Restaurants and cafes in Stuttgart

Influence of location and type of cuisine on future ratings

Report

1. Introduction of the problem

For sure, the ratings of a restaurant are highly dependent on the quality of the food, the service and the whole atmosphere.
BUT beside these criteria of a running business, are there fundamental conditions I can set before opening a business which are influencing the future rating?
E.g. before I'm opening a restaurant business I have to decide

- where to locate it and
- which kind of cusine I want to offer.

  When taking the decision I should consider that this cuisine is demanded by customers in this area of the city.

  Of course, I can't look in the future or asking all the residents, but I can check how the existing businesses perform.

Target audience:
Future business owner

2. Data

I want to investigate this problem for the city of Stuttgart in Germany.

- As level of detail on the locations I'm choosing the ZIP-codes in the city area. This ZIP-code areas should sufficient enough to represent the different neighborhoods --> Web-source

In [103]:
```
df_zip_merged.head()
```

Out[103]:

	PLZ	Stadtteil	Bad Cannstatt	Botnang	Feuerbach	Frauenkopf	Stuttgart-Mitte	Stuttgart-Nord	Stuttgart-Ost
0	70173	Stuttgart-Mitte	0	0	0	0	1	0	0
1	70174	Stuttgart-Mitte	0	0	0	0	1	0	0
2	70174	Stuttgart-Nord	0	0	0	0	0	1	0
3	70174	Stuttgart-West	0	0	0	0	0	0	0
4	70176	Stuttgart-Mitte	0	0	0	0	1	0	0

- For each ZIP-code area I can pull the geo coordinates to get reference points per neighborhood. --> GEOPY

In [104]:
```
df_coords.head()
```

Out[104]:

	PLZ	Latitude	Longitude
0	70173	48.777845	9.178425
1	70174	48.782793	9.169334
2	70176	48.777425	9.161045
3	70178	48.769172	9.167613
4	70180	48.764008	9.174807

- With the geo coordinates I can get a list of food-related businesses from Foursquare which are close to each neighborhood

In [105]:
```
df_venues.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 731 entries, 0 to 730
Data columns (total 9 columns):
PLZ                 731 non-null int64
Latitude            731 non-null float64
Longitude           731 non-null float64
Venue_id            731 non-null object
Venue_name          731 non-null object
Venue_latitude      731 non-null float64
Venue_longitude     731 non-null float64
Venue_PLZ           731 non-null object
Venue_category      731 non-null object
dtypes: float64(4), int64(1), object(4)
memory usage: 51.5+ KB
```

In [106]:
```
df_venues.head(5)
```

Out[106]:

	PLZ	Latitude	Longitude	Venue_id	Venue_name	Venue_latitude	Venue_
0	70173	48.777845	9.178425	4b1ce1eff964a520410a24e3	Oggi Tavola Mediterranea	48.778835	
1	70173	48.777845	9.178425	4be5b3e42457a59389dcab15	Sushi-Ya	48.777461	
2	70173	48.777845	9.178425	5924978f8173cb087c0b283d	Sansibar	48.775331	
3	70173	48.777845	9.178425	4d958e319079b1f7e5d8e309	Cafe Nast	48.776088	
4	70173	48.777845	9.178425	5465f08a498e762c1d8cf899	Herr Kächele: Maultaschen und Mehr	48.775616	

- For exotic cuisine I won't have enough data that's why I'm focusing on the top10-cuisines only.

In [107]:
```
df_top10cat
```

Out[107]:

	Venue_category	Appearances
9	Café	54
29	Italian Restaurant	51
25	German Restaurant	49
4	Bakery	46
7	Burger Joint	13
2	Asian Restaurant	11
60	Turkish Restaurant	11
53	Sushi Restaurant	11
19	Fast Food Restaurant	11
40	Pizza Place	10

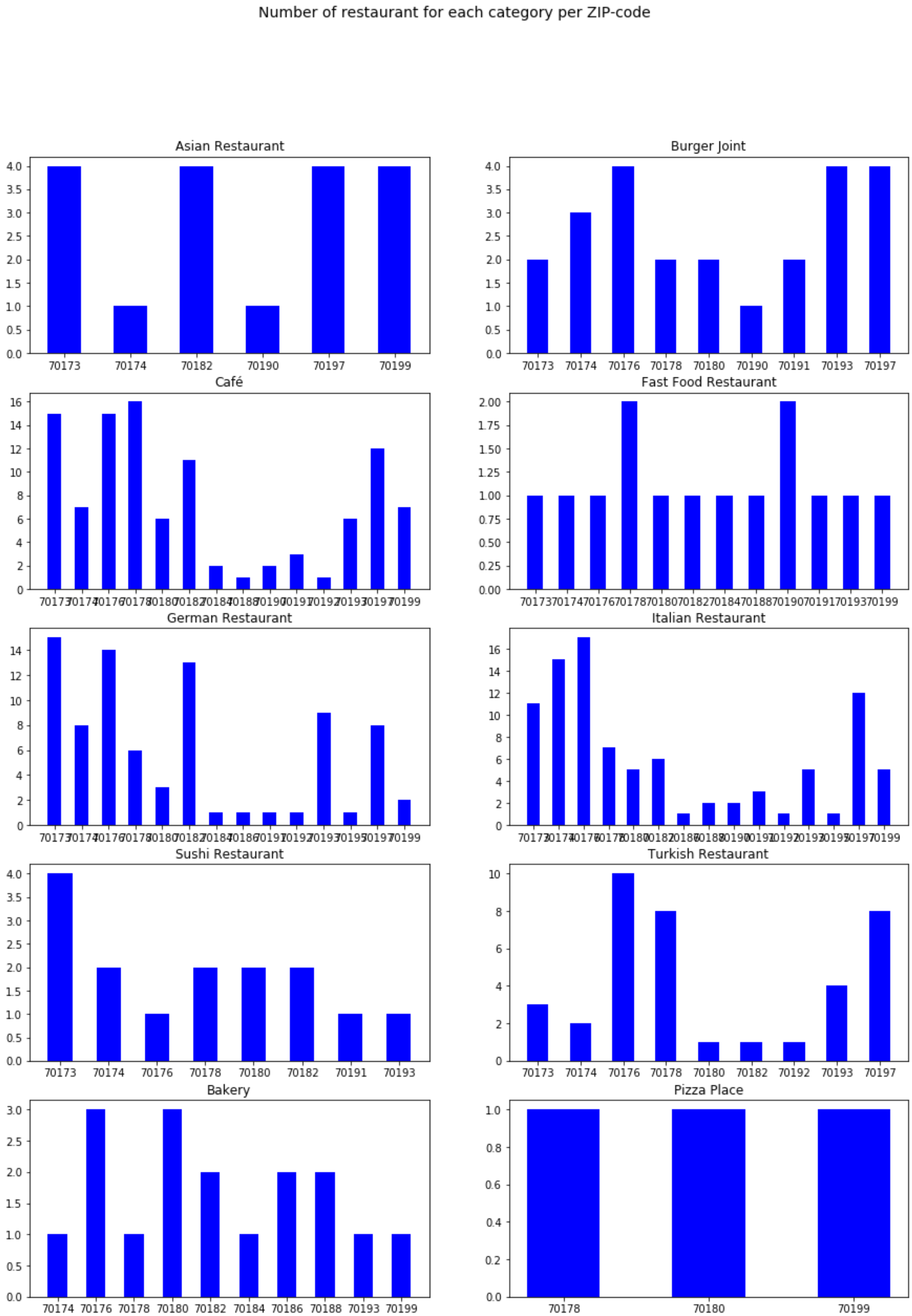
- For each food-related business I'm pulling the rating from Foursquare.

In [145]:
```
df_venues_con.head(5)
```

Out[145]:

	Venue_id	Venue_name	Venue_latitude	Venue_longitude	Venue_PLZ	Venue_
1	4b1ce1eff964a520410a24e3	Oggi Tavola Mediterranea	48.778835	9.176266	70173 Stuttgart	F
2	4b44c3f8f964a52088fb25e3	Vapiano	48.780076	9.177558	70173 Stuttgart	F
3	4b4823d9f964a520f64826e3	Bierhaus West	48.781574	9.165426	70174 Stuttgart	F
4	4b48a6a8f964a520a35126e3	Alte Kanzlei	48.777640	9.178852	70173 Stuttgart	F
6	4b4c5dfef964a520aab126e3	Stern Kebap	48.789112	9.195198	Deutschland	F

In [146]:
```
fig, ax = plt.subplots(5, 2, figsize=(15,20))
fig.suptitle("Number of restaurant for each category per ZIP-code", fontsize=1
4)

ax[0][0].bar(x_bins_cat0, series_cat0, color='b', width=0.5)
ax[0][0].set_title(str(cat0))

ax[0][1].bar(x_bins_cat1, series_cat1, color='b', width=0.5)
ax[0][1].set_title(str(cat1))

ax[1][0].bar(x_bins_cat2, series_cat2, color='b', width=0.5)
ax[1][0].set_title(str(cat2))

ax[1][1].bar(x_bins_cat3, series_cat3, color='b', width=0.5)
ax[1][1].set_title(str(cat3))

ax[2][0].bar(x_bins_cat4, series_cat4, color='b', width=0.5)
ax[2][0].set_title(str(cat4))

ax[2][1].bar(x_bins_cat5, series_cat5, color='b', width=0.5)
ax[2][1].set_title(str(cat5))

ax[3][0].bar(x_bins_cat6, series_cat6, color='b', width=0.5)
ax[3][0].set_title(str(cat6))

ax[3][1].bar(x_bins_cat7, series_cat7, color='b', width=0.5)
ax[3][1].set_title(str(cat7))

ax[4][0].bar(x_bins_cat8, series_cat8, color='b', width=0.5)
ax[4][0].set_title(str(cat8))

ax[4][1].bar(x_bins_cat9, series_cat9, color='b', width=0.5)
ax[4][1].set_title(str(cat9))

plt.show()
```



- My dataset consists of "rating" (as target) and "close neighborhoods" and "cuisine" (as features)

In [148]:
```
df_str_group.head()
```

Out[148]:

	Venue_id	Venue_name	Venue_category	Rating	70173	70174	70176	7017(
0	4b1ce1eff964a520410a24e3	Oggi Tavola Mediterranea	Italian Restaurant	8.6	1	1	0	(
1	4b44c3f8f964a52088fb25e3	Vapiano	Italian Restaurant	6.9	1	1	0	(
2	4b4823d9f964a520f64826e3	Bierhaus West	German Restaurant	6.6	0	1	1	(
3	4b48a6a8f964a520a35126e3	Alte Kanzlei	German Restaurant	7.0	1	0	0	(
4	4b4c5dfef964a520aab126e3	Stern Kebap	Fast Food Restaurant	6.1	0	0	0	(

5 rows × 21 columns

3. Methodology

Methodology:
I'm choosing the Linear Regression-method to predict the ratings, because the coefficients for each feature will show me if this feature plays a crucial role and it's worth considering.

4. Results

--> but the MSE is quite small with 0.66

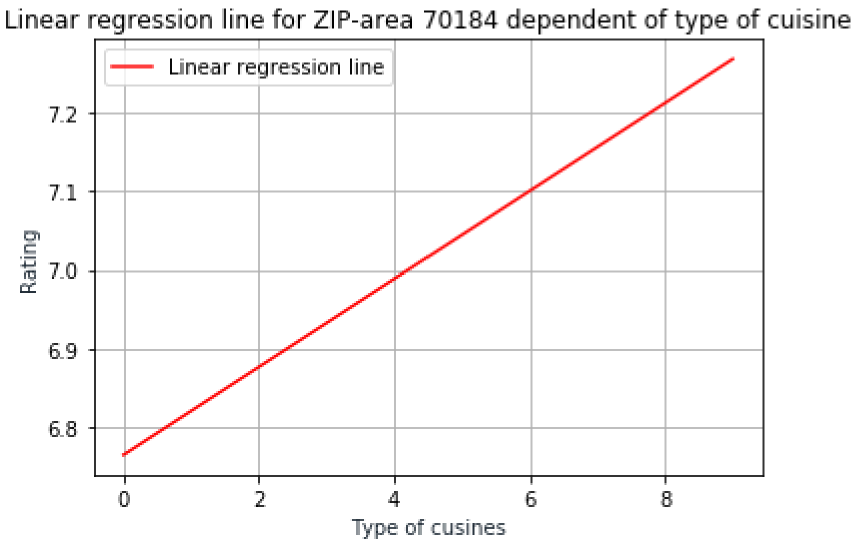
Let's have a look on the coefficients

In [151]:
```
df_coeff_lr.head(30)
```

Out[151]:

	Coefficients
Feature	
70184	0.798
70193	0.765
70173	0.629
70188	0.531
70199	0.290
70178	0.242
70180	0.196
Num_category	0.056
70190	0.022
70176	0.001
70192	-0.013
70182	-0.128
70191	-0.132
70174	-0.396
70186	-0.437
70197	-0.446
70195	-1.865

In [152]:
```
plt.plot(x_values, y_lin, '-r', label='Linear regression line')
plt.title('Linear regression line for ZIP-area ' + str(df_coeff_lr.index[0]) +
" dependent of type of cuisine")
plt.xlabel('Type of cusines', color='#1C2833')
plt.ylabel('Rating', color='#1C2833')
plt.legend(loc='upper left')
plt.grid()
plt.show()
```



Linear regression line shows the likely range of a restaurant rating in ZIP-area 70184.
ZIP-area is the area, where the restaurants have the best rating

> min. rating is 6.75
> max. rating is 7.3
> type of cusine has only limited impact as the MSE is 0.66

5. Discussion

Unfortunately the results are not really significant due to limited data.
It would be very helpful to have the number of ratings since existance of the restaurant.
But at least the results show in which areas are the best restaurants.

6. Conclusion

Conclusion:
The impact of location and cusine on the rating is not significant enough to be considered as crucial criteria

NOTEBOOK

In [1]:
```
import pandas as pd
import numpy as np
from geopy.geocoders import Nominatim # convert an address into latitude and l
ongitude values
from geopy.distance import geodesic

import requests # library to handle requests
from pandas.io.json import json_normalize # tranform JSON file into a pandas d
ataframe

# Matplotlib and associated plotting modules
import matplotlib.cm as cm
import matplotlib.colors as colors
```

1. Hoods in Stuttgart, GER

1.1 Load ZIP-codes in Stuttgart city

In [2]:
```
df_zip = pd.read_excel("701_Stuttgart_ZIP.xlsx", header = 0)
df_zip.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
PLZ          30 non-null int64
Stadtteil    30 non-null object
dtypes: int64(1), object(1)
memory usage: 560.0+ bytes
```

In [3]:
```
df_zip.head()
```

Out[3]:

	PLZ	Stadtteil
0	70173	Stuttgart-Mitte
1	70174	Stuttgart-Mitte
2	70174	Stuttgart-Nord
3	70174	Stuttgart-West
4	70176	Stuttgart-Mitte

In [4]:
```
# No distinct ZIP-codes -> One Hot Encoding of hoods
df_onehot_hoods = pd.get_dummies(df_zip["Stadtteil"])
df_onehot_hoods.head()
```

Out[4]:

	Bad Cannstatt	Botnang	Feuerbach	Frauenkopf	Stuttgart-Mitte	Stuttgart-Nord	Stuttgart-Ost	Stuttgart-Süd	Stuttg W
0	0	0	0	0	1	0	0	0	
1	0	0	0	0	1	0	0	0	
2	0	0	0	0	0	1	0	0	
3	0	0	0	0	0	0	0	0	
4	0	0	0	0	1	0	0	0	

In [5]:
```
df_onehot_hoods.shape
```

Out[5]: (30, 9)

In [6]:
```
df_zip_merged = df_zip.join(df_onehot_hoods, how="left")
df_zip_merged.head()
```

Out[6]:

	PLZ	Stadtteil	Bad Cannstatt	Botnang	Feuerbach	Frauenkopf	Stuttgart-Mitte	Stuttgart-Nord	Stuttgart-Ost
0	70173	Stuttgart-Mitte	0	0	0	0	1	0	0
1	70174	Stuttgart-Mitte	0	0	0	0	1	0	0
2	70174	Stuttgart-Nord	0	0	0	0	0	1	0
3	70174	Stuttgart-West	0	0	0	0	0	0	0
4	70176	Stuttgart-Mitte	0	0	0	0	1	0	0

In [7]:
```
df_zip_merged.shape
```
Out[7]: (30, 11)

In [8]:
```
#remove col "Stadtteil"
df_zip_merged.drop("Stadtteil", inplace=True, axis=1)
```

In [9]:
```
df_zip_merged.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 10 columns):
PLZ                  30 non-null int64
Bad Cannstatt        30 non-null uint8
Botnang              30 non-null uint8
Feuerbach            30 non-null uint8
Frauenkopf           30 non-null uint8
Stuttgart-Mitte      30 non-null uint8
Stuttgart-Nord       30 non-null uint8
Stuttgart-Ost        30 non-null uint8
Stuttgart-Süd        30 non-null uint8
Stuttgart-West       30 non-null uint8
dtypes: int64(1), uint8(9)
memory usage: 590.0 bytes
```

In [10]:
```
# Distinct ZIPs
df_zip_grouped = df_zip_merged.groupby(["PLZ"]).sum().reset_index()
df_zip_grouped.head()
```

Out[10]:

	PLZ	Bad Cannstatt	Botnang	Feuerbach	Frauenkopf	Stuttgart-Mitte	Stuttgart-Nord	Stuttgart-Ost	Stuttgart-Süd
0	70173	0	0	0	0	1	0	0	0
1	70174	0	0	0	0	1	1	0	0
2	70176	0	0	0	0	1	0	0	0
3	70178	0	0	0	0	1	0	0	1
4	70180	0	0	0	0	1	0	0	1

In [11]:
```
df_zip_grouped.shape
```
Out[11]: (16, 10)

1.2 Get geo coords for ZIP-codes

In [12]:

```
# Function to pull geo coordinates based on ZIP codes
def getCoordsSTR(zip_codes):
    address_suffix = " Stuttgart, Germany"
    address_zip = ""
    list_coords = []

    for code in zip_codes:
        address_zip = str(code)
        address = address_zip + address_suffix

        geolocator = Nominatim(user_agent="foursquare_agent")
        location = geolocator.geocode(address)
        latitude = location.latitude
        longitude = location.longitude

        list_coords.append([code, latitude, longitude])

    df_list = pd.DataFrame(list_coords, columns=["PLZ", "Latitude", "Longitud
e"])

    return(df_list)
```

In [13]:

```
df_coords = getCoordsSTR(df_zip_grouped["PLZ"])
df_coords.head()
```

Out[13]:

	PLZ	Latitude	Longitude
0	70173	48.777845	9.178425
1	70174	48.782793	9.169334
2	70176	48.777425	9.161045
3	70178	48.769172	9.167613
4	70180	48.764008	9.174807

In [14]:
```
df_str_geo = pd.merge(df_zip_grouped, df_coords, how="left", on="PLZ")
df_str_geo.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16 entries, 0 to 15
Data columns (total 12 columns):
PLZ                  16 non-null int64
Bad Cannstatt        16 non-null uint8
Botnang              16 non-null uint8
Feuerbach            16 non-null uint8
Frauenkopf           16 non-null uint8
Stuttgart-Mitte      16 non-null uint8
Stuttgart-Nord       16 non-null uint8
Stuttgart-Ost        16 non-null uint8
Stuttgart-Süd        16 non-null uint8
Stuttgart-West       16 non-null uint8
Latitude             16 non-null float64
Longitude            16 non-null float64
dtypes: float64(2), int64(1), uint8(9)
memory usage: 656.0 bytes
```

1.3 Get distances between ZIPs

In [15]:
```
df_source = df_str_geo.loc[:, ["PLZ"]]
```

In [16]:
```
# Create distance matrix with all source-sink-relations

list_relations = []

for index_source, row_source in df_source.iterrows():
    for index_sink, row_sink in df_source.iterrows():
        list_relations.append([row_source[0], row_sink[0]])

df_dist = pd.DataFrame(list_relations, columns=["Source", "Sink"])
df_dist.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 256 entries, 0 to 255
Data columns (total 2 columns):
Source    256 non-null int64
Sink      256 non-null int64
dtypes: int64(2)
memory usage: 4.1 KB
```

In [17]:
```
df_dist.head()
```

Out[17]:

	Source	Sink
0	70173	70173
1	70173	70174
2	70173	70176
3	70173	70178
4	70173	70180

In [18]:
```
df_provide_geo = df_str_geo.loc[:, ["PLZ", "Latitude", "Longitude"]]
df_provide_geo.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16 entries, 0 to 15
Data columns (total 3 columns):
PLZ          16 non-null int64
Latitude     16 non-null float64
Longitude    16 non-null float64
dtypes: float64(2), int64(1)
memory usage: 512.0 bytes
```

In [19]:
```
# Map coordinates for source
df_dist1 = pd.merge(df_dist, df_provide_geo, how="left", left_on="Source", rig
ht_on="PLZ", copy=False)
df_dist1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 256 entries, 0 to 255
Data columns (total 5 columns):
Source       256 non-null int64
Sink         256 non-null int64
PLZ          256 non-null int64
Latitude     256 non-null float64
Longitude    256 non-null float64
dtypes: float64(2), int64(3)
memory usage: 12.0 KB
```

In [20]:
```
df_dist1.rename(columns={"Latitude": "Source_lat", "Longitude":"Source_lngt"},
inplace=True)
df_dist1.drop("PLZ", axis=1, inplace=True)
df_dist1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 256 entries, 0 to 255
Data columns (total 4 columns):
Source         256 non-null int64
Sink           256 non-null int64
Source_lat     256 non-null float64
Source_lngt    256 non-null float64
dtypes: float64(2), int64(2)
memory usage: 10.0 KB
```

In [21]:
```
# Map coordinates for source
df_dist2 = pd.merge(df_dist1, df_provide_geo, how="left", left_on="Sink", righ
t_on="PLZ", copy=False)
df_dist2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 256 entries, 0 to 255
Data columns (total 7 columns):
Source          256 non-null int64
Sink            256 non-null int64
Source_lat      256 non-null float64
Source_lngt     256 non-null float64
PLZ             256 non-null int64
Latitude        256 non-null float64
Longitude       256 non-null float64
dtypes: float64(4), int64(3)
memory usage: 16.0 KB
```

In [22]:
```
df_dist2.rename(columns={"Latitude": "Sink_lat", "Longitude":"Sink_lngt"}, inp
lace=True)
df_dist2.drop("PLZ", axis=1, inplace=True)
df_dist2.head()
```

Out[22]:

	Source	Sink	Source_lat	Source_lngt	Sink_lat	Sink_lngt
0	70173	70173	48.777845	9.178425	48.777845	9.178425
1	70173	70174	48.777845	9.178425	48.782793	9.169334
2	70173	70176	48.777845	9.178425	48.777425	9.161045
3	70173	70178	48.777845	9.178425	48.769172	9.167613
4	70173	70180	48.777845	9.178425	48.764008	9.174807

In [23]:
```
# Now we have a distance matrix we can use to pull the distances

list_distances = []

for index, row in df_dist2.iterrows():
    source_lat = row[2]
    source_lngt = row[3]
    sink_lat = row[4]
    sink_lngt = row[5]
    source = (source_lat, source_lngt)
    sink = (sink_lat, sink_lngt)

    list_distances.append(np.round(geodesic(source, sink).meters, 0))

len(list_distances)
```

Out[23]: 256

In [24]:
```
df_result_dist = pd.DataFrame(list_distances, columns=["Distance_m"])
df_result_dist.head()
```

Out[24]:

	Distance_m
0	0.0
1	866.0
2	1278.0
3	1250.0
4	1562.0

In [25]:
```
df_dist3 = df_dist2.join(df_result_dist, how="left")
df_dist3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 256 entries, 0 to 255
Data columns (total 7 columns):
Source         256 non-null int64
Sink           256 non-null int64
Source_lat     256 non-null float64
Source_lngt    256 non-null float64
Sink_lat       256 non-null float64
Sink_lngt      256 non-null float64
Distance_m     256 non-null float64
dtypes: float64(5), int64(2)
memory usage: 26.0 KB
```

In [26]:
```
# Drop zero distances
index_zero = df_dist3[df_dist3["Distance_m"] == 0].index
df_dist3.drop(index_zero, axis=0, inplace=True)
df_dist3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 240 entries, 1 to 254
Data columns (total 7 columns):
Source         240 non-null int64
Sink           240 non-null int64
Source_lat     240 non-null float64
Source_lngt    240 non-null float64
Sink_lat       240 non-null float64
Sink_lngt      240 non-null float64
Distance_m     240 non-null float64
dtypes: float64(5), int64(2)
memory usage: 15.0 KB
```

In [27]:
```
# Group by Source to get distance to closest adjacent ZIP-area (min distance)
df_dist_grouped = df_dist3.groupby(["Source"])["Distance_m"].min().reset_index
()
```

In [28]:
```
df_str_input = pd.merge(df_str_geo, df_dist_grouped, how="left", left_on="PLZ"
, right_on="Source")
df_str_input.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16 entries, 0 to 15
Data columns (total 14 columns):
PLZ                  16 non-null int64
Bad Cannstatt        16 non-null uint8
Botnang              16 non-null uint8
Feuerbach            16 non-null uint8
Frauenkopf           16 non-null uint8
Stuttgart-Mitte      16 non-null uint8
Stuttgart-Nord       16 non-null uint8
Stuttgart-Ost        16 non-null uint8
Stuttgart-Süd        16 non-null uint8
Stuttgart-West       16 non-null uint8
Latitude             16 non-null float64
Longitude            16 non-null float64
Source               16 non-null int64
Distance_m           16 non-null float64
dtypes: float64(3), int64(2), uint8(9)
memory usage: 912.0 bytes
```

In [29]:
```
df_str_input.drop("Source", axis=1)
```

Out[29]:

	PLZ	Bad Cannstatt	Botnang	Feuerbach	Frauenkopf	Stuttgart-Mitte	Stuttgart-Nord	Stuttgart-Ost	Stuttgart Sü
0	70173	0	0	0	0	1	0	0	
1	70174	0	0	0	0	1	1	0	
2	70176	0	0	0	0	1	0	0	
3	70178	0	0	0	0	1	0	0	
4	70180	0	0	0	0	1	0	0	
5	70182	0	0	0	0	1	0	0	
6	70184	0	0	0	1	1	0	1	
7	70186	0	0	0	0	0	0	1	
8	70188	0	0	0	0	1	0	1	
9	70190	0	0	0	0	1	0	1	
10	70191	1	0	0	0	0	1	0	
11	70192	0	0	1	0	0	1	0	
12	70193	0	0	0	0	0	1	0	
13	70195	0	1	0	0	0	0	0	
14	70197	0	0	0	0	0	0	0	
15	70199	0	0	0	0	0	0	0	

2. Foursquare information on restaurants in ZIP-areas

Foursquare credentials

In [30]:
```
credentials = pd.read_excel("../Credentials.xlsx", header=0)
credentials.columns
```

Out[30]:
```
Index(['Provider', 'Key', 'Value'], dtype='object')
```

In [31]:
```
provider = "Foursquare"
cred_fsquare = credentials[credentials["Provider"] == provider]

CLIENT_ID = cred_fsquare[cred_fsquare["Key"] == "CLIENT_ID"].values[0][2] # yo
ur Foursquare ID
CLIENT_SECRET = cred_fsquare[cred_fsquare["Key"] == "CLIENT_SECRET"].values[0]
[2] # your Foursquare Secret
ACCESS_TOKEN = cred_fsquare[cred_fsquare["Key"] == "ACCESS_TOKEN"].values[0][2
] # your FourSquare Access Token
VERSION = '20210101' # Foursquare API version
```

2.1 Get list of restaurants from foursquare

In [32]:
```
#Pull food-venues for each ZIP from Foursquare; using max. distance as radius

def exploreFoodVenues(zip_code, zip_lat, zip_lng, radius):
    # Explore Top100
    LIMIT = 100
    # Category ID for category "FOOD"
    CAT_ID = "4d4b7105d754a06374d81259"

    venues_list = []

    for code, lat, lng, r_m in zip(zip_code, zip_lat, zip_lng, radius):
        # create the API request URL
        ven_expl_url = 'https://api.foursquare.com/v2/venues/explore?&client_i
d={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}&categoryId={}'.format(
            CLIENT_ID, CLIENT_SECRET, VERSION, lat, lng, r_m, LIMIT, CAT_ID)

        # make the GET request
        ven_results = requests.get(ven_expl_url).json()["response"]['groups'][
0]['items']

        try:
            # try to get ZIP-code
            venues_list.append([(code, lat, lng, v['venue']['id'], v['venue'][
'name'], v['venue']['location']['lat'],
                                 v['venue']['location']['lng'], v['venue']['lo
cation']['formattedAddress'][1],
                                 v['venue']['categories'][0]['name']) for v in
ven_results])
        except:
            # use dummy ZIP
            venues_list.append([(code, lat, lng, v['venue']['id'], v['venue'][
'name'], v['venue']['location']['lat'],
                                 v['venue']['location']['lng'], "n/a",
                                 v['venue']['categories'][0]['name']) for v in
ven_results])

    nearby_venues = pd.DataFrame([item for venue_list in venues_list for item
in venue_list])
    nearby_venues.columns = ['PLZ', 'Latitude', 'Longitude', 'Venue_id',
                         'Venue_name', 'Venue_latitude', 'Venue_longitude', 'V
enue_PLZ', 'Venue_category']
    return(nearby_venues)
```

In [33]:
```
df_venues = exploreFoodVenues(df_str_input["PLZ"], df_str_input["Latitude"],
                              df_str_input["Longitude"], df_str_input["Distanc
e_m"])
df_venues.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 731 entries, 0 to 730
Data columns (total 9 columns):
PLZ                  731 non-null int64
Latitude             731 non-null float64
Longitude            731 non-null float64
Venue_id             731 non-null object
Venue_name           731 non-null object
Venue_latitude       731 non-null float64
Venue_longitude      731 non-null float64
Venue_PLZ            731 non-null object
Venue_category       731 non-null object
dtypes: float64(4), int64(1), object(4)
memory usage: 51.5+ KB
```

In [34]:
```
# List of unique Venue_IDs with no. of appearances in Explore-search
df_venues_unique = df_venues.groupby(df_venues.columns.to_list()[3:9])["PLZ"].
count().to_frame().reset_index()
df_venues_unique.rename(columns={"PLZ":"Appearances"}, inplace=True)
df_venues_unique.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 465 entries, 0 to 464
Data columns (total 7 columns):
Venue_id             465 non-null object
Venue_name           465 non-null object
Venue_latitude       465 non-null float64
Venue_longitude      465 non-null float64
Venue_PLZ            465 non-null object
Venue_category       465 non-null object
Appearances          465 non-null int64
dtypes: float64(2), int64(1), object(4)
memory usage: 25.5+ KB
```

2.2 Identify restaurant belonging to Top10-categories

In [35]:
```
df_all_cat = df_venues_unique.groupby(["Venue_category"])["Venue_id"].count().
to_frame().reset_index().sort_values(
    by="Venue_id", ascending=False)
df_all_cat.rename(columns={"Venue_id":"Appearances"}, inplace=True)

# Remove category "Restaurant" because its meaningless
index_restaurant = df_all_cat[df_all_cat["Venue_category"] == "Restaurant"].in
dex
df_all_cat.drop(index_restaurant, axis=0, inplace=True)
df_top10cat = df_all_cat.iloc[0:10, :]
df_top10cat
```

Out[35]:

	Venue_category	Appearances
9	Café	54
29	Italian Restaurant	51
25	German Restaurant	49
4	Bakery	46
7	Burger Joint	13
2	Asian Restaurant	11
60	Turkish Restaurant	11
53	Sushi Restaurant	11
19	Fast Food Restaurant	11
40	Pizza Place	10

In [36]:
```
list_top10 = df_top10cat["Venue_category"].to_list()
list_top10
```

Out[36]:
```
['Café',
 'Italian Restaurant',
 'German Restaurant',
 'Bakery',
 'Burger Joint',
 'Asian Restaurant',
 'Turkish Restaurant',
 'Sushi Restaurant',
 'Fast Food Restaurant',
 'Pizza Place']
```

In [37]:
```
# Venues belonging to top10 categories
bool_series= df_venues_unique["Venue_category"].isin(list_top10)
df_venues_top = df_venues_unique[bool_series]
df_venues_top.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 267 entries, 0 to 464
Data columns (total 7 columns):
Venue_id           267 non-null object
Venue_name         267 non-null object
Venue_latitude     267 non-null float64
Venue_longitude    267 non-null float64
Venue_PLZ          267 non-null object
Venue_category     267 non-null object
Appearances        267 non-null int64
dtypes: float64(2), int64(1), object(4)
memory usage: 16.7+ KB
```

2.3 Get ratings for selected venues from Foursquare

In [38]:
```
#Pull venue-ratings
"""
def pullVenueInfos(list_venues):

    info_list = []

    for venue_id in list_venues:
        # create the API request URL
        ven_info_url = 'https://api.foursquare.com/v2/venues/{}?&client_id={}&
client_secret={}&v={}'.format(
            venue_id, CLIENT_ID, CLIENT_SECRET, VERSION)

        # make the GET request
        ven_info = requests.get(ven_info_url).json()['response']['venue']

        try:
            # try to get rating
            info_list.append([venue_id, ven_info['rating']])
        except:
            # use dummy rating
            info_list.append([venue_id, "n/a"])

    return_ven_info = pd.DataFrame(info_list, columns=["Venue_id", "Rating"])

    return(return_ven_info)
"""
```

In [59]:
```
df_venue_ratings = pd.read_excel("EXPORT-venue-ratings.xlsx", header=0)
df_venue_ratings.drop("Unnamed: 0", axis=1, inplace=True)
df_venue_ratings.head()
```

Out[59]:

	Venue_id	Rating
0	4b15579bf964a52041ab23e3	NaN
1	4b1ce1eff964a520410a24e3	8.6
2	4b44c3f8f964a52088fb25e3	6.9
3	4b4823d9f964a520f64826e3	6.6
4	4b48a6a8f964a520a35126e3	7.0

In [60]:
```
df_venue_ratings.shape
```

Out[60]:
```
(267, 2)
```

In [61]:
```
# Make a backup of ratings
#df_venue_ratings.to_excel("EXPORT-venue-ratings.xlsx")
```

In [62]:
```
# Merge ratings with other information
df_venues_con = pd.merge(df_venues_top, df_venue_ratings, how="left", on="Venu
e_id")
df_venues_con.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 305 entries, 0 to 304
Data columns (total 8 columns):
Venue_id            305 non-null object
Venue_name          305 non-null object
Venue_latitude      305 non-null float64
Venue_longitude     305 non-null float64
Venue_PLZ           305 non-null object
Venue_category      305 non-null object
Appearances         305 non-null int64
Rating              212 non-null float64
dtypes: float64(3), int64(1), object(4)
memory usage: 21.4+ KB
```

In [111]:
```
# Drop venues with rating = "n/a"
index_na = df_venues_con[df_venues_con["Rating"] == "n/a"].index
df_venues_con.drop(index_na, axis=0, inplace=True)
df_venues_con.dropna(how="any", axis=0, inplace=True)
df_venues_con.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 212 entries, 1 to 300
Data columns (total 8 columns):
Venue_id           212 non-null object
Venue_name         212 non-null object
Venue_latitude     212 non-null float64
Venue_longitude    212 non-null float64
Venue_PLZ          212 non-null object
Venue_category     212 non-null object
Appearances        212 non-null int64
Rating             212 non-null float64
dtypes: float64(3), int64(1), object(4)
memory usage: 14.9+ KB
```

2.4 Final preparation of input data

Taking venue information and information on adjacent hoods

In [112]:
```
df_venues_cut = df_venues.loc[:, ["PLZ", "Venue_id"]]
```

In [113]:
```
df_str_venues = pd.merge(df_venues_con, df_venues_cut, how="left", on="Venue_i
d")
df_str_venues.rename(columns={"PLZ":"Adj_zip_codes"}, inplace=True)
df_str_venues.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 409 entries, 0 to 408
Data columns (total 9 columns):
Venue_id           409 non-null object
Venue_name         409 non-null object
Venue_latitude     409 non-null float64
Venue_longitude    409 non-null float64
Venue_PLZ          409 non-null object
Venue_category     409 non-null object
Appearances        409 non-null int64
Rating             409 non-null float64
Adj_zip_codes      409 non-null int64
dtypes: float64(3), int64(2), object(4)
memory usage: 32.0+ KB
```

In [114]:
```
df_venues_per_zip = df_str_venues.groupby(["Adj_zip_codes", "Venue_category"])
["Venue_id"].count().reset_index()
df_venues_per_zip.sort_values(by="Adj_zip_codes", inplace=True)
df_venues_per_zip.head()
```

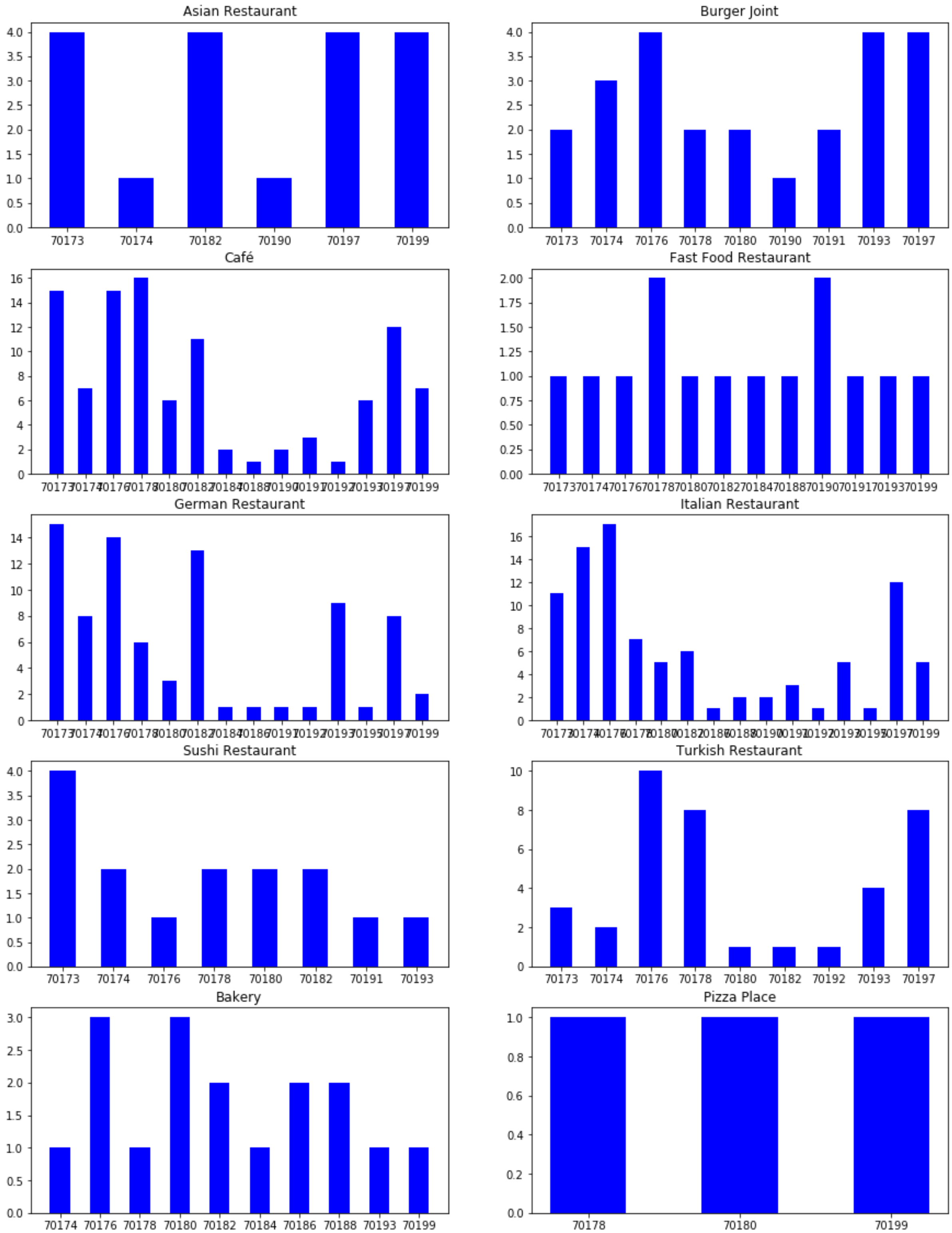
Out[114]:

	Adj_zip_codes	Venue_category	Venue_id
0	70173	Asian Restaurant	4
1	70173	Burger Joint	2
2	70173	Café	15
3	70173	Fast Food Restaurant	1
4	70173	German Restaurant	15

In [115]:
```
import numpy as np
import matplotlib.pyplot as plt

# List of all categories
categories = df_venues_per_zip["Venue_category"].unique()

# Category 0
cat0 = categories[0]
df_cat0 = df_venues_per_zip[df_venues_per_zip["Venue_category"] == categories[
0]]
x_bins_cat0 = df_cat0["Adj_zip_codes"].astype(str).tolist()
series_cat0 = df_cat0["Venue_id"].tolist()

# Category 1
cat1 = categories[1]
df_cat1 = df_venues_per_zip[df_venues_per_zip["Venue_category"] == categories[
1]]
x_bins_cat1 = df_cat1["Adj_zip_codes"].astype(str).tolist()
series_cat1 = df_cat1["Venue_id"].tolist()

# Category 2
cat2 = categories[2]
df_cat2 = df_venues_per_zip[df_venues_per_zip["Venue_category"] == categories[
2]]
x_bins_cat2 = df_cat2["Adj_zip_codes"].astype(str).tolist()
series_cat2 = df_cat2["Venue_id"].tolist()

# Category 3
cat3 = categories[3]
df_cat3 = df_venues_per_zip[df_venues_per_zip["Venue_category"] == categories[
3]]
x_bins_cat3 = df_cat3["Adj_zip_codes"].astype(str).tolist()
series_cat3 = df_cat3["Venue_id"].tolist()

# Category 4
cat4 = categories[4]
df_cat4 = df_venues_per_zip[df_venues_per_zip["Venue_category"] == categories[
4]]
x_bins_cat4 = df_cat4["Adj_zip_codes"].astype(str).tolist()
series_cat4 = df_cat4["Venue_id"].tolist()

# Category 5
cat5 = categories[5]
df_cat5 = df_venues_per_zip[df_venues_per_zip["Venue_category"] == categories[
5]]
x_bins_cat5 = df_cat5["Adj_zip_codes"].astype(str).tolist()
series_cat5 = df_cat5["Venue_id"].tolist()

# Category 6
cat6 = categories[6]
df_cat6 = df_venues_per_zip[df_venues_per_zip["Venue_category"] == categories[
6]]
x_bins_cat6 = df_cat6["Adj_zip_codes"].astype(str).tolist()
series_cat6 = df_cat6["Venue_id"].tolist()

# Category 7
```

```
cat7 = categories[7]
df_cat7 = df_venues_per_zip[df_venues_per_zip["Venue_category"] == categories[
7]]
x_bins_cat7 = df_cat7["Adj_zip_codes"].astype(str).tolist()
series_cat7 = df_cat7["Venue_id"].tolist()

# Category 8
cat8 = categories[8]
df_cat8 = df_venues_per_zip[df_venues_per_zip["Venue_category"] == categories[
8]]
x_bins_cat8 = df_cat8["Adj_zip_codes"].astype(str).tolist()
series_cat8 = df_cat8["Venue_id"].tolist()

# Category 9
cat9 = categories[9]
df_cat9 = df_venues_per_zip[df_venues_per_zip["Venue_category"] == categories[
9]]
x_bins_cat9 = df_cat9["Adj_zip_codes"].astype(str).tolist()
series_cat9 = df_cat9["Venue_id"].tolist()

fig, ax = plt.subplots(5, 2, figsize=(15,20))
fig.suptitle("Number of restaurant for each category per ZIP-code", fontsize=1
4)

ax[0][0].bar(x_bins_cat0, series_cat0, color='b', width=0.5)
ax[0][0].set_title(str(cat0))

ax[0][1].bar(x_bins_cat1, series_cat1, color='b', width=0.5)
ax[0][1].set_title(str(cat1))

ax[1][0].bar(x_bins_cat2, series_cat2, color='b', width=0.5)
ax[1][0].set_title(str(cat2))

ax[1][1].bar(x_bins_cat3, series_cat3, color='b', width=0.5)
ax[1][1].set_title(str(cat3))

ax[2][0].bar(x_bins_cat4, series_cat4, color='b', width=0.5)
ax[2][0].set_title(str(cat4))

ax[2][1].bar(x_bins_cat5, series_cat5, color='b', width=0.5)
ax[2][1].set_title(str(cat5))

ax[3][0].bar(x_bins_cat6, series_cat6, color='b', width=0.5)
ax[3][0].set_title(str(cat6))

ax[3][1].bar(x_bins_cat7, series_cat7, color='b', width=0.5)
ax[3][1].set_title(str(cat7))

ax[4][0].bar(x_bins_cat8, series_cat8, color='b', width=0.5)
ax[4][0].set_title(str(cat8))

ax[4][1].bar(x_bins_cat9, series_cat9, color='b', width=0.5)
ax[4][1].set_title(str(cat9))

plt.show()
```

Number of restaurant for each category per ZIP-code



In [116]:
```
# One hot encoding of adj_zip_codes
onehot_zip_codes = pd.get_dummies(df_str_venues["Adj_zip_codes"])
onehot_zip_codes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 409 entries, 0 to 408
Data columns (total 16 columns):
70173    409 non-null uint8
70174    409 non-null uint8
70176    409 non-null uint8
70178    409 non-null uint8
70180    409 non-null uint8
70182    409 non-null uint8
70184    409 non-null uint8
70186    409 non-null uint8
70188    409 non-null uint8
70190    409 non-null uint8
70191    409 non-null uint8
70192    409 non-null uint8
70193    409 non-null uint8
70195    409 non-null uint8
70197    409 non-null uint8
70199    409 non-null uint8
dtypes: uint8(16)
memory usage: 9.6 KB
```

In [117]:
```
onehot_zip_codes.shape
```

Out[117]:
```
(409, 16)
```

In [118]:
```
df_str_venues_oh = df_str_venues.join(onehot_zip_codes, how="left")
df_str_venues_oh.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 409 entries, 0 to 408
Data columns (total 25 columns):
Venue_id             409 non-null object
Venue_name           409 non-null object
Venue_latitude       409 non-null float64
Venue_longitude      409 non-null float64
Venue_PLZ            409 non-null object
Venue_category       409 non-null object
Appearances          409 non-null int64
Rating               409 non-null float64
Adj_zip_codes        409 non-null int64
70173                409 non-null uint8
70174                409 non-null uint8
70176                409 non-null uint8
70178                409 non-null uint8
70180                409 non-null uint8
70182                409 non-null uint8
70184                409 non-null uint8
70186                409 non-null uint8
70188                409 non-null uint8
70190                409 non-null uint8
70191                409 non-null uint8
70192                409 non-null uint8
70193                409 non-null uint8
70195                409 non-null uint8
70197                409 non-null uint8
70199                409 non-null uint8
dtypes: float64(3), int64(2), object(4), uint8(16)
memory usage: 58.3+ KB
```

In [119]:
```
df_str_venues_oh["Rating"] = pd.to_numeric(df_str_venues_oh["Rating"])
df_str_venues_oh.describe()
```

Out[119]:

	Venue_latitude	Venue_longitude	Appearances	Rating	Adj_zip_codes	70173
count	409.000000	409.000000	409.000000	409.000000	409.000000	409.000000
mean	48.775125	9.168861	1.728606	7.216870	70182.674817	0.134474
std	0.006479	0.013272	0.562024	0.696338	8.999694	0.341579
min	48.760014	9.136989	1.000000	4.800000	70173.000000	0.000000
25%	48.771246	9.156801	1.000000	6.800000	70176.000000	0.000000
50%	48.774375	9.168917	2.000000	7.200000	70178.000000	0.000000
75%	48.777783	9.177596	2.000000	7.700000	70192.000000	0.000000
max	48.805550	9.214821	3.000000	8.900000	70199.000000	1.000000

8 rows × 21 columns

In [120]:
```
# Drop unnecessary columns
df_str_reduced = df_str_venues_oh.drop(["Venue_latitude", "Venue_longitude",
"Venue_PLZ", "Appearances", "Adj_zip_codes"], axis=1)
df_str_reduced.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 409 entries, 0 to 408
Data columns (total 20 columns):
Venue_id          409 non-null object
Venue_name        409 non-null object
Venue_category    409 non-null object
Rating            409 non-null float64
70173             409 non-null uint8
70174             409 non-null uint8
70176             409 non-null uint8
70178             409 non-null uint8
70180             409 non-null uint8
70182             409 non-null uint8
70184             409 non-null uint8
70186             409 non-null uint8
70188             409 non-null uint8
70190             409 non-null uint8
70191             409 non-null uint8
70192             409 non-null uint8
70193             409 non-null uint8
70195             409 non-null uint8
70197             409 non-null uint8
70199             409 non-null uint8
dtypes: float64(1), object(3), uint8(16)
memory usage: 42.4+ KB
```

In [121]:
```
zip_cols = df_str_reduced.columns.to_list()[4:20]
```

In [122]:
```
# Group by Venues
df_str_group = df_str_reduced.groupby(["Venue_id", "Venue_name", "Venue_catego
ry", "Rating"])[zip_cols].max().reset_index()
```

In [123]:
```
# Transform "Venue_category" into numeric
from sklearn import preprocessing
le_category = preprocessing.LabelEncoder()
le_category.fit(df_str_group["Venue_category"])
```

Out[123]:
```
LabelEncoder()
```

In [124]:
```
df_str_group["Num_category"] = le_category.transform(df_str_group["Venue_categ
ory"])
```

In [125]:
```
df_str_group.head()
```

Out[125]:

	Venue_id	Venue_name	Venue_category	Rating	70173	70174	70176	7017
0	4b1ce1eff964a520410a24e3	Oggi Tavola Mediterranea	Italian Restaurant	8.6	1	1	0	(
1	4b44c3f8f964a52088fb25e3	Vapiano	Italian Restaurant	6.9	1	1	0	(
2	4b4823d9f964a520f64826e3	Bierhaus West	German Restaurant	6.6	0	1	1	(
3	4b48a6a8f964a520a35126e3	Alte Kanzlei	German Restaurant	7.0	1	0	0	(
4	4b4c5dfef964a520aab126e3	Stern Kebap	Fast Food Restaurant	6.1	0	0	0	(

5 rows × 21 columns

In [126]:
```
df_str_group.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 176 entries, 0 to 175
Data columns (total 21 columns):
Venue_id          176 non-null object
Venue_name        176 non-null object
Venue_category    176 non-null object
Rating            176 non-null float64
70173             176 non-null uint8
70174             176 non-null uint8
70176             176 non-null uint8
70178             176 non-null uint8
70180             176 non-null uint8
70182             176 non-null uint8
70184             176 non-null uint8
70186             176 non-null uint8
70188             176 non-null uint8
70190             176 non-null uint8
70191             176 non-null uint8
70192             176 non-null uint8
70193             176 non-null uint8
70195             176 non-null uint8
70197             176 non-null uint8
70199             176 non-null uint8
Num_category      176 non-null int32
dtypes: float64(1), int32(1), object(3), uint8(16)
memory usage: 9.0+ KB
```

In [127]:
```
# Summary of Categories
orig_categories = le_category.classes_
new_labels = le_category.transform(orig_categories)
category_legend = pd.DataFrame({'Original': orig_categories, 'New': new_labels
})
category_legend
```

Out[127]:

	Original	New
0	Asian Restaurant	0
1	Bakery	1
2	Burger Joint	2
3	Café	3
4	Fast Food Restaurant	4
5	German Restaurant	5
6	Italian Restaurant	6
7	Pizza Place	7
8	Sushi Restaurant	8
9	Turkish Restaurant	9

3. ML to predict rating based on location and restaurant category

3.1 Prepare input data for ML-model

In [128]:
```
#Prepare features
x_data = np.asanyarray(df_str_group.iloc[:, 4:22])
x_cols = df_str_group.columns.to_list()[4:22]
```

In [129]:
```
# Prepare target labels
y_data = np.asanyarray(df_str_group.loc[:, ["Rating"]])
```

In [130]:
```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, test_size=
0.2, random_state=1)
```

In [131]:
```
y_test.shape
```

Out[131]: (36, 1)

3.2 Linear regression

In [132]:
```
from sklearn.linear_model import LinearRegression
```

In [133]:
```
LR_model = LinearRegression(fit_intercept = True)
```

In [135]:
```
LR_model.fit(x_train, y_train)
```

Out[135]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=Fals
e)
```

In [136]:
```
LR_model.coef_
```

Out[136]:
```
array([[ 6.28636035e-01, -3.95551476e-01,  5.84663288e-04,
         2.41899258e-01,  1.95913003e-01, -1.27639758e-01,
         7.98057348e-01, -4.36527794e-01,  5.30953414e-01,
         2.18148910e-02, -1.32459018e-01, -1.33674033e-02,
         7.64931346e-01, -1.86537638e+00, -4.45801548e-01,
         2.89966163e-01,  5.59666480e-02]])
```

In [137]:
```
LR_model.intercept_
```

Out[137]:
```
array([6.76464514])
```

In [138]:
```
LR_model.score(x_train, y_train)
```

Out[138]:
```
0.2457069741693402
```

--> Very poor R-squared value. Means the model doesn't really fit

In [139]:
```
y_predict = LR_model.predict(x_test)
```

In [140]:
```
from sklearn.metrics import mean_squared_error
```

In [141]:
```
mse_lr = mean_squared_error(y_test, y_predict)
mse_lr
```

Out[141]:
```
0.6593930004303137
```

--> but the MSE is quite small with 0.66

Let's have a look on the coefficients

In [142]:
```
df_coeff_lr = pd.DataFrame({"Feature": x_cols, "Coefficients": np.round(LR_model.coef_[0], 3)}).set_index("Feature")
df_coeff_lr.sort_values(by="Coefficients", ascending=False, inplace=True)
df_coeff_lr.head(30)
```

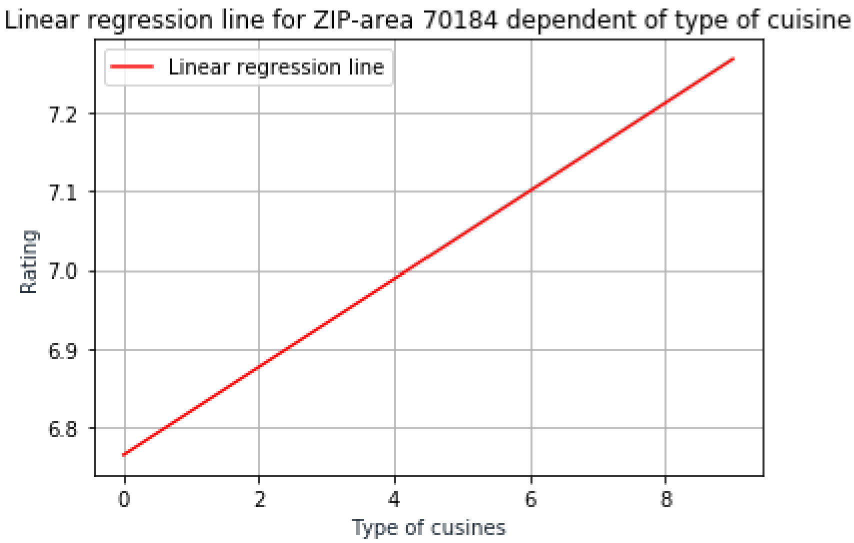
Out[142]:

	Coefficients
Feature	
70184	0.798
70193	0.765
70173	0.629
70188	0.531
70199	0.290
70178	0.242
70180	0.196
Num_category	0.056
70190	0.022
70176	0.001
70192	-0.013
70182	-0.128
70191	-0.132
70174	-0.396
70186	-0.437
70197	-0.446
70195	-1.865

3.3 Conclusion of Linear regression model

In [143]:
```
x_values = np.linspace(0, 9, num=10)
y_lin = df_coeff_lr.loc["Num_category", "Coefficients"] * x_values + LR_model.intercept_
```

In [144]:
```
plt.plot(x_values, y_lin, '-r', label='Linear regression line')
plt.title('Linear regression line for ZIP-area ' + str(df_coeff_lr.index[0]) +
" dependent of type of cuisine")
plt.xlabel('Type of cusines', color='#1C2833')
plt.ylabel('Rating', color='#1C2833')
plt.legend(loc='upper left')
plt.grid()
plt.show()
```



Linear regression line shows the likely range of a restaurant rating in ZIP-area 70184.
ZIP-area is the area, where the restaurants have the best rating

> min. rating is 6.75
> max. rating is 7.3
> type of cusine has only limited impact as the MSE is 0.66

Conclusion:
The impact of location and cusine on the rating is not significant enough to be considered as crucial criteria

In [ ]: