Coursera Capstone –
Vienna Neighbourhood Recommender

Content

Introduction

This project aims on building a simple recommender for neighbourhoods. It should take data about how the neighbourhood should look like and return the neighbourhoods that fit the criteria best. The information that is provided to the recommender will contain features like areas density of different venues, as well as trending venues. The latter will be taken from the foursquare places API. It will also contain the average price to rent a flat, as well as the population density.

Business Problem

We are developing this project for a client that is in the online estate agent business. Our client website allows the user to search for flats for rent in a certain area the visitor must choose before. Since there are a lot of neighbourhoods, many visitors face problems to find out which ones they could like. Especially for people that move to a new city this can be hard and take a lot of time.

To outperform competing websites, our client wants to add a recommendation engine that first asks the visitor a few questions about his preferences on how his preferred neighbourhood should look like (e.g. how important are parks, restaurants, etc.) and what his budget is. Based on this data, a list or neighbourhoods should be provided that would suit the visitors needs best (ordered by relevance).

In the first step of development we need to prove the feasibility. This will be done by creating a jupyter notebook that shows how to build a recommender based on XGBoost and only data for Vienna. All further steps that are necessary provide the recommender on the client's website are not part of this project and will be realised in follow up projects.

Data Inputs

To provide enough high-quality features for our predictor to meet the client's requirements, we need to access three different data sources. The source and the structure of the data as well as its usage are described in the subsections below.

Open Street Maps (Overpass)

Our most important data source will be Open Street Maps (OSM). Most people know this open source community mapping project for providing free online maps. For many regions of the world, the project also features highly accurate data about which places (e.g. amenities like park benches, restaurants, etc.) exist there. The feature that makes OSM extremely valuable for data science is an API called overpass, which allows us to access all the information of OSM from Python via a simple query language.

In this project, we will first use OSM data to retrieve the boundaries of the city's districts. We can download these boundaries in json. We will use the names of the boundaries as well as the corresponding ID's to download data about places in the single districts. We will also use the information about boundaries for visualization via folium. At this point some data wrangling and conversion will be necessary to create a valid geojson file.

Next we will download data about places in our districts. Referencing a district boundaries ID (relation ID), overpass allows us to get all items that are placed within these boundaries. We will pick 6 types of places (parks, restaurants, universities, dust bins, doctors and supermarkets) for our feasibility study and download them for each district. We then group them by district and calculate their density based on the districts area, which we can calculate from the geojson of the boundaries.

Immopreise.at

The second data source we will use is a website called https://www.immopreise.at which features lists of flat prices for all districts in Austria. We will download data about Vienna's districts manually and import them from .csv because we don't expect them to change regularly. The each of the districts we populated with data about places before, we will add a column containing its average flat rental prices per square meter.

Foursquare

The third source of data we will use is the foursquare places API. From there we will download the top trending venues of the category **food** for each of the districts and add it to our existing feature set. We will use the districts centroid point as reference and get all trending venues within a radius of 2000m.

Methodology

Most of the data exploration step was done beforehand using the open street maps online service because this is a very convenient way to get an overview on the structure of the available data in real time.

The first step to accomplish our task is to gather all the data we need. We get the bounds Vienna's districts via from Open Street Maps via the overpass API. Based on the spatial data, we then retrieve trending restaurants from the foursquare places API. We also gather data about parks, doctors, universities, dust bins, restaurants and supermarkets from overpass.

We will use XGBoost as machine learning algorithm for predicting the district that fits our preferences best. It's a little bit uncommon to use a gradient boosting algorithm for this kind of task, but I believe that this can be a quite convenient, fast and especially very scalable way to build a multiclass recommendation system, based on a small set of samples. Since we train a new classifier for each of the districts which just predicts how well this one district fits our preferences, scaling up only means to add one new classifier per district, resulting in a very predictable amount of memory and computation usage.

After data wrangling, we train a OneVsAll classifier based on XGBoost. This means that we train a binary classifier for each of our districts. This enables us to predict whether each district meets our preferences which are represented by a construed district.

At the end we test our classifier with a construed district and visualize our results.

Results

The result we get when we run our classifier on a construed test district that represents our test preferences, we get a list of districts and their specific probability to fit the test district. The list can look like this:

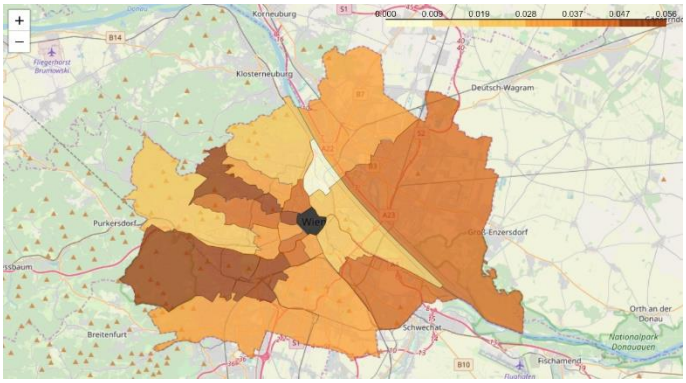
```
In [54]: df_weighted_districts
```

Out[54]:

	d_ref	weight	d_name	d_price
0	13	0.055909	Hietzing	15.24
2	12	0.048920	Meidling	14.37
3	8	0.048714	Josefstadt	15.59
4	17	0.047813	Hernals	14.55
5	5	0.046351	Margareten	15.50
6	6	0.045877	Mariahilf	16.42
7	9	0.040782	Alsergrund	16.27
8	11	0.040498	Simmering	14.10
9	22	0.038491	Donaustadt	16.00

As we can see, in for our test preferences, the 13th district of Vienna would fit best, followed by the 12th and so on. In our list we also included the prices, although we already dropped the lines with prices that were too high.

We can now visualize our result using folium:



That looks quite realistic at the first glance and we can clearly see that the darker a district is, the closer it fits our preferences. The black district in the middle is irrelevant since its average price per square meter is too expensive for our example user.

As we have a complex feature set, it will be hard to check how relevant our result really is. Before we could go from exploration to production, we would need to spend more time on model validation.

Discussion

Getting the data from foursquare and OSM together required quite some data wrangling, but pandas does the job quite well. After data wrangling, creating the classifier was quite easy. Fortunately the OneVsRestClassifier class from sklearn does all the heavy lifting when building a multiclass classifier with XGBoost. We could have also used a normal DecisionTreeclassifier since it can do multiclass predictions out of the box, allowing to create only one classifier for the whole dataset. I didn't choose this apporach because it would have only been possible to return the one district that fits best. Also scalability would have been an issue with this approach.

The results we got from our test case seem to be quite good and they allow us to clearly visualize which district might suite the user best. Anyway, further improvements in feature scaling must be made. To

create a usable product, it would be necessary to think about a way how to express the user's preferences so that they correspond with the scale of the dataset we trained our predictor on.

I also think that if this was a real product, we would have to think about whether districts are really a good bounding area for retrieving places. If we were working with an estate agency, we most likely would be able to get a list of all flats they offer. Our predictions would be much more interesting if we could use the single flats coordinates and a specified radius as bounding area for data retrieval. This way, the user would get very precise recommendation on which flat would fit his needs best.

Conclusion

Getting data from OSM and foursquare is quite convenient. Especially OSM offers an incredible amount of data for free (if you bring enough time). The ability to create a .geojson file from any data brings us a bunch of new opportunities for using folium choropleth maps.

Finally, I can say, that working with geo data is a lot of fun and offers many opportunities. The approach we took is quite unusual because we only have one sample per class and many classes, but I think it can be quite useful.