Coursera Capstone –

Vienna Neighbourhood Recommender

# 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](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