Report

Here I took the datapoints of commercial places of New York, United States. Earlier I was planning to choose Prayagraj-my hometown but eventually choose New York- a big city because of the availability of huge number of data. The following dataset is collected by taking reference from the given links.

- https://towardsdatascience.com/loading-data-from-openstreetmap-with-python-and-the-overpass-api-513882a27fd0
- https://wiki.openstreetmap.org/wiki/Overpass_API/Overpass_QL#By_tag_.28has-kv.29
- https://mygeodata.cloud/osm/data/download/

Link to my dataset: https://drive.google.com/file/d/1QgGJfIgFLkDDK0DtKenW7vLfkGLHJEy2/view?usp=sharing

Here we have 34 features including latitude and longitude. We also have 32 other features but most of them are unfilled(**NaN**). I had dropped it later.

I have kept a criteria for dropping columns- If the column has more than 55% of it's value unfilled, then I will drop it. After dropping most of the irrelevant features, I was left with five important features. They were latitude, longitude, name, street and market type.

As name and street columns are almost unique(please refer to Jupyter Notebook) so we didn't use them for our purpose. These two fields acts only as extra information.

Market_type column has six unique values but there was a huge difference between its maximum and minimum datapoints count(714 and 1). To make my dataset less imbalanced, I have merge the five other datapoints to make it a single class. So the labels I got after were binary(supermarket and notSupermarket).

Even after doing, I still didn't see our datapoints in not good proportion.If we use market_type as label to classify our datapoints and keep our metric accuracy, then it won't be a good classification as our dataset is imbalanced.

Also I have clustered the data based on nearest neighbour concept. I took two features i.e latitude and longitude to cluster and as these two are location points and the algorithm which is best for this type of clustering is KMeans. Here I kept K=4. If you ask

me why four, then i will say that it's an assumption formed after the visualisation. That's why visualisation is done before predicting anything.

After getting the clusters, I have to classify the points based on commercial labels. I have already shown how I collected labels and how did I merge it to get binary labels. Even after merging, my labels are still unbalanced.

Here I have used **Decision Tree Classifier** to classify datapoints. As the datapoints are unbalanced, that's why I have used **F1_Score** instead of accuracy to measure the goodness of my model.

F1 score - **F1 Score** is the weighted average of Precision and Recall. Therefore, this **score** takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but **F1** is usually **more** useful than accuracy, especially if you have an uneven class distribution.

For this dataset, I have got the F1_score of 0.85.