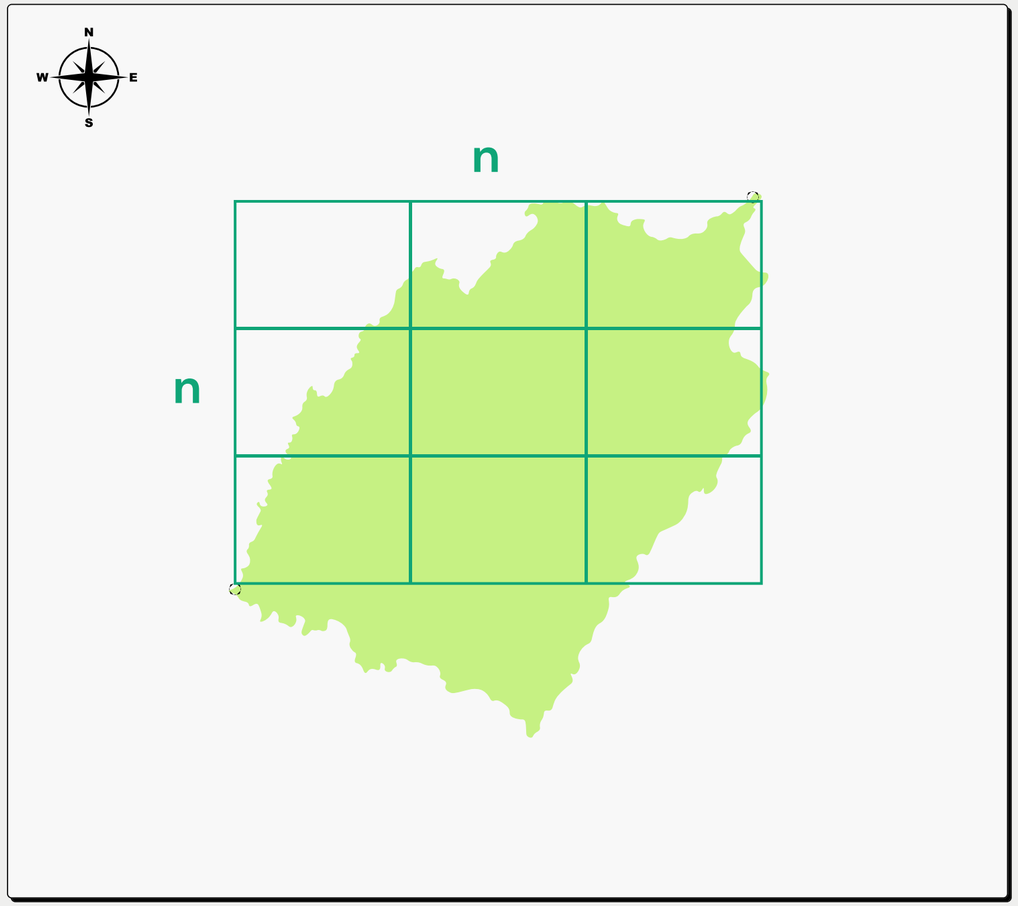
Perform Searches for Each Category and Sub-Category:

- Iterate over each category and its corresponding sub-categories.
- For each sub-category (or search query):
  - Perform the search using the perform_search function.
  - Click the update button using the click_update_button function.
  - Zoom the map to a specific label ('200 m') and width (57 px) using the zoom_until_label_and_width function.
  - Scroll down the page a specified number of times (scroll_n) using the scroll_down_n function.
  - Extract information from the map within the given latitude and longitude range using the get_info function. This function takes parameters for the coordinates, driver, number of results to fetch, category, and sub-category.



Connecting Population

Grid Division and Data Collection



Define the Rectangle:

- Identify the southwest corner (latitude1, longitude1) and northeast corner (latitude2, longitude2) of the rectangle.
- These points set the boundaries for the area to be analyzed.

Grid Division:

- Divide the rectangle into an n*n grid of blocks.
- Calculate the size of each grid cell:

```
cell_height = (latitude2 - latitude1) / n
cell_width = (longitude2 - longitude1) / n
```

Map Coordinates to Grid:

- Convert latitude and longitude to corresponding rows and columns:

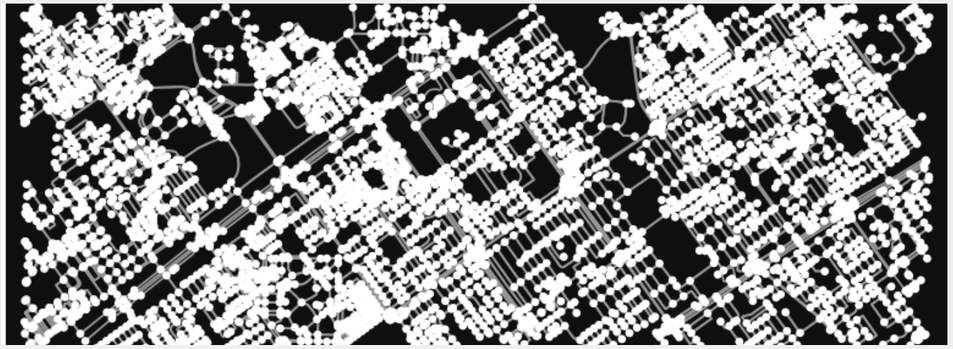
```
row = (lat - latitude1) / ((latitude2 - latitude1) / n)
col = (lon - longitude1) / ((longitude2 - longitude1) / n)
```

OSMnx Data Collection:

- For each grid cell center, create a circle with radius r.
- Use OSMnx to fetch all data points within each circle, assuming the number of data points is proportional to the population.

Connecting Population

Population Calculation



Grid Cell Centers:

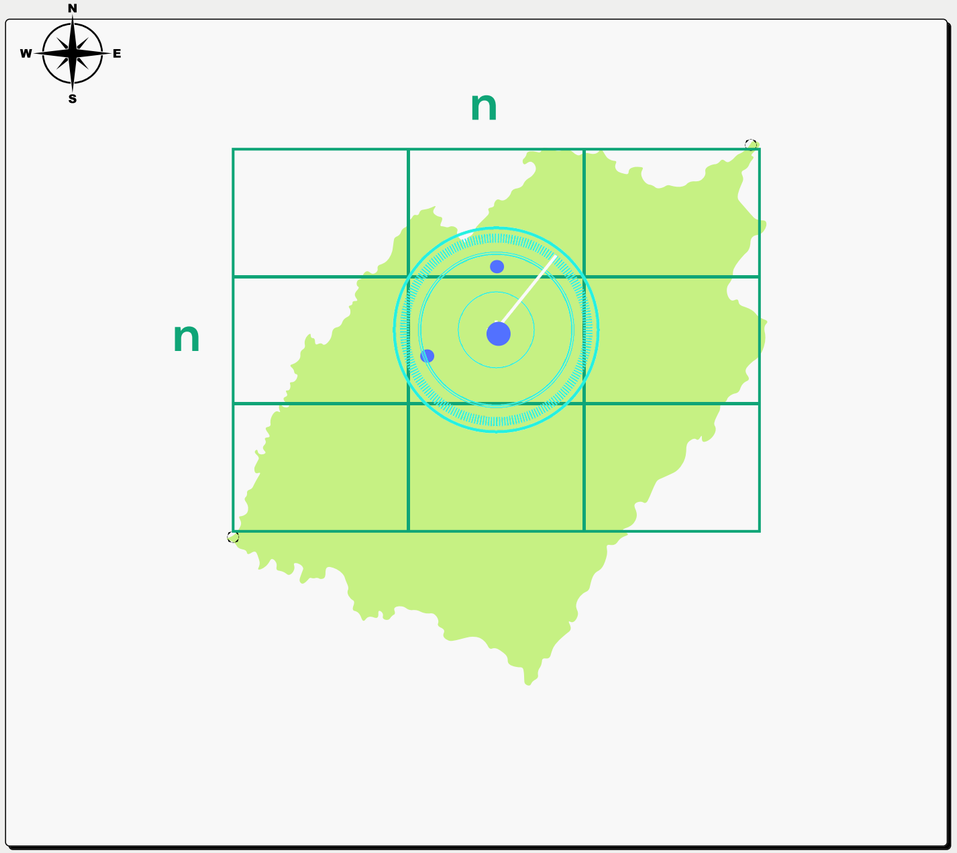
- We find the center point of each square in our grid.

Data Points Around Centers:

- We use a tool called osmnx to find interesting points (like buildings) near each center point, within a circle of radius 'r'. This circle size controls how far we look for people.
- We assume that more points around a center mean there are more people in that area.

Connecting Population:

- We consider not just the center point, but also the points within the circle.
- The farther a point is from the center, the less it counts towards the population for that square. This is because areas closer to the center are likely more crowded.



Connecting Population

Introduction to Convolution and Kernel

- **Convolution:** A mathematical operation that blends two functions to produce a third function.
- **Kernel Matrix:** Defines weights applied to data points based on proximity to a grid cell center.

Applying Convolution for Population Density

Convolution Process:

- Iterate over each grid cell.
- For each cell, iterate over its data points.
- Calculate the distance from each data point to the cell's center.
- Apply the corresponding weight from the kernel matrix.
- Sum the weighted data points and normalize by the total weight to compute the population density for the grid cell.

Kernel Matrix:

0.2,	0.2,	0.2,	0.2,	0.2,	0.2,	0.2
0.2,	0.4,	0.4,	0.4,	0.4,	0.4,	0.2
0.2,	0.4,	0.6,	0.6,	0.6,	0.4,	0.2
0.2,	0.4,	0.6,	0.8,	0.6,	0.4,	0.2
0.2,	0.4,	0.6,	0.6,	0.6,	0.4,	0.2
0.2,	0.4,	0.4,	0.4,	0.4,	0.4,	0.2
0.2,	0.2,	0.2 ,	0.2,	0.2,	0.2,	0.2

Benefits of Using Convolution

- **Accurate Density Estimation:** Reflects population distribution more realistically by considering spatial-population proximity.
- **Scalability:** Adaptable to varying densities and sizes of data points within each grid cell.

Bayesian Approach to Rating Computation

Problem Statement: Suppose we want to got to restaurant we will check the rating's in google so we will choose the best place with more rating and with the number of the people who had written the reviews

To Solve this Problem we are going to use the **Bayesian Approach**

Formula:

$$\text{Rating} = \frac{(W \cdot R) + \prod_{i=1}^{n} r_i}{W + n}$$

For any length **n** and any score no only for the rating review this problem can be used to solve foir enhancing the reliability and balance of ratings for items with varying numbers of user reviews.

Source: Stackoverflow

Feature Engineering?

The process of using domain knowledge to create features that make machine learning algorithms work better.

Purpose:

- Enhancing Predictive Power
- Reducing Noise
- Handling Missing Values

OUTPUT:

```
Hotel-Hostelsforstayingnearme
FuelStation-PetrolPump
Office-InsuranceAgency
EducationalInstitute-DrivingSchool
```

Original Features:

Category: The main category of a location (e.g., "Restaurant").

Sub-Category: A specific type within the category (e.g., "FastFood").

Combined Feature:(After Feature Engineering)

Category-Subcategory: A merged feature (e.g., "Restaurant-FastFood").

Hyperparameter Tuning with Data Filtering

Define the Rectangle:

- **Objective:** Filter and save subsets of a dataset based on specific row and column values.
- **Context:** Hyperparameter tuning in machine learning involves systematically exploring various configurations to find the optimal set of parameters.

Process:

- Load a CSV file containing the data into a DataFrame.
- Set the range for the rows and columns to be iterated over.
- Create a folder to store the filtered CSV files.
- Iterate over each combination of row and column values.
- Filter the DataFrame to include only the rows and columns matching the current iteration values.
- Save the filtered DataFrame to a CSV file if it is not empty.

Outcome:

- The filtered datasets are saved as CSV files in the specified folder, facilitating the evaluation of different data subsets similar to hyperparameter tuning in machine learning. This allows for a systematic exploration of various configurations to identify the optimal setup.

Understanding K-Nearest Neighbors (KNN)

What is KNN?

Supervised Machine Learning algorithm used for classification and regression tasks.
It operates on the principle that similar exist closer to each other.

How does KNN work?

Step 1: Choose the number of neighbors (K).
Step 2: Calculate the distance between the query point and all points in the dataset using a distance metric
(e.g., Euclidean distance).
Step 3: Identify the K-nearest neighbors to the query point.
Step 4: For classification, assign the class with the majority vote among the K neighbors.
For regression, return the average of the K neighbors' values.

Applying KNN for Distinct Feature Selection

Objective:

- Use KNN to find distinct features based on input data, incorporating custom distance metrics to handle various factors like geographical location, weight, and rating.

Custom Distance Metric:

- Defined a custom distance metric that combines geographical distance, weight factor, and rating factor.
- Formula :

$$\text{Custom Distance} = \left(\frac{w \cdot r + \left(\text{weight\_factor} \cdot \text{distance} \cdot \frac{1}{\text{rating\_factor}} \right)}{r + (\text{weight\_factor} \cdot \text{distance})} \right)$$

- Adjusted distance using weights and ratings to ensure meaningful comparisons.

Implementation Steps:

- Data Preparation: Loaded multiple datasets containing features like Latitude, Longitude, weights, and FinalRating.
- KNN Model Fitting: Used the custom distance metric to fit the KNN model on the dataset.
- Finding Distinct Features: Queried the model to find the K-nearest distinct features for a given input feature ('Restaurant-FastFood').
- Results: Extracted the top K distinct features and saved the results in a CSV file

Why KNN for this task?

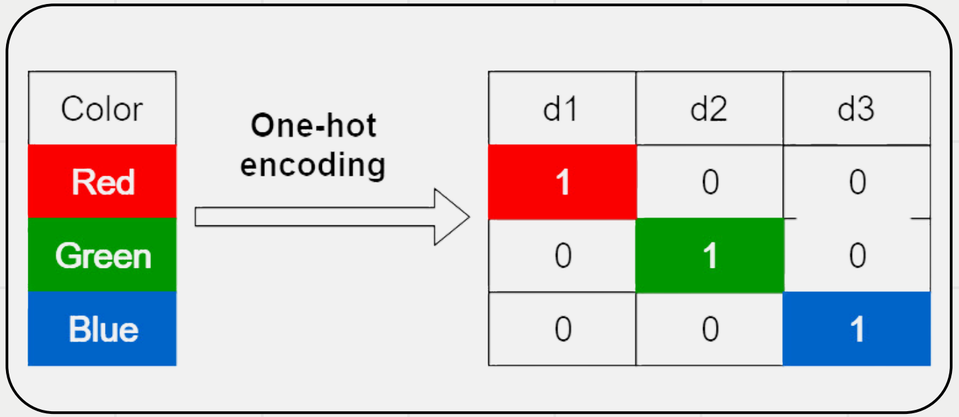
- Versatility: Capable of handling complex, multidimensional data.
- Customizability: Easily adaptable to use custom distance metrics.
- Effectiveness: Efficiently identifies the nearest neighbors and distinct features based on multiple factors.

K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is used in the provided code to identify distinct features based on geographical coordinates, weights, and ratings. The process starts with extracting relevant columns (Latitude, Longitude, weights, and FinalRating) from the dataset. A custom distance metric combines geographical distance with weights and ratings, ensuring both spatial proximity and feature attributes are considered. The KNN model is then fitted using this custom metric. For a given input feature, the KNN model finds the nearest neighbors with the smallest distances. Distinct features are selected from these neighbors, aiming to identify k unique features closest to the input feature, scored based on their proximity (1 - score). The results are aggregated and saved to a CSV file for further analysis.

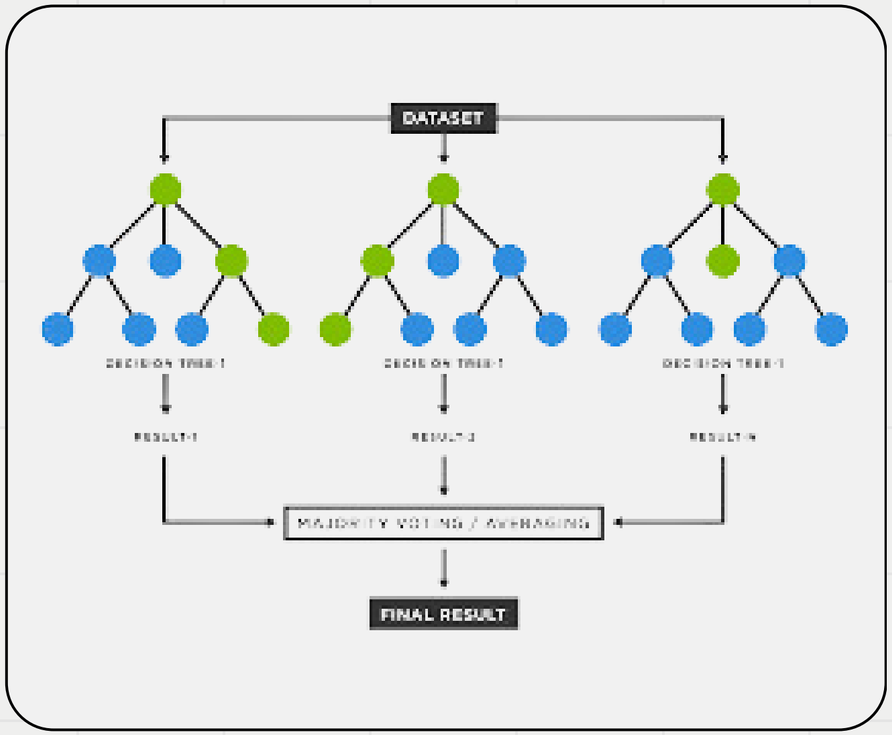
One-Hot Encoding

It is used to transform the categorical data into a numerical format. In this method it converts each unique category value into a new binary column (feature), where the presence of a category is marked with (“1") and the absence is marked with (“0"). For example, if a dataset includes a 'Category-Subcategory' column with values like 'Restaurant-FastFood' and 'Restaurant-DineIn', one-hot encoding will create separate binary columns for each unique value. In our Model, one-hot encoding is applied to the 'Category-Subcategory' column, resulting in a set of binary features that are then used as inputs to a Random Forest Regressor. This helps us to allows the model for predicting the 'score'.



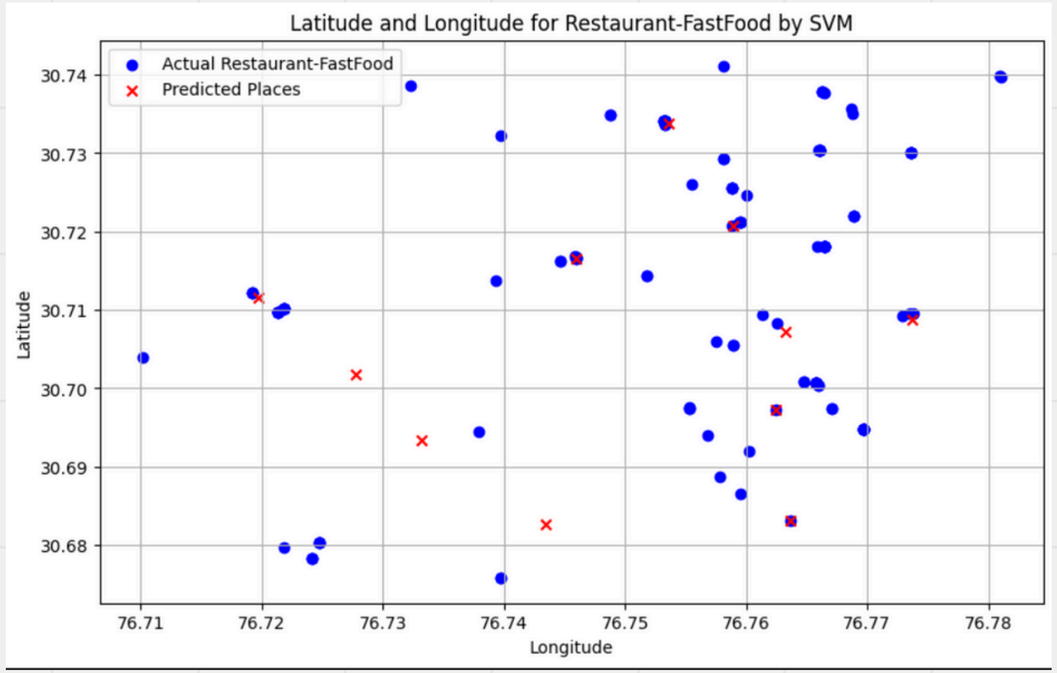
Random Forest

Random Forest Regressor is used to determine the importance of features in predicting the 'score' from a dataset. Initially, it loads the data and applies one-hot encoding to the **'Category-Subcategory'** column, transforming categorical values into numerical features. These encoded features are then combined with the original dataset, excluding the 'Category-Subcategory' column. The features (X) and the target variable (y), which is the 'score', are separated. The Random Forest Regressor is trained on this dataset, utilizing multiple decision trees to aggregate predictions and improve accuracy. Feature importances are then extracted from the trained model, indicating the contribution of each feature to the prediction. By doing this the model selectes the top 4 'Category-Subcategory' features, for predicting the 'score'.



Support Vector Machines(SVM)

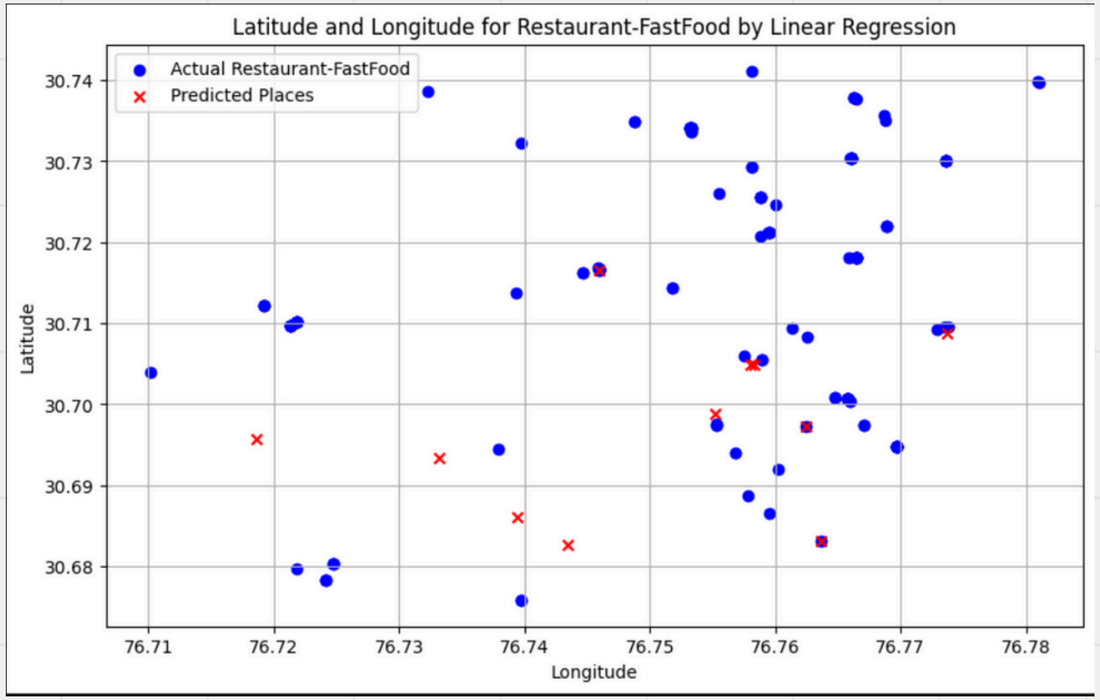
Support Vector Regression (SVR) is a supervised learning method used for prediction tasks. In our model, SVR predicts 'FinalRating' based on various location and demographic features. After training, the model identifies the top 10 locations with the highest predicted ratings by extracting the indices of the highest predicted values and mapping them back to their corresponding Latitude and Longitude. This process pinpoints key locations based on the model's learned relationships between features and the target variable.



Linear Regression

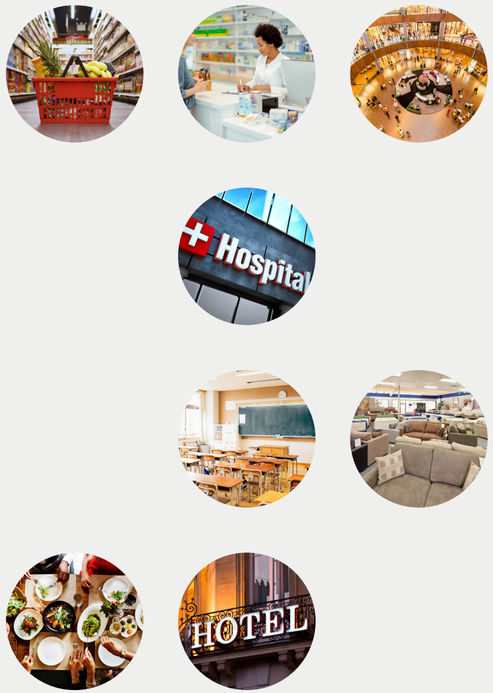
Linear Regression, a supervised learning technique, is used to predict a continuous target variable ('FinalRating' in this case) by fitting a linear equation to the data. This equation assigns weights (coefficients) to different features (Latitude, Longitude, etc.) to understand their collective influence on the target variable.

In this project, the Linear Regression model is trained on a training set to learn the relationships between features and 'FinalRating'. After training, it predicts 'FinalRating' for a test set. The top 10 locations with the highest predicted ratings are then identified by mapping the highest predicted values back to their geographic coordinates (Latitude and Longitude)



Goal Items

1. Get a structured city layout
2. Mobility and less chaos interlocation



Breaking the Problem

Predict and deploy coordinate for each category occurrences.

Feature Extraction

features also include category and subcategory search query

Google Maps features include Name, Rating, Number of Reviews, latitude, and longitude

ML Predictive Model

Train machine learning models to predict optimal coordinates

Deployment

Create for inputting categories and receiving optimal coordinates. Integrate the model with a mobile city layout

Thank You

Testing Finnished Accuracy: