The Battle of Neighborhoods- Milan neighborhoods price houses

Gabriele Cecere

1st January 2021

Introduction

Creative Capital of fashion and luxury, Milan is a cutting-edge city, very active in economic and cultural profile, a reality to be discovered to find its hidden treasures of poetic views, Art Nouveau buildings and small architectural gems ranging from Gothic to Romanesque style, scattered throughout its beautiful and ancient historic center, a reason that drives millions of visitors from all over the world every day to visit the city of charm and glamor.

Milan is also listed within the top 10 cities for investments in Western Europe and mentioned as one of the preferred destinations for investing in the real estate sector.

Business Problem

The aim of the project is to investigate about what are some of the variables that contribute to drive up the house's prices in some neighborhoods respect other neighborhoods. In particular the study try to analize if there is some relationship between the presence of specific venues in a neighborhood and the price's houses in that neighborhood.

Data Description

To perform all the analysis and give an answer to our problem, we need to rely on some data.

- First of all, we need to obtain the house prices' for each area of Milan. For this aim, we need to web scrape data from **immobiliare.it**, an italian platform for publishing and searching for real estate ads.

  Zone: Neighborhoods in each area

  Vendita: Average sale price per square meters in each area

  Affitto: Average rent price per square meters in each area (we aren't use this variable)

- Next, to obtain the geo location of each neighborhood we use **ArcGIS API**. Part of the Esri Geospatial Cloud, ArcGIS enables to connect people, locations, and data using interactive maps. Through this API we obtain 2 new columns:

  Latitude: Latitude for each neighborhood

  Longitude: Longitude for each neighborhood

- Finally, we use **Foursquare API** to obtain data about different venues in each neighborhood. Foursquare is the most trusted, independent location data platform for understanding how people move through the real world. Using this API we obtain 5 addictional columns:

  Neighborhood: Name of the neighborhood

Neighborhood Latitude: Latitude of the neighborhood

Neighborhood Longitude: Longitude of the neighborhood

Venue: Name of the venue

Venue Category: Category of the venue

Methodology

To develop the project and solve our business problem, we will use python. The first step consists in importing the requiered libraries:

```
# Import python libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import matplotlib.colors as colors
import seaborn as sns
import requests
from bs4 import BeautifulSoup
from arcgis.geocoding import geocode
from arcgis.gis import GIS
gis = GIS()
import folium
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import RidgeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score,recall_score
from sklearn.metrics import f1_score
```

- *pandas:* Used to collect and to manipulate data in HTML and then for data analysis
- *numpy:* Used to do math operations with DataFrame columns
- *matplotlib and Seaborn:* Used to generate differnt plots
- *requests:* Used to handle http requests
- *BeautifulSoup:* Used to web scraping data
- *arcgis:* Used to obtain the geo location of each neighborhood
- *folium:* Used to generate maps for Milan
- *sklearn:* Used to develop and test the model

Data Collection

a) House price data

First of all we proceed to web scraping the house prices' for each area of Milan from the immobiliare.it webpage. To do that, we use the following code:

```
url = 'https://www.immobiliare.it/mercato-immobiliare/lombardia/milano/'
r = requests.get(url)
print(r.status_code)
```

```
soup = BeautifulSoup(r.text,'html.parser')
neighborhood_table = soup.find('table')
neighborhoods_prices = []
for data in neighborhood_table.find_all('tbody'):
    rows = data.find_all('tr')
    for row in rows:
        neighborhood = row.find('a',class_='nd-table__url').text
        avg_sale_price = row.find_all('td')[1].text
        data = {'neighborhood':neighborhood,
                'avg_sale_price':avg_sale_price}
        neighborhoods_prices.append(data)
```

With the purpose of better visualization and better manipulaton of the data, we proceed to insert it inside a DataFrame

```
df=pd.DataFrame(neighborhoods_prices)
df.head()
```

	neighborhood	avg_sale_price
0	Centro	9.366
1	Arco della Pace, Arena, Pagano	7.764
2	Genova, Ticinese	7.117
3	Quadronno, Palestro, Guastalla	7.898
4	Garibaldi, Moscova, Porta Nuova	8.581

The dataframe gather the average sale price per squared meter of houses in different zones of Milan. For a better Analysis and to correctly positioned the neighborhoods on the map, we renamed and split them.

```
neighborhoods=['Piazza Duomo','Pagano Metro','Ticinese,San Vittore','Guastalla','Garibaldi Repubblica,Brera',
 'De Angeli - Monte Rosa,Tre Torri,Portello','Navigli','Porta Romana,XXII Marzo','Buenos Aires - Venezia','Stazione Centrale',
'Lancetti,Giovanni Battista Bertini,Isola', 'Mac Mahon,Villapizzone,Maggiore - Musocco,Metro Rho Fieramilano','Bande Nere,Lorenteggio',
'Barona,Ronchetto sul Naviglio,San Cristoforo,Giambellino','Gratosoglio - Ticinello,Stadera,Tibaldi','Brenta,Calvairate',
'Adriano,Padova','Niguarda,Bicocca,Greco,Bruzzano','Tortona,Washington','Quarto Oggiaro,Bovisa,Dergano,Affori,Bovisasca,Comasina',
'Forze Armate,Selinunte,San Siro,Trenno,Gallaratese,QT 8', 'Quartiere degli Olmi,Baggio,Quarto Cagnino,Quinto Romano',
'Vigentina,Ex OM - Morivione,Ripamonti', 'Mecenate,Parco Monlué - Ponte Lambro', 'Città Studi,Corsica', 'Maciachini - Maggiolina',
'Viale Monza', 'Parco Lambro - Cimiano,Lambrate', 'Piazza Loreto', 'Triulzo Superiore,Rogoredo','Lodi - Corvetto']
```

```
df1 = pd.DataFrame(neighborhoods,columns=['neighborhood'])
df1['avg_sale_price'] = df.avg_sale_price
```

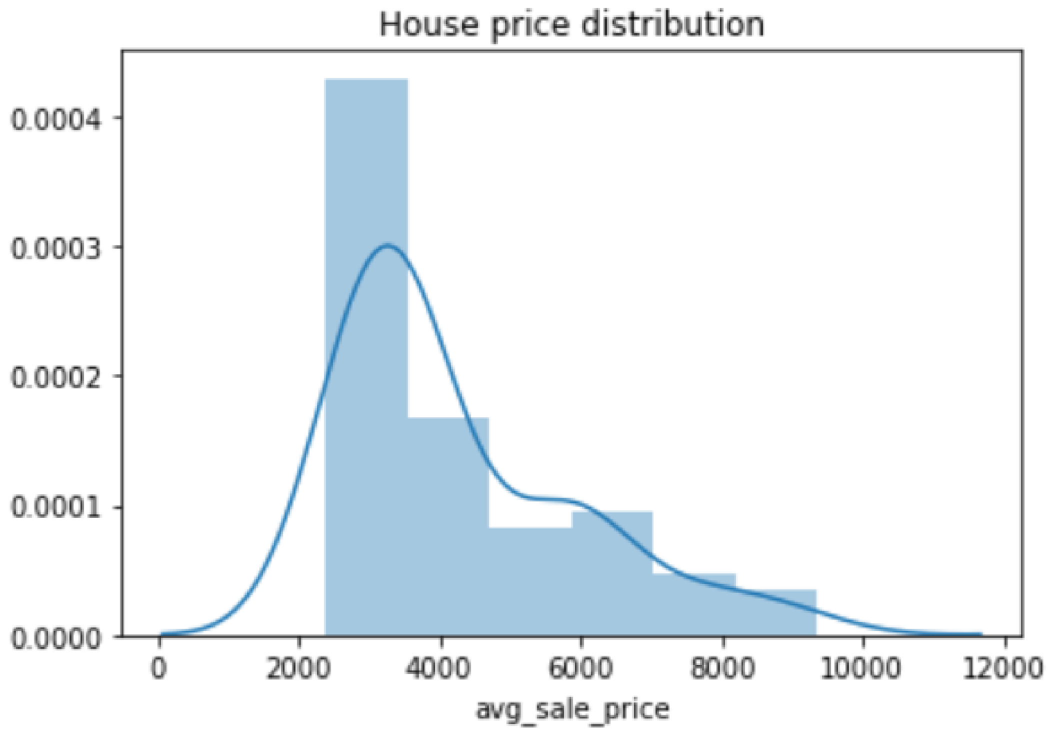
```
df1 = (df1.assign(neighborhood = df1['neighborhood'].str.split(','))
         .explode('neighborhood')
         .reset_index(drop=True))
df1['avg_sale_price'] = round(df1['avg_sale_price'].astype(str).astype(float)*1000)
df1.head()
```

	neighborhood	avg_sale_price
0	Piazza Duomo	9366.0
1	Pagano Metro	7764.0
2	Ticinese	7117.0
3	San Vittore	7117.0
4	Guastalla	7898.0

Now results interesting to observe the distribution of the hose price. We use the seaborn library as follow:

```
sns.distplot(df1["avg_sale_price"],kde=True).set_title('House price distribution');
```

And we obtain the following plot:



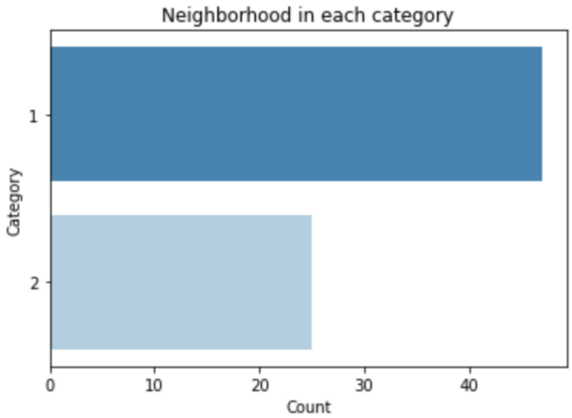
From the graph we can see that the price distribution is right skewed with the mayority of price concentrated beween approximatly 2300 and 3500, as we can see from the following table with the data description:

```
df1.describe()
```

	avg_sale_price
count	72.000000
mean	4243.680556
std	1704.649952
min	2372.000000
25%	3068.000000
50%	3542.500000
75%	5534.000000
max	9366.000000

Now we proceed to cluster the data in two categories of price: **1** = *Low Price (Price <= 4.000)* and **2** = *Medium/High Price (Price > 4.000)*

```
sns.barplot(df1.price_category.value_counts(),[1,2],orient='h',palette='Blues_r').set_title('Neighborhood in each category')
plt.xlabel('Count')
plt.ylabel('Category');
```



Now using the ArcGis API we obtain the latitude and longitude cordinates for each neighborhood and insert it into the df1 dataframe

```python
def get_x_y_milan(address1):
    lat_coords = 0
    lng_coords = 0
    g = geocode(address='{}, Milan,Italy,IT'.format(address1))[0]
    lng_coords = g['location']['x']
    lat_coords = g['location']['y']
    return lat_coords ,lng_coords
```

```python
lat=[]
lng=[]
for i in range(0,df1.shape[0]):
    latit = get_x_y_milan(df1.neighborhood[i])[0]
    longit = get_x_y_milan(df1.neighborhood[i])[1]
    lat.append(latit)
    lng.append(longit)
```

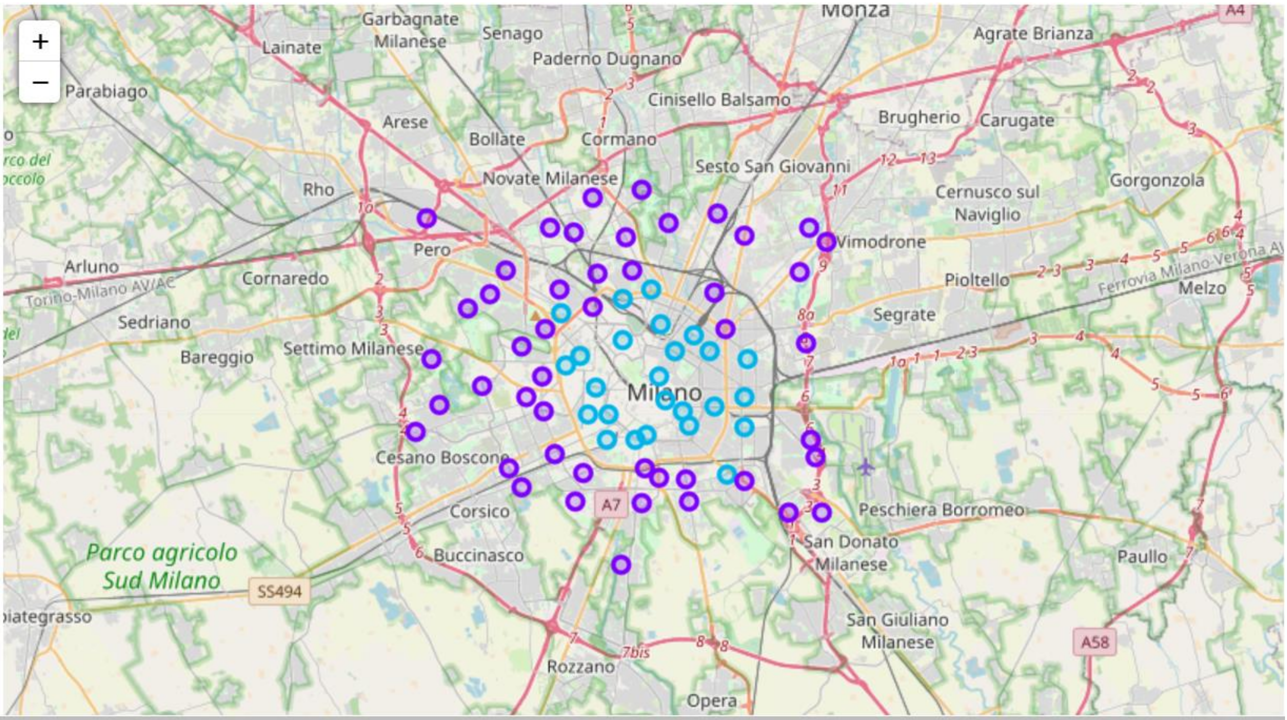
```python
df1['latitude'] = lat
df1['longitude'] = lng
```

```python
df1.head()
```

	neighborhood	avg_sale_price	price_category	latitude	longitude
0	Piazza Duomo	9366.0	2.0	45.46468	9.19049
1	Pagano Metro	7764.0	2.0	45.46839	9.16043
2	Ticinese	7117.0	2.0	45.45394	9.18224
3	San Vittore	7117.0	2.0	45.46013	9.16633
4	Guastalla	7898.0	2.0	45.46113	9.19850

As we now have the neighborhood coordinates, we can proceed to visualize the Milan map using the folio library and we plot also the price category clusters

```python
# First of all we need the Milan coordinates
milan_lat, milan_long = get_x_y_milan('Milan')

map_clusters = folium.Map(location=[milan_lat, milan_long], zoom_start=11)

kclusters = 3

# set color scheme for the clusters
x = np.arange(kclusters)
ys = [i + x + (i*x)**2 for i in range(kclusters)]
colors_array = cm.rainbow(np.linspace(0, 0.5, len(ys)))
rainbow = [colors.rgb2hex(i) for i in colors_array]

# add markers to the map
markers_colors = []
for lat, lon, poi, cluster in zip(df1['latitude'], df1['longitude'], df1['neighborhood'], df1['price_category']):
    label = folium.Popup('Price Category ' + str(int(cluster)) + '\n' + str(poi) , parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=rainbow[int(cluster-1)],
        fill=True,
        fill_color=rainbow[int(cluster-1)]
        ).add_to(map_clusters)

map_clusters
```



b) Venues Data

Next step is to obtain the list of venues in each neighborhood. We use the Foursquare API: First we create variables with the API credentials and after that, we define a function to obtain for each neighborhood all the venues in a range of 500 meters.

```
# Foursquare API credentials
CLIENT_ID = 'FVJVMGB5ZRPXVF5MQQG3K0X0HJIIDUIK4OY4PBUMSRP5QCIE' # your Foursquare ID
CLIENT_SECRET = 'ZNJRC0B2JKYYSYLTLO4CPPOHQXO3SPSOJOKGDHUDTUMWSQLK' # your Foursquare Secret
VERSION = '20180605' # Foursquare API version
LIMIT = 100 # A default Foursquare API limit value

print('Your credentails:')
print('CLIENT_ID: ' + CLIENT_ID)
print('CLIENT_SECRET:' + CLIENT_SECRET)
```

```
Your credentails:
CLIENT_ID: FVJVMGB5ZRPXVF5MQQG3K0X0HJIIDUIK4OY4PBUMSRP5QCIE
CLIENT_SECRET:ZNJRC0B2JKYYSYLTLO4CPPOHQXO3SPSOJOKGDHUDTUMWSQLK
```

```
def getNearbyVenues(names, latitudes, longitudes, radius=500):
    venues_list=[]
    for name, lat, lng in zip(names, latitudes, longitudes):
        print(name)

        # create the API request URL
        url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}'.format(
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            lat,
            lng,
            radius
            )
        # make the GET request
        results = requests.get(url).json()["response"]['groups'][0]['items']

        # return only relevant information for each nearby venue
        venues_list.append([(
            name,
            lat,
            lng,
            v['venue']['name'],
            v['venue']['categories'][0]['name']) for v in results])

    nearby_venues = pd.DataFrame([item for venue_list in venues_list for item in venue_list])
    nearby_venues.columns = ['neighborhood',
                  'neighborhood latitude',
                  'neighborhood longitude',
                  'venue',
                  'venue category']

    return(nearby_venues)
```

```python
milan_venues = getNearbyVenues(df1['neighborhood'], df1['latitude'], df1['longitude'])
```

We obtain the following dataframe:

```python
print(milan_venues.shape)
milan_venues.head()
```

```
(1430, 5)
```

	neighborhood	neighborhood latitude	neighborhood longitude	venue	venue category
0	Piazza Duomo	45.46468	9.19049	Piazza del Duomo	Plaza
1	Piazza Duomo	45.46468	9.19049	Galleria Vittorio Emanuele II	Monument / Landmark
2	Piazza Duomo	45.46468	9.19049	Terrazze del Duomo	Scenic Lookout
3	Piazza Duomo	45.46468	9.19049	Room Mate Giulia Hotel	Hotel
4	Piazza Duomo	45.46468	9.19049	Luini	Bakery

One Hot Encoding

Because the venues are categorial datatype, to proceed to model it and use machine learning to solve our problem, we need to convert it into numerical datatype. The one hot encoding represent the best choice in this case.

```python
milan_onehot = pd.get_dummies(milan_venues[['venue category']], prefix="", prefix_sep="")
milan_onehot
```

	Abruzzo Restaurant	Accessories Store	Adult Education Center	American Restaurant	Arcade	Argentinian Restaurant	Art Gallery	Art Museum	Asian Restaurant	Athletics & Sports	...	Tram Station	Trattor
0	0	0	0	0	0	0	0	0	0	0	...	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	
2	0	0	0	0	0	0	0	0	0	0	...	0	
3	0	0	0	0	0	0	0	0	0	0	...	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
1425	0	0	0	0	0	0	0	0	0	0	...	0	
1426	0	0	0	0	0	0	0	0	0	0	...	0	
1427	0	0	0	0	0	0	0	0	0	0	...	0	
1428	0	0	0	0	0	0	0	0	0	0	...	0	
1429	0	0	0	0	0	0	0	0	0	0	...	0	

1430 rows × 226 columns

Next we add back the neighborhood column as a first column:

```
# add neighborhood column back to dataframe
milan_onehot['neighborhood'] = milan_venues['neighborhood']
# move neighborhood column to the first column
fixed_columns = [milan_onehot.columns[-1]] + list(milan_onehot.columns[:-1])
milan_onehot = milan_onehot[fixed_columns]

milan_onehot.head()
```

	neighborhood	Abruzzo Restaurant	Accessories Store	Adult Education Center	American Restaurant	Arcade	Argentinian Restaurant	Art Gallery	Art Museum	Asian Restaurant	...	Tram Station	Tratt
0	Piazza Duomo	0	0	0	0	0	0	0	0	0	...	0	
1	Piazza Duomo	0	0	0	0	0	0	0	0	0	...	0	
2	Piazza Duomo	0	0	0	0	0	0	0	0	0	...	0	
3	Piazza Duomo	0	0	0	0	0	0	0	0	0	...	0	
4	Piazza Duomo	0	0	0	0	0	0	0	0	0	...	0	

5 rows × 227 columns

Now we grouped it by the neighborhood, summing for each neighborhood all the category venues:

```
milan_grouped = milan_onehot.groupby('neighborhood').sum().reset_index()
milan_grouped
```

	neighborhood	Abruzzo Restaurant	Accessories Store	Adult Education Center	American Restaurant	Arcade	Argentinian Restaurant	Art Gallery	Art Museum	Asian Restaurant	...	Tram Station	Trat
0	Adriano	0	0	0	0	0	0	0	0	0	...	0	
1	Affori	0	0	0	0	0	0	0	0	0	...	0	
2	Baggio	0	0	0	0	0	0	0	0	0	...	0	
3	Bande Nere	0	0	0	0	0	0	0	0	0	...	0	
4	Barona	0	0	0	0	0	0	0	0	0	...	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
67	Viale Monza	0	0	0	0	0	0	0	0	0	...	3	
68	Vigentina	1	0	0	0	0	0	0	0	0	...	3	
69	Villapizzone	0	0	0	0	0	0	0	0	0	...	3	
70	Washington	0	0	0	0	0	0	0	0	1	...	0	
71	XXII Marzo	0	0	0	0	0	0	0	0	0	...	0	

Now we proceed to join it with previous dataframe:

```
df_joined = milan_grouped.join(df1,on='neighborhood')
```

Clean the data

We can see taht in total there are 226 venues categories. Investigating the categories we can note that we can proceed to group them in a same bigger category. In this way we can reduce our features from 226 to 34. The categories are the following:

```python
italian_restaurants = pd.DataFrame(np.sum(df_joined[['Italian Restaurant', 'Restaurant', 'Trattoria/Osteria',
                                                     'Mediterranean Restaurant']],axis=1),columns=['Italian Restaurants'])
pizza_places=pd.DataFrame(df_joined['Pizza Place'])
coffee_and_snacks = pd.DataFrame(np.sum(df_joined[['Café', 'Bar', 'Sandwich Place', 'Bistro','Piadineria','Coffee Shop','Breakfast Spot'
                                                   'Snack Place', 'Cafeteria', 'Juice Bar', 'Tea Room', 'College Cafeteria', 'Creperie',
                                                   'Salad Place', 'Buffet']],axis=1),columns=['Coffee and Snacks'])
hotels = pd.DataFrame(np.sum(df_joined[['Hotel', 'Bed & Breakfast', 'Hostel']],axis=1),columns=['Hotels'])
ice_cream_shops = pd.DataFrame(np.sum(df_joined[['Ice Cream Shop', 'Frozen Yogurt Shop']],axis=1),columns=['Ice Cream Shops'])
plazas = pd.DataFrame(np.sum(df_joined[['Plaza', 'Pedestrian Plaza']],axis=1),columns=['Plazas'])
markets_and_supermarkets = pd.DataFrame(np.sum(df_joined[['Supermarket', 'Convenience Store', 'Grocery Store', 'Food & Drink Shop',
                                                          'Market', 'Cheese Shop', 'Farmers Market', 'Health Food Store', 'Discount Stor
                                                          'Gourmet Shop', 'Chocolate Shop', 'Deli / Bodega', 'Butcher']],
                                                axis=1),columns=['Markets and Supermarkets'])
transports = pd.DataFrame(np.sum(df_joined[['Tram Station', 'Bus Stop', 'Bus Station', 'Train Station']],axis=1),columns=['Transports'])
japanese_and_sushi = pd.DataFrame(np.sum(df_joined[['Japanese Restaurant', 'Sushi Restaurant', 'Asian Restaurant', 'Ramen Restaurant',
                                              'Noodle House']],axis=1),columns=['Japanese and Sushi'])
nightlife_bars = pd.DataFrame(np.sum(df_joined[['Cocktail Bar', 'Wine Bar', 'Brewery', 'Hotel Bar','Beer Garden', 'Lounge', 'Karaoke Bar
                                                 'Dive Bar', 'Beer Bar', 'Speakeasy']],axis=1),columns=['Nightlife Bars'])
chinese_restaurants = pd.DataFrame(np.sum(df_joined[['Chinese Restaurant', 'Dim Sum Restaurant', 'Szechuan Restaurant']],
                                          axis=1),columns=['Chinese Restaurants'])
bakeries = pd.DataFrame(np.sum(df_joined[['Bakery', 'Dessert Shop', 'Cupcake Shop', 'Pastry Shop']],axis=1),columns=['Bakeries'])
parks = pd.DataFrame(np.sum(df_joined[['Park', 'Playground', 'Recreation Center', 'Skate Park', 'Garden Center',
                                       'Theme Park Ride / Attraction', 'Beach']],axis=1),columns=['Parks'])
pubs_and_diner = pd.DataFrame(np.sum(df_joined[['Pub', 'Diner', 'Steakhouse', 'Gastropub', 'Irish Pub']],
                                     axis=1),columns=['Pubs and Diner'])
retail_stores = pd.DataFrame(np.sum(df_joined[['Clothing Store', 'Boutique', 'Shoe Store', 'Department Store', 'Furniture / Home Store',
                                               'Pet Store', 'Cosmetics Shop', 'Gift Shop', 'Mobile Phone Shop', 'Sporting Goods Shop',
                                               'Bike Shop', 'Shopping Plaza', 'Men\'s Store', 'Accessories Store', 'Thrift / Vintage Sto
                                               'Kitchen Supply Store', 'Toy / Game Store', 'Candy Store', 'Flower Shop', 'Video Game Sto
                                               'Board Shop', 'Hobby Shop', 'Smoke Shop']],axis=1),columns=['Retail Stores'])
```

```python
books_and_music_stores = pd.DataFrame(np.sum(df_joined[['Bookstore', 'Record Shop', 'Music Store']],axis=1),columns=['Book and Music Sto
wellness = pd.DataFrame(np.sum(df_joined[['Gym / Fitness Center', 'Gym', 'Pool', 'Spa','Climbing Gym', 'Pool Hall',
                                          'Yoga Studio']],axis=1),columns=['Wellness'])
seafood_restaurant =pd.DataFrame(df_joined['Seafood Restaurant'])
sport_clubs = pd.DataFrame(np.sum(df_joined[['Soccer Field', 'Athletics & Sports', 'Soccer Stadium', 'Tennis Court',
                                             'Stadium', 'Tennis Stadium',
                                             'Golf Course']],axis=1),columns=['Sport Clubs'])
art_places = pd.DataFrame(np.sum(df_joined[['Art Gallery', 'Performing Arts Venue', 'Art Museum', 'Music Venue', 'Science Museum',
                                            'Theater', 'Comedy Club', 'Street Art', 'Public Art', 'Opera House', 'History Museum',
                                            'College Arts Building', 'Jazz Club', 'Ballroom']],axis=1),columns=['Art Places'])
vegan_restaurant = pd.DataFrame(df_joined['Vegetarian / Vegan Restaurant'])
fast_foods = pd.DataFrame(np.sum(df_joined[['Fast Food Restaurant', 'Burger Joint', 'Food Court', 'Fried Chicken Joint',
                                            'Food Truck']],axis=1),columns=['Fast Foods'])
pharmacies = pd.DataFrame(np.sum(df_joined[['Pharmacy', 'Medical Center']],axis=1),columns=['Pharmacies'])
nightclubs = pd.DataFrame(np.sum(df_joined[['Nightclub', 'Rock Club', 'Other Nightlife']],axis=1),columns=['Night Clubs'])
kebabs = pd.DataFrame(np.sum(df_joined[['Kebab Restaurant', 'Falafel Restaurant', 'Doner Restaurant']],axis=1),columns=['Kebabs'])
electronic_stores = pd.DataFrame(np.sum(df_joined[['Electronics Store', 'Hardware Store']],axis=1),columns=['Electronic Stores'])
monuments = pd.DataFrame(np.sum(df_joined[['Monument / Landmark', 'Church', 'Historic Site', 'Fountain',
                                           'Outdoor Sculpture']],axis=1),columns=['Monuments'])
international_ethnic_restaurants = pd.DataFrame(np.sum(df_joined[['Indian Restaurant', 'Mexican Restaurant', 'Spanish Restaurant',
                                                                  'Brazilian Restaurant', 'Moroccan Restaurant', 'Sri Lankan Restaur
                                                                  'Argentinian Restaurant', 'Roman Restaurant', 'German Restaurant',
                                                                  'Tapas Restaurant', 'Turkish Restaurant', 'Vietnamese Restaurant',
                                                                  'South American Restaurant', 'Filipino Restaurant', 'Greek Restaur
                                                                  'Peruvian Restaurant', 'Middle Eastern Restaurant', 'Thai Restaura
                                                                  'American Restaurant', 'French Restaurant', 'Russian Restaurant',
                                                                  'Lebanese Restaurant']],axis=1),columns=['International/Ethic Rest
cinemas = pd.DataFrame(np.sum(df_joined[['Multiplex', 'Movie Theater']],axis=1),columns=['Cinemas'])
malls = pd.DataFrame(np.sum(df_joined[['Shopping Mall', 'Outlet Store']],axis=1),columns=['Malls'])
wine_shops =pd.DataFrame(df_joined['Wine Shop'])
```

```python
local_restaurants= pd.DataFrame(np.sum(df_joined[['Sicilian Restaurant', 'Puglia Restaurant', 'Abruzzo Restaurant',
                                                  'Tuscan Restaurant', 'Sardinian Restaurant']],axis=1),columns=['Local Restaurants'])
studios = pd.DataFrame(np.sum(df_joined[['Design Studio', 'Photography Lab']],axis=1),columns=['Studios'])
offices_and_coworking = pd.DataFrame(np.sum(df_joined[['Coworking Space', 'Office',
                                                       'Business Service']],axis=1),columns=['Offices and Coworking'])
```

```python
df_final = pd.concat([italian_restaurants,pizza_places,coffee_and_snacks,hotels,ice_cream_shops,plazas,markets_and_supermarkets,
                      transports,japanese_and_sushi,nightlife_bars,chinese_restaurants,bakeries,parks,pubs_and_diner,retail_stores,
                      books_and_music_stores, wellness,seafood_restaurant,sport_clubs,art_places,vegan_restaurant,fast_foods,
                      pharmacies,nightclubs, kebabs,electronic_stores,monuments,international_ethnic_restaurants,cinemas,malls,
                      wine_shops,local_restaurants, studios,offices_and_coworking],axis=1)
```

```
df_final
```

	Italian Restaurants	Pizza Place	Coffee and Snacks	Hotels	Ice Cream Shops	Plazas	Markets and Supermarkets	Transports	Japanese and Sushi	Nightlife Bars	...	Kebabs	Electronic Stores	Monume
0	0	0	0	0	0	0	0	0	0	0	...	0	0	
1	1	3	2	2	0	0	2	0	0	1	...	1	0	
2	1	0	3	0	0	1	1	0	1	0	...	0	0	
3	3	2	3	3	2	1	1	1	2	0	...	0	0	
4	1	0	2	0	0	0	0	0	1	0	...	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
67	4	1	2	1	2	1	1	3	0	1	...	0	0	
68	1	1	3	0	1	1	2	3	0	0	...	0	0	
69	5	1	2	4	1	3	1	3	0	1	...	0	1	
70	5	2	3	1	1	1	3	0	2	1	...	0	1	
71	5	4	2	2	0	1	1	0	1	2	...	0	1	

72 rows × 34 columns

Finally, before to model our data, we create a new feature: the presence of a Metro station in the neighborhood, to insert on the final dataframe. We import it from a excel spreadsheet:

```
metro = pd.read_excel('Metro.xlsx',index_col='Neighborhood')
```

```
metro.head()
```

Neighborhood	Metro Station
Piazza Duomo	1
Pagano Metro	1
Ticinese	1
San Vittore	1
Guastalla	1

```
df_final=df_final.join(metro,on='neighborhood')
```

```
df_final['Price Category'] = df_joined['price_category'].values
```

```
df_final.set_index('neighborhood',inplace=True)
```

Machine Learning Model

First of all, we need to split the data in 2 parts: X with the features and y with the labels. Futhermore we use part of the same data to train the model and the other to test it. (we use 14 samples to test the model)

```
X=df_final.iloc[:,0:-1]
```

```
y=df_final.iloc[:,-1]
```

```
X_train, X_test, y_train, y_test = X[:-14], X[-14:], y[:-14], y[-14:]
```

To choose the better model, we have taken 6 classification models and proceeded to tune its parameters through the grid search procedure.

The best model that resulted is the Random forest with an accurancy of approx. 85% max_features= 'sqrt' and n_estimators= 100

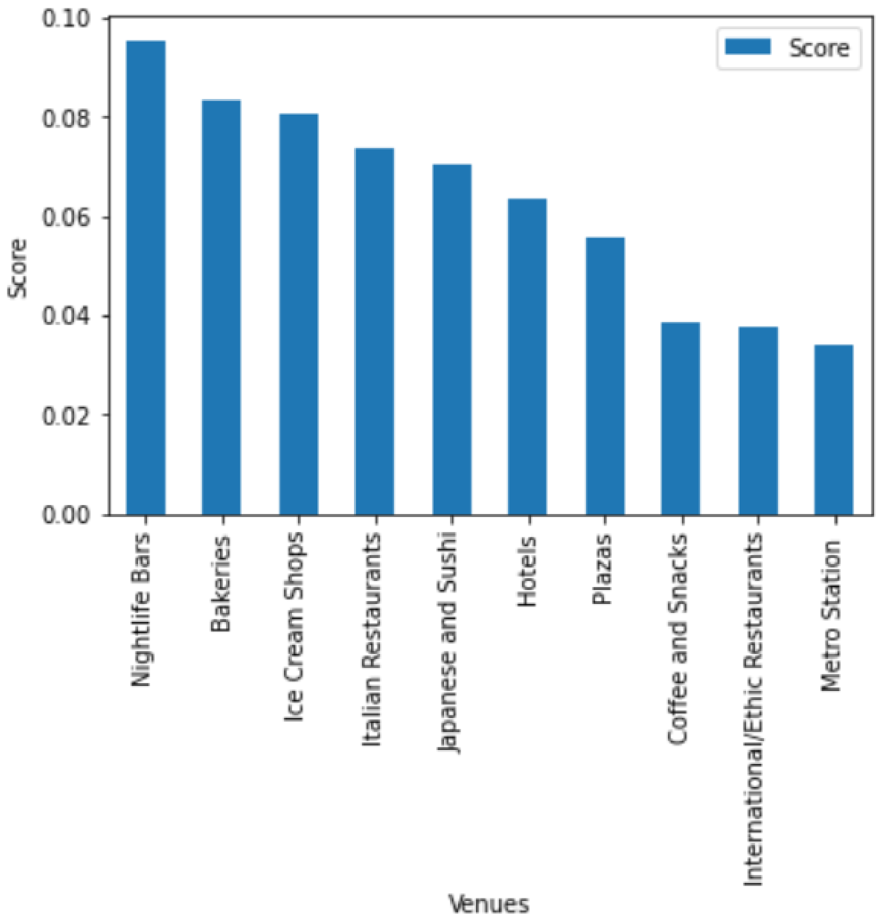
Random Forest ¶

```
# example of grid searching key hyperparameters for RandomForestClassifier
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier

# define models and parameters
model = RandomForestClassifier()
n_estimators = [10, 100, 1000]
max_features = ['sqrt', 'log2']

# define grid search
grid = dict(n_estimators=n_estimators,max_features=max_features)
cv = RepeatedStratifiedKFold(n_splits=8, n_repeats=3, random_state=1)
grid_search = GridSearchCV(estimator=model, param_grid=grid, n_jobs=-1, cv=cv, scoring='accuracy',error_score=0)
grid_result = grid_search.fit(X_train, y_train)

# summarize results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
```

```
Best: 0.846726 using {'max_features': 'sqrt', 'n_estimators': 100}
```

So we use the Random Forest Model to train our data.

We also analyzed the features importance from the model and plot it as follow:



As you can see the 10 most important features from the model refear primarly to venues related with the city nightlife, tourism and the presence of a Metro staton in the neighborhood.

Performance Measures

Accurancy through Cross Validation

```
from sklearn.model_selection import cross_val_score
cross_val_score(rnd_clf, X_train, y_train, cv=3, scoring="accuracy")
```

```
array([0.85      , 0.78947368, 0.78947368])
```

Precision, Recall and F1 Score

```
from sklearn.metrics import precision_score,recall_score

precision_score(y_test ,y_pred)
```

```
1.0
```

```
recall_score(y_test,y_pred)
```

```
0.75
```

```
from sklearn.metrics import f1_score
f1_score(y_test,y_pred)
```

```
0.8571428571428571
```

From this metrics the model seems to works pretty good. Now we can use the model to predict the test data and comapare it with the real data:

neighborhood		
Selinunte	1.0	1.0
Stadera	1.0	1.0
Stazione Centrale	2.0	2.0
Tibaldi	1.0	2.0
Ticinese	2.0	2.0
Tortona	2.0	2.0
Tre Torri	2.0	2.0
Trenno	1.0	1.0
Triulzo Superiore	1.0	1.0
Viale Monza	1.0	1.0
Vigentina	1.0	1.0
Villapizzone	1.0	2.0
Washington	2.0	2.0
XXII Marzo	2.0	2.0

From the comparison table we can appreciate that our model made an error only in 2 observation so we can confirm that the model works good.

Conclusion

The model had the aim to investigate the existence of some type of relationship between the house prices in different neighborhood of Milan and the venues that there are in the same neighborhood.

The model showed that the 10 most important venues to classify a neighborhood with a presence of low or medium/high price houses are: Nightlife Bars, Bakeries, Ice Cream Shops, Italian Restaurants, Japanese and sushi, Hotels, Plazas, Coffee and Snacks, International Restaurants and Metro station. This seems reasonable because the venues are related to cool neighborhoods of Milan where is concentrated the nightlife, tourism zones and neighborhoods with the presence of a Metro station that represent a key element to move easy from one part to other parts of the city.

From the performance metrics we have demonstrated that the model works pretty good in the prediction of the category price of the neighborhood.

We conclude that if we want estimate the price of an house, in addiction to the classic parameters like the number of rooms,square feet of the apartment and others, we can include also the presence of a venues in the neighborhood.