Porto Business School
/ University of Porto

VANWONEN

Projeto 3

PGBIA 13 – Grupo 3

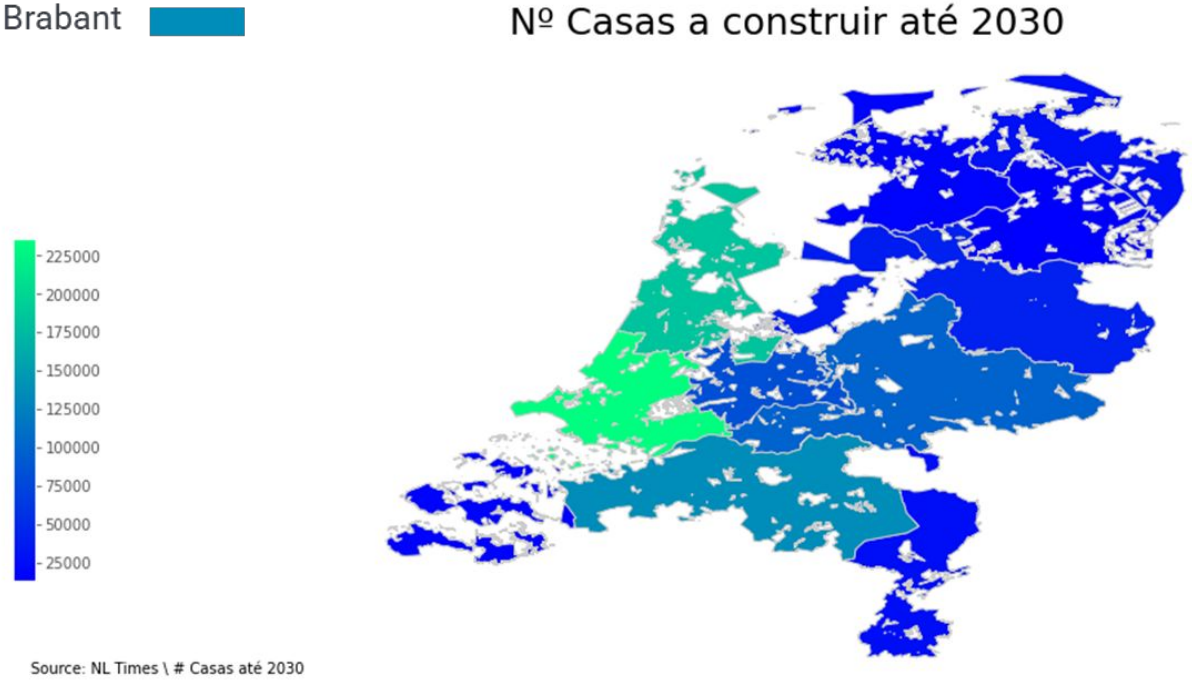
1. Business Understanding

Business Understanding | Enquadramento

Detalhes do mercado dos Países Baixos

- Províncias dos Países Baixos concordam em construir, até 2030, 917.000 novas casas, em que dessas aproximadamente 600.000 serão a preços acessíveis:
  - 350k serão casas para alugar e para comprar no segmento médio ~ aprox. 40% das casas
  - 250k serão casas de rendas sociais ~ aprox. 30% das casas
- As zonas com maior incidência serão:
  - South Holland
  - North Holland
  - North Brabant

A partir de 2023, todas as novas casas construídas terão que ser auto-sustentáveis

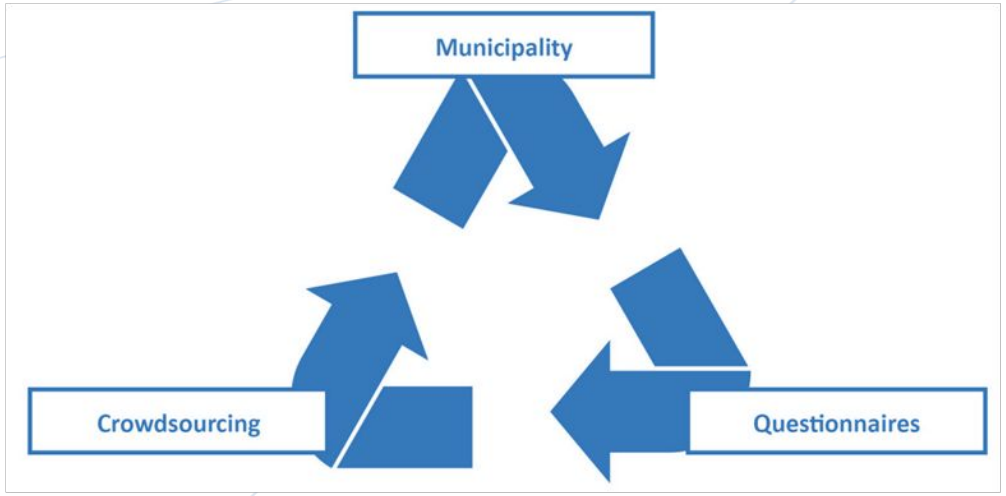



```
# Setting up Packages
import json
from datetime import datetime, timedelta
import requests
import pandas as pd
import geopandas as gpd
import matplotlib.pyplot as plt


# Print the map 1
# provincias.plot(figsize=(20, 20), edgecolor='white', linewidth=1, color='lightblue')


# Print the map 2
# Set the range for the choropleth
title = 'Nº Casas a construir até 2030'
col = 'Housing units pledged through 2030'
source = 'Source: NL Times \ # Casas até 2030'
vmin = provincias['Housing units pledged through 2030'].min()
vmax = provincias['Housing units pledged through 2030'].max()
cmap = 'winter'

# Create figure and axes for Matplotlib
fig, ax = plt.subplots(1, figsize=(20, 8))
# Remove the axis
ax.axis('off')
provincias.plot(column='Housing units pledged through 2030', ax=ax, edgecolor='0.8', linewidth=1, cmap=cmap)
# Add a title
ax.set_title(title, fontdict={'fontsize': '25', 'fontweight': '3'})
# Create an annotation for the data source
ax.annotate(source, xy=(0.1, .08), xycoords='figure fraction', horizontalalignment='left',
            verticalalignment='bottom', fontsize=10)

# Create colorbar as a legend
sm = plt.cm.ScalarMappable(norm=plt.Normalize(vmin=vmin, vmax=vmax), cmap=cmap)
# Empty array for the data range
sm._A = []
# Add the colorbar to the figure
cbaxes = fig.add_axes([0.15, 0.25, 0.01, 0.4])
cbar = fig.colorbar(sm, cax=cbaxes)
```

Business Understanding | Enquadramento

Fluxo de trabalhos de uma empresa do ramo imobiliário

Pesquisa e Aquisição de terreno → Desenvolvimento do Projeto → Venda a consumidor ou investidor



1. Criar casas e ambientes de vida para as gerações mais saudáveis, com espaços que fazem bem às pessoas e ao planeta, numa sociedade preparada para o futuro
2. Foco em desenvolver propriedades próprias ou para aluguer, desde transformações nos centros mais urbanos até novas construções em áreas rurais
3. Sonho de construir e desenvolver bairros inclusivos em que os moradores cuidam uns dos outros e se sentem em casa
4. Adaptar construção às necessidades das comunidades. Combinação com imóveis comerciais, serviços de saúde e sempre em conjuntos com todos os stakeholders
5. Desenvolvimento de conceitos e ideias para discussão com municípios. Aprofundamos o que uma cidade precisa e adaptam-se àquilo que a cidade é e às histórias que ali vivem.

Business Understanding | Objetivos

Objetivos do negócio de empresas imobiliárias | VanWonen

Mais e melhores contactos

- Quais os clientes a contactar
- Descobrir as melhores formas de falar com estes clientes, usando as melhores técnicas e métodos de machine learning e análise de dados

Mais Subscrições

- Aumentar o nº de demonstrações de interesse por parte dos potenciais clientes
- Garantir que as subscrições se transformam em momentos de contacto mais próximo com empresa

Aumento de Vendas

- Converter subscrições em Vendas de casas
- Garantir antecipação das vendas para fecho mais rápido de determinados projetos e foco nos seguintes

Business Understanding | Enquadramento

Segmento de clientes Assertivos e enérgicos

Segmento de clientes ambiciosos e dinâmicos

Segmento de clientes sociais e abertos

Segmento de clientes sérios, atenciosos e com pés no chão

Regressão:

Prever o preço máximo que o utilizador está disposto a pagar por um apartamento num determinado projeto.

Target: Max_value

Granularidade: Subscrição no projeto

Exemplos de Features: Dados de cliente como salário anual, agregado familiar, área do apartamento e número de quartos desejado e localização.

Classificação:

Verificar se os clientes desejam lugar de estacionamento.

Target: Lugar de Estacionamento (0/1)

Granularidade: Subscrição no projeto

Exemplos de Features: Dados de cliente como salário anual, agregado familiar, área do apartamento e número de quartos desejado e localização.

Porto Business School / University of Porto

Recomendação:

Recomendar um projecto a um utilizador usando *collaborative filtering*.

Target: ID de Projeto

Granularidade: User

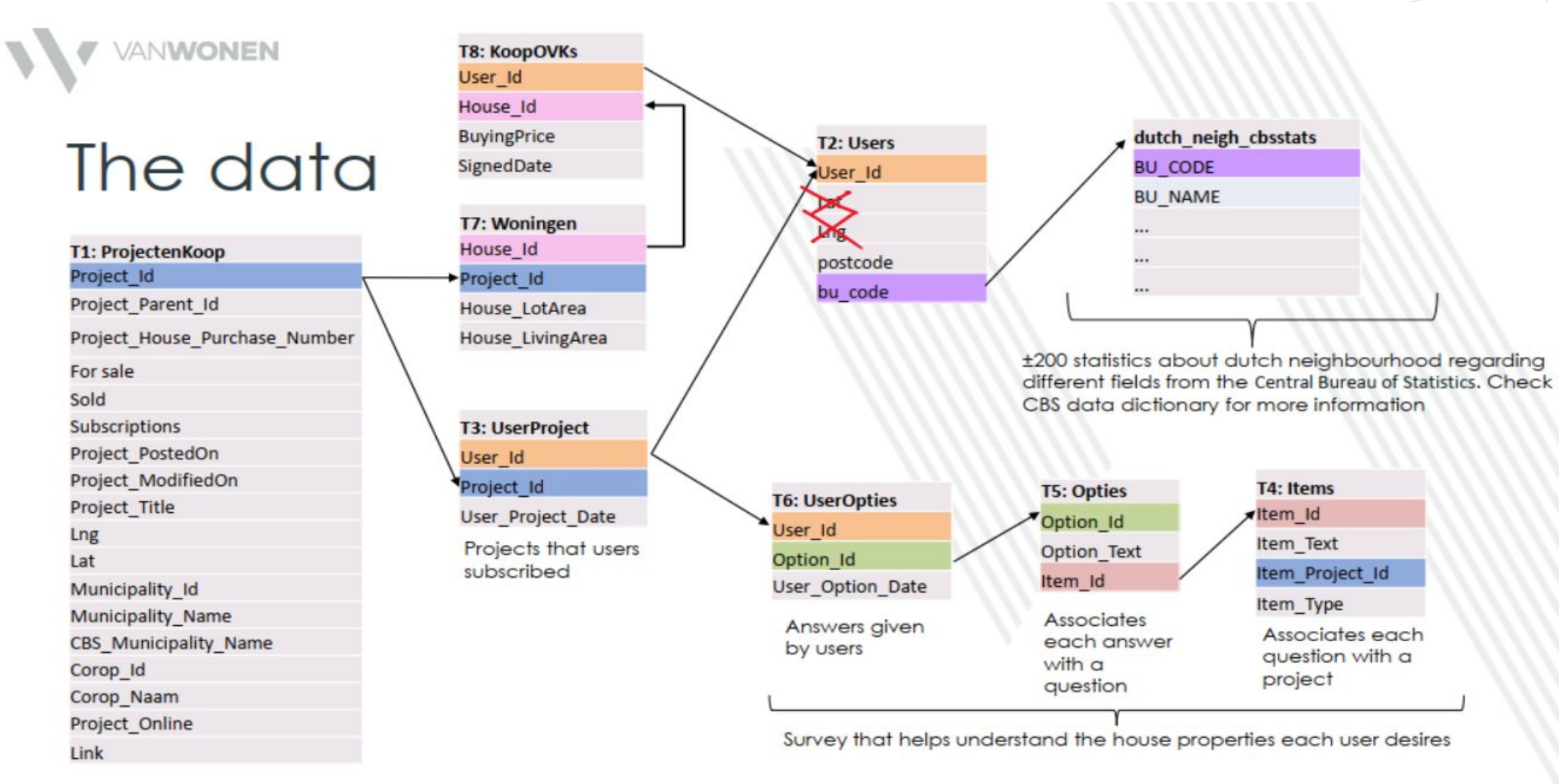
Exemplos de Features: Uset_ID vs Project_ID

user	proj1	proj2	proj3	proj4	proj5
1	1	0	0	1	0
2	1	0	1	1	0
3	1	0	0	1	0
4	1	1	0	1	0

2. Data Understanding

Data understanding | Informação geral

Esquema de base de dados da empresa



Data understanding | Dicionário

Exemplo de algumas métricas e o seu significado

Table ID	Table Name	Feature	Missing Values	Total Data	Distincts	All clear (Y/N)	Meaning
T4	Items	Item_Id	0	5697	5697	Y	Question ID (de cada projeto)
T4	Items	Item_Text	0	5697	977	Y	Dutch Question (de cada projeto)
T4	Items	Item_Project_Id	0	5697	81	Y	Project ID
T4	Items	Item_Type	0	5697	3	N	Plataforma usada p/ preencher o inquerito?
T4	Items	Item_Text_translated	0	5697	908	Y	Question translated to english (de cada projeto)
T5	Opties	Option_Id	0	31855	31855	Y	ID das resposta às perguntas (de cada user)
T5	Opties	Option_Text	0	31855	31855	Y	Resposta às perguntas (Holandês) (de cada user)
T5	Opties	Item_Id	0	31855	5697	Y	Question ID (de cada projeto)
T5	Opties	Option_Text_Translated	0	31855	2416	Y	Resposta às perguntas (Inglês) (de cada user)
T6	User Opties	User_Id	0	647700	27576	Y	User_Id
T6	User Opties	Option_Id	0	647700	10617	Y	ID das resposta às perguntas (de cada user)
T6	User Opties	User_Option_Date	0	647700	50444	Y	Data que o user respondeu às opções de resposta às perguntas

Dicionario_Vanwonen.xlsx

Data Understanding | Data Exploration – Elementos do negócio

Exemplos de exploração de dados

Data Understanding | Data Exploration – Elementos do negócio

Exemplos de exploração de dados

Data Understanding | Data Exploration - Projetos

Detalhes de Projetos e a sua localização

173

Projetos Existentes à data *

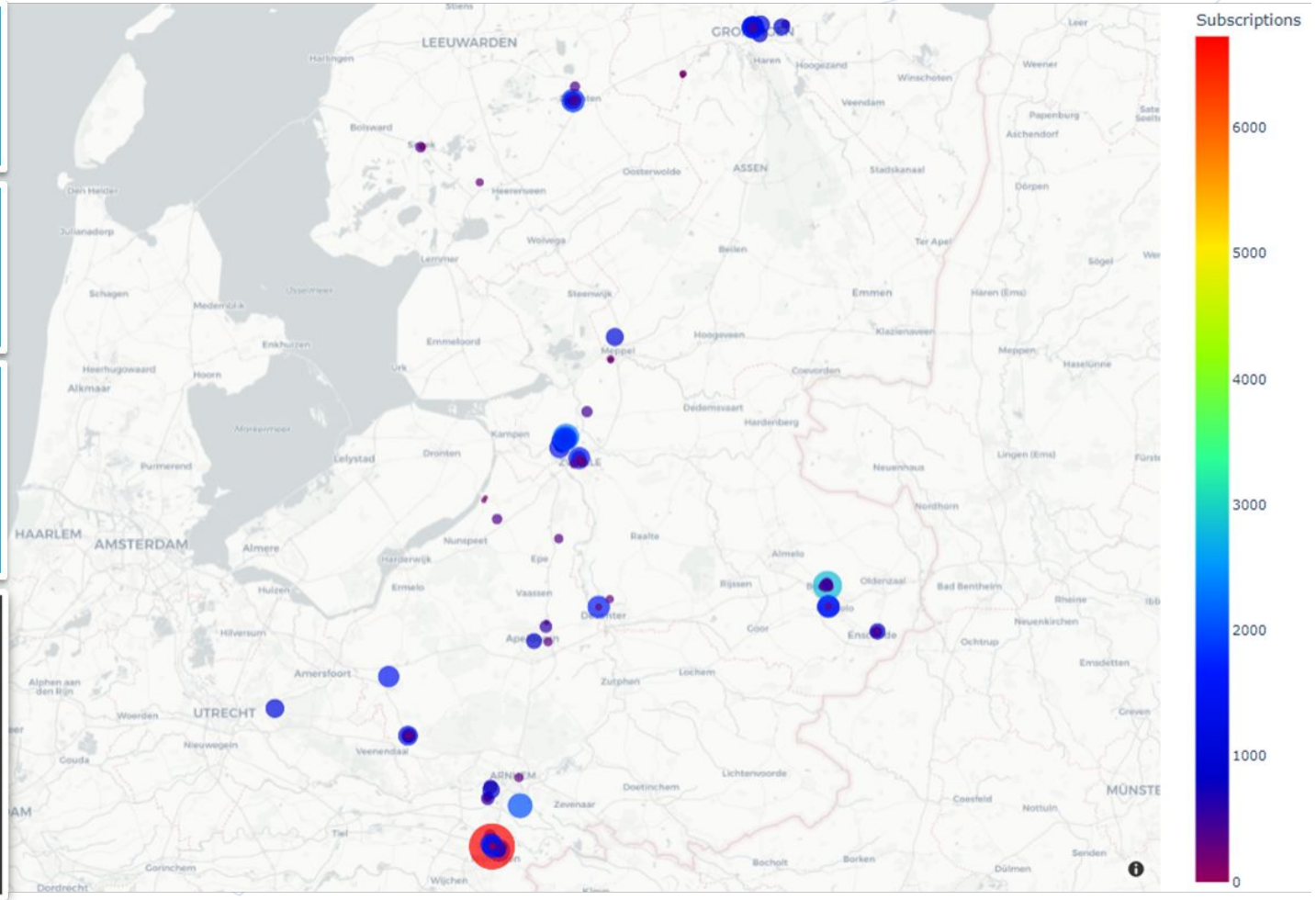
22

Projetos com casas à venda à data*

TOP 3 Projetos com mais subscrições:

Heart of the Waal jump – 6728
Bornsche Sizes – 2809
Breezicht North – 2317

	Corop_Naam	Number of Projects	Number of Houses	Number of Houses Sold	avg(BuyingPrice)
0	Noord-Overijssel	46	704	615	360839.045455
1	Arnhem/Nijmegen	36	1000	830	412222.143373
2	Twente	23	387	359	375292.339833
3	Veluwe	19	388	353	333682.141643
4	Overig Groningen	17	556	433	347711.161663
5	Zuidoost-Friesland	15	216	176	307355.113636
6	Zuidwest-Drenthe	6	53	50	293330.000000
7	Zuidwest-Friesland	5	54	33	350787.878788
8	Zuidwest-Overijssel	4	27	10	364500.000000
9	Agglomeratie 's-Gravenhage	1	0	0	NaN
10	Utrecht	1	0	0	NaN

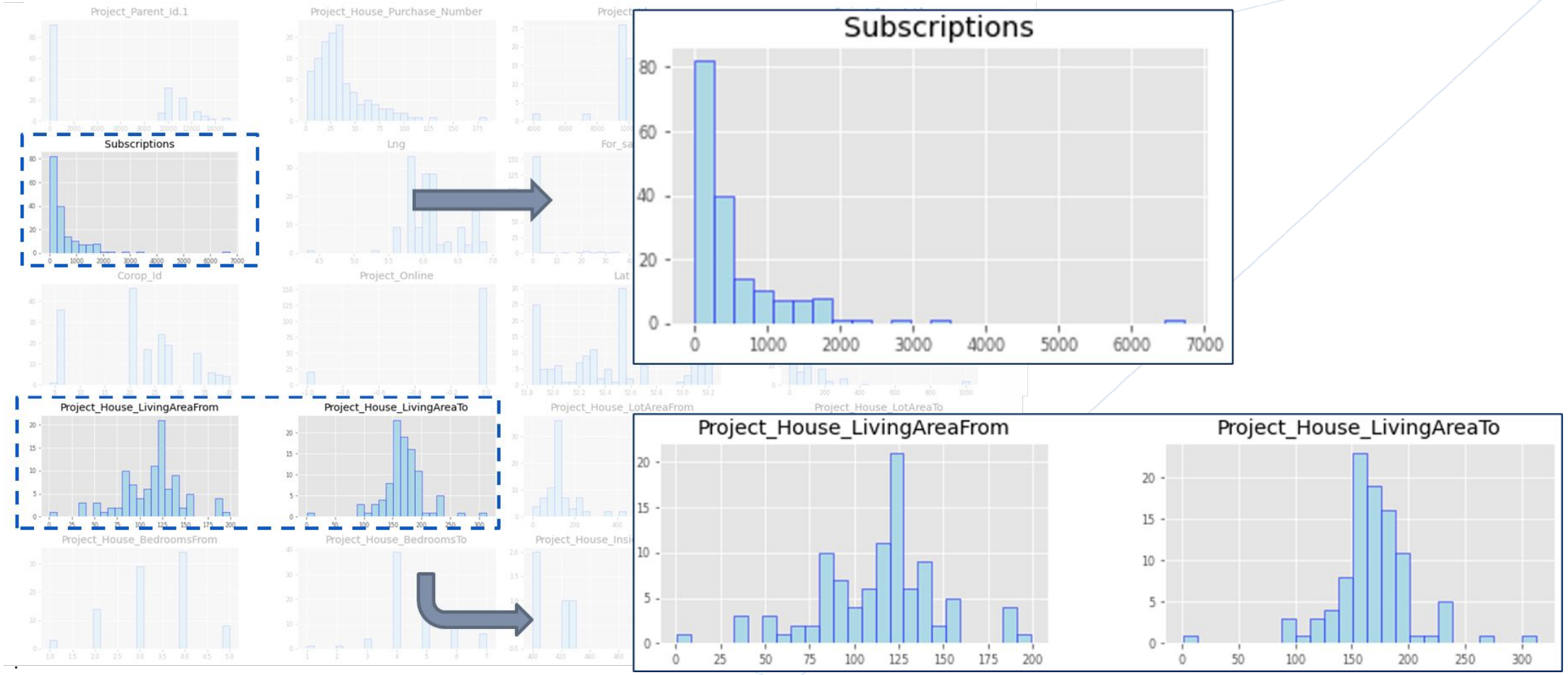


Data Understanding | Data Exploration - Projetos

Histogramas das features da tabela de projetos

Data Understanding | Data Exploration - Projetos

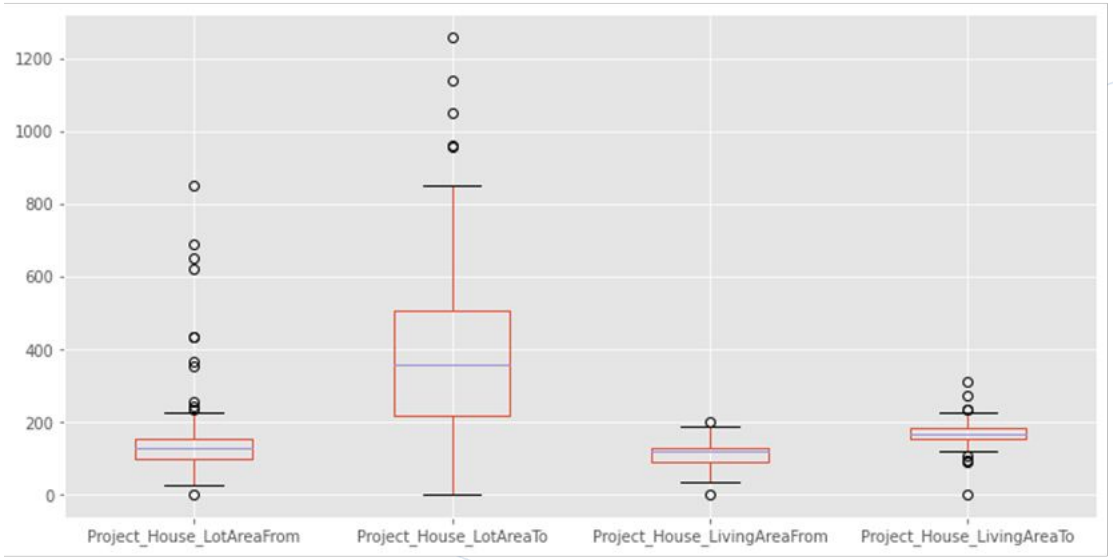
Histogramas das features da tabela de projetos



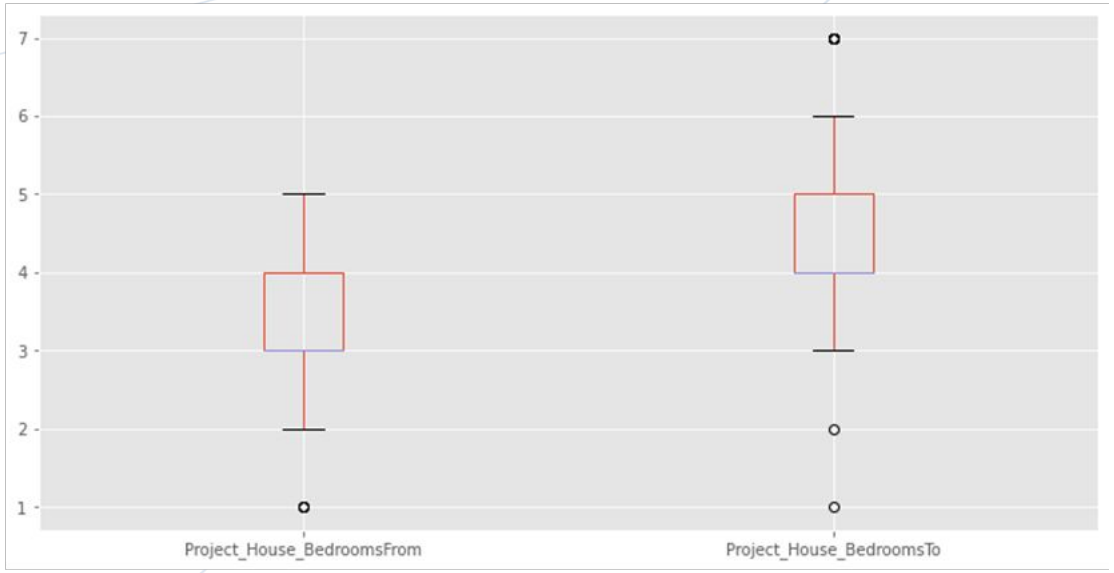
Data Understanding | Data Exploration - Projetos

Boxplots de features quantitativas dos projetos

Área de Lote vs Área Int. Casa



Nº de Quartos



Boxplots de área do Lote (por projeto) mostram muito maior dispersão havendo muita diferença entre as áreas mais pequenas e as áreas maiores das casas

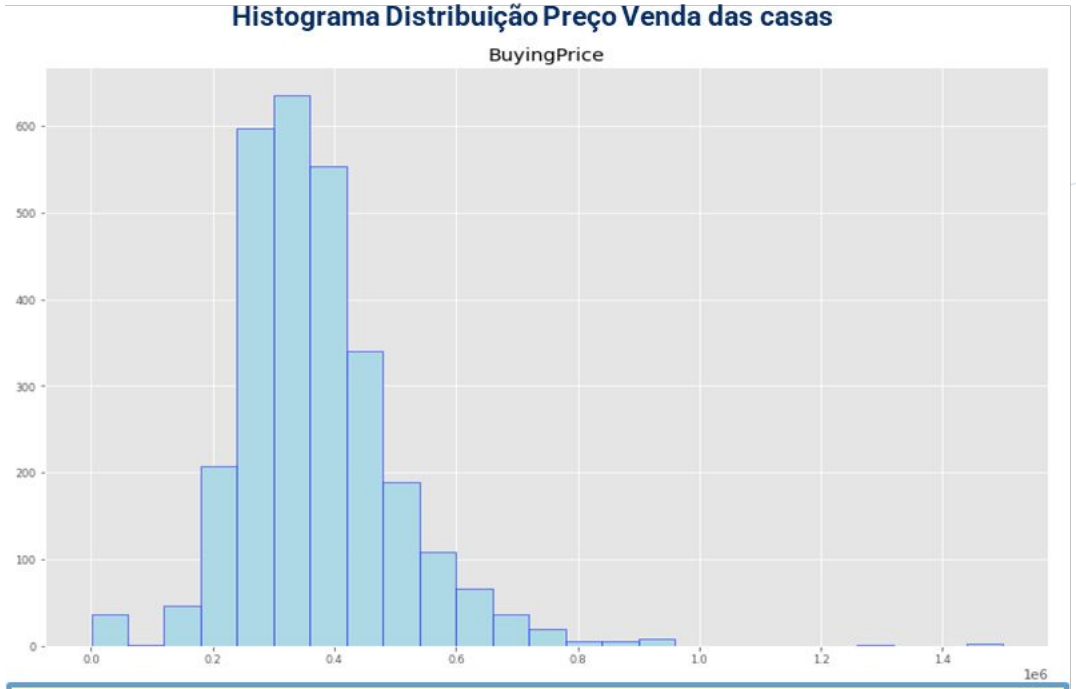
No nº de quartos vemos um valor de mediana = ao valor do Q1, mostrando uma grande importância dos projetos com casas de 3 casas e de 4 casas

Data Understanding | Data Exploration – Elementos do negócio

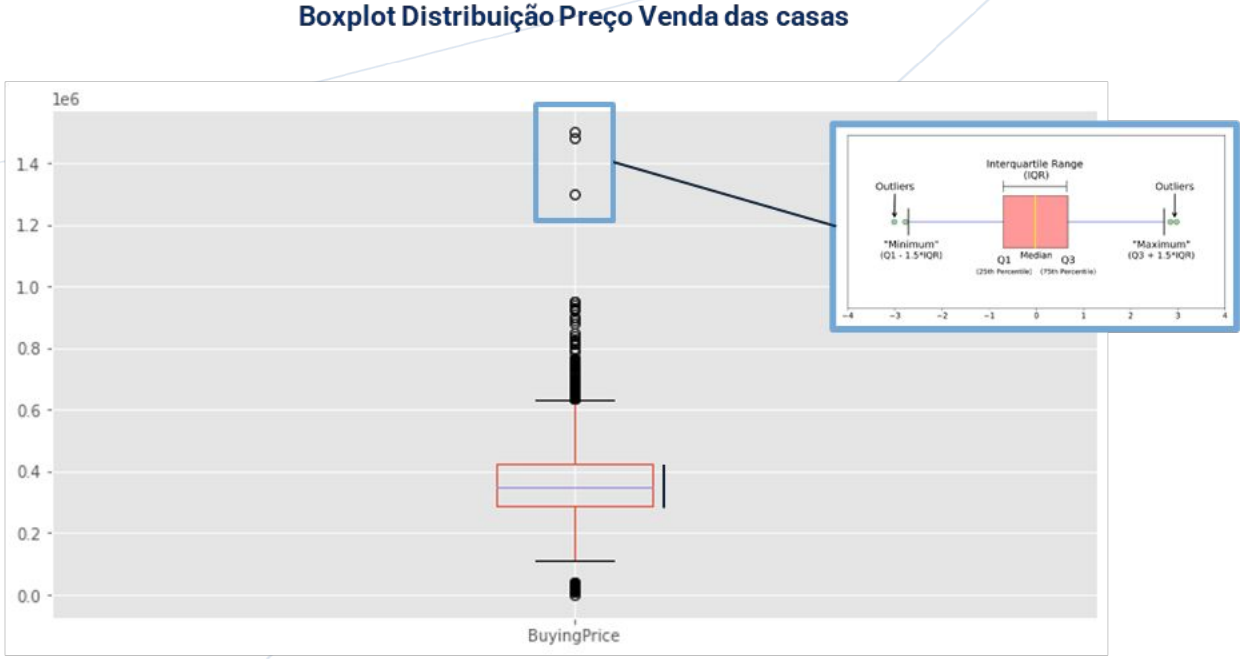
Exemplos de exploração de dados

Data Understanding | Data Exploration – Casas

Distribuição de features quantitativas das casas



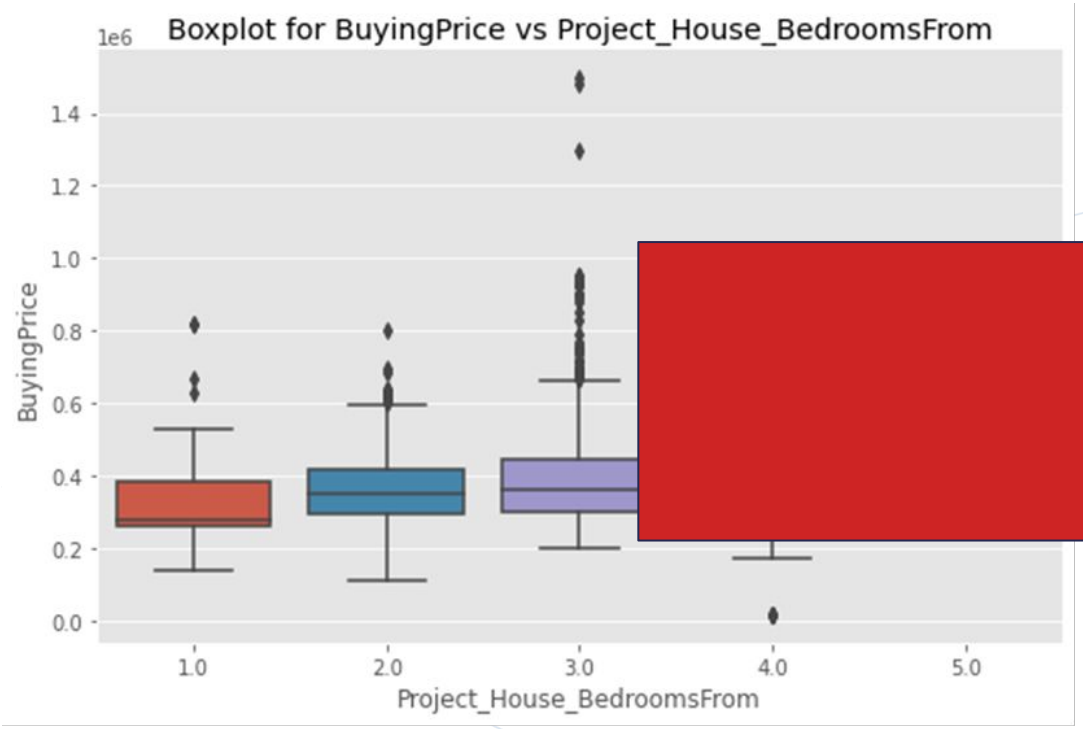
O preço de vendas das casas mostra uma distribuição alongada à direita, com pico de vendas aproximado dos 400 k€

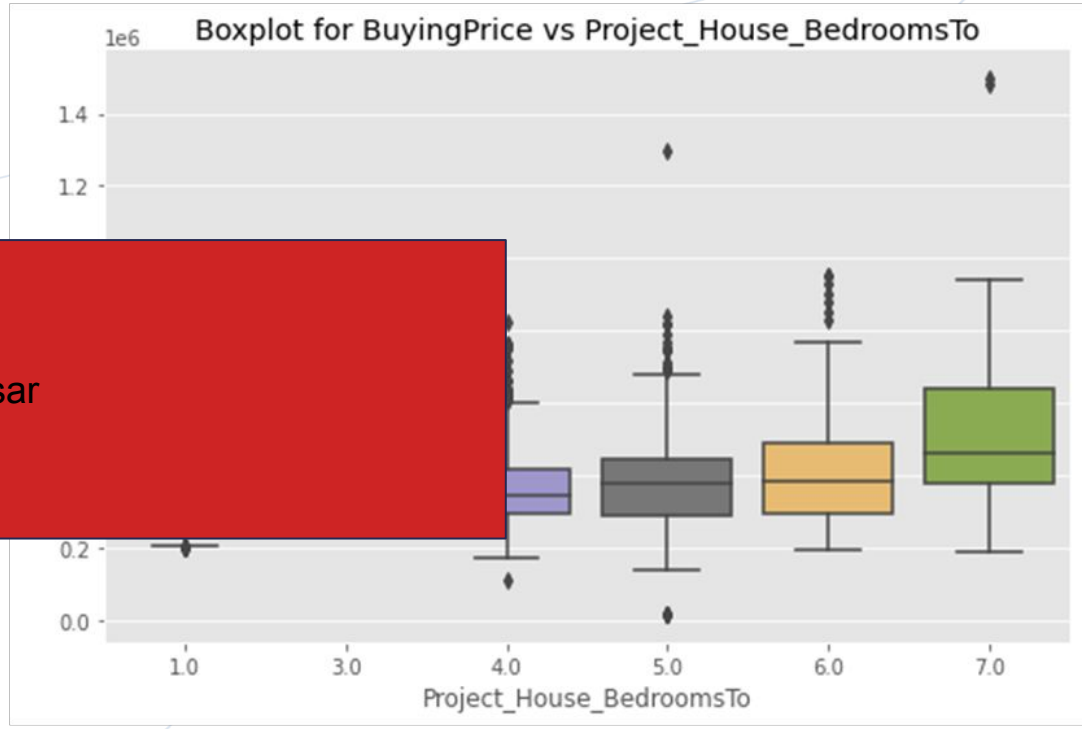


O boxplot mostra-nos a existência de algumas casas que foram compradas "fora de preço" sendo elas outliers (valores acima de 1.2 M€)

Data understanding | Data Exploration – Casas

Boxplots features das casas vendidas



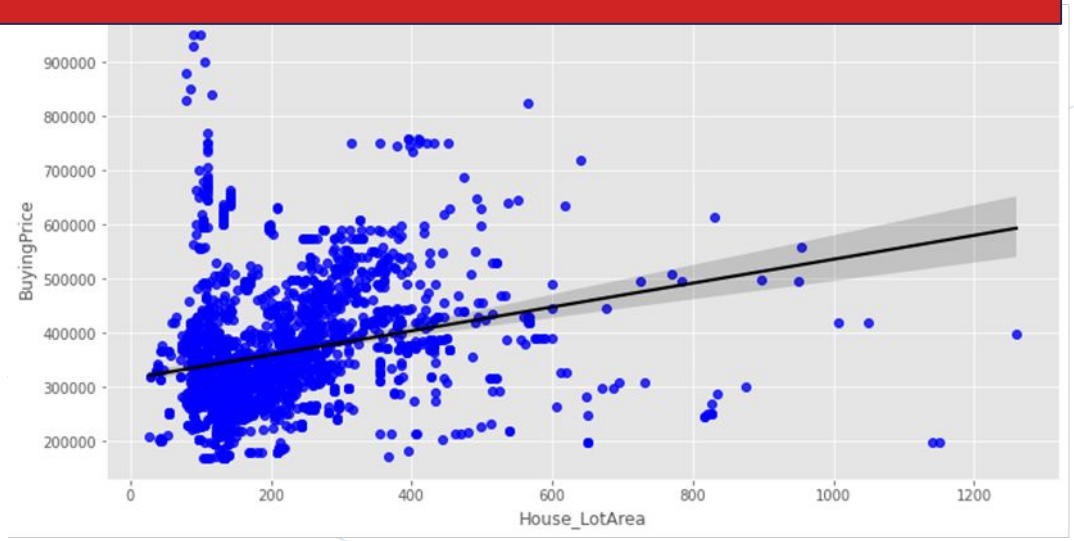


repensar

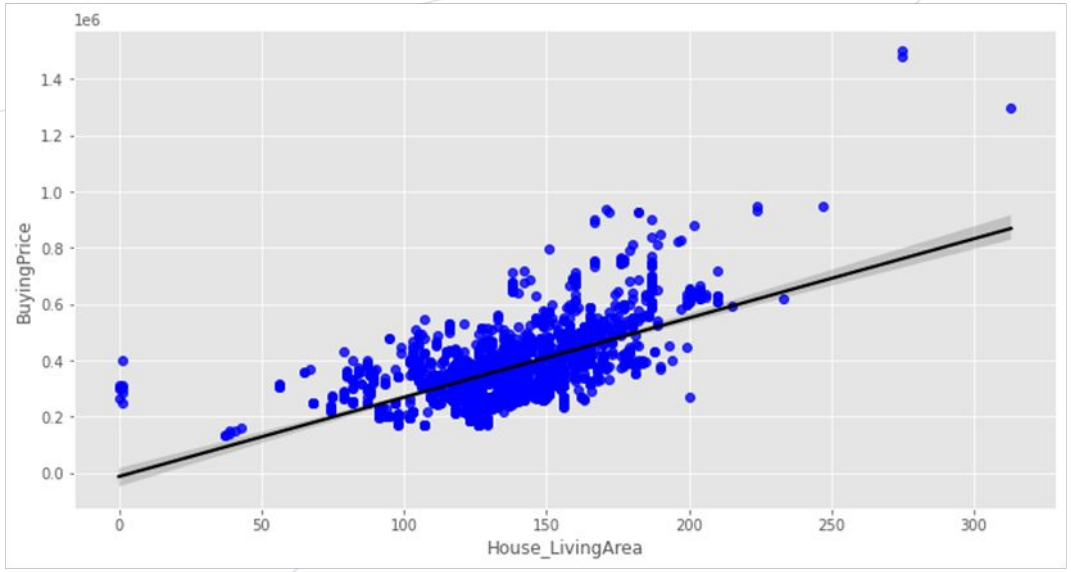
Ao observamos a variação do preço das casas vendidas por quarto, naturalmente que existe uma relação de maior preço de venda quanto mais quartos existir na casa

Data understanding | Data Exploration – Casas

ver se conseguimos colocar cada bola por regiao (por cores)



Dispersão da área interiores das casas vendidas



Maior dispersão das áreas dos lotes mostra que, independentemente do valor da casa, existem com alguma frequência casas baratas com lotes grandes e casas caras com lotes pequenos

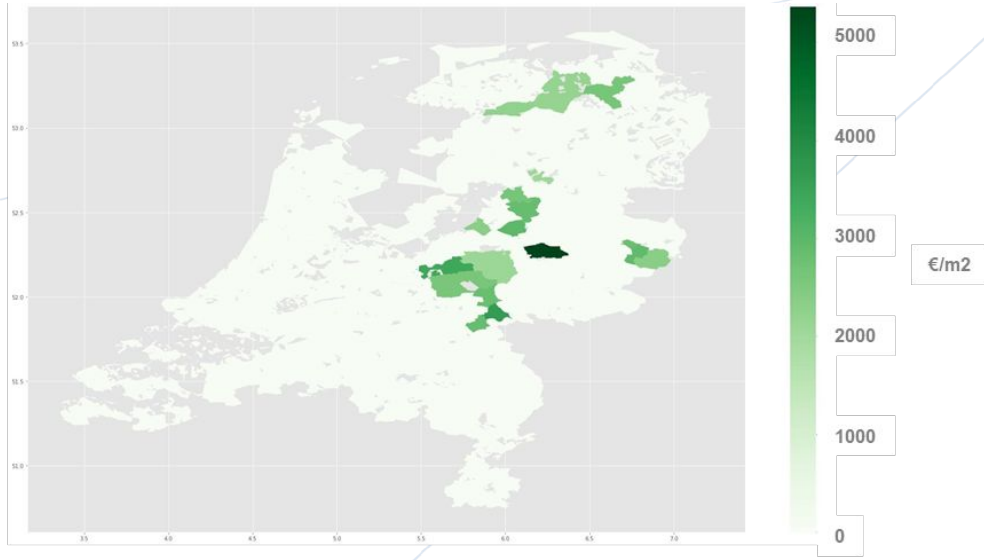
Por outro lado, quando nos referimos à área interior da casa, já parece existir uma menor dispersão e consequente relação mais direta entre preço e área

Data understanding | Data Exploration – Casas

Porto Business School / University of Porto

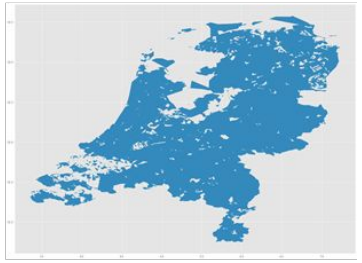
Preço médio por m2 das casas vendidas pela VanWonen

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import csv
import json
import os,re
import networkx as nx
import geopandas as gpd

mydb = mysql.connector.connect(**config)

query = 'SELECT * FROM grupo3.Houses_Sales_T8'
data8 = pd.read_sql(query, con = mydb)

query = 'SELECT * FROM grupo3.Houses_FromProject_T7'
data7 = pd.read_sql(query, con = mydb)

query = 'SELECT * FROM grupo3.Projects_T1'
data1 = pd.read_sql(query, con = mydb)

query = 'SELECT * FROM grupo3.INE_NL_T0'
data0 = pd.read_sql(query, con = mydb)

df_House_Sales = data8[['House_Id','BuyingPrice']]
df_House_Sales

df_House_Proj = data7[['House_Id','Project_Id','House_LivingArea']]
df_House_Proj

df_proj = data1[['Project_Id','Municipality_Name']]
df_proj

df_ine = municipio
df_ine

data_temp = df_House_Sales.merge(df_House_Proj,'left',on=None,left_on='House_Id',right_on='House_Id')
data_temp2 = data_temp.merge(df_proj,'left',on=None,left_on='Project_Id',right_on='Project_Id')


data_temp2 = data_temp2.dropna() # do total de 2861, temos 2659 resultados sem N/A (normalmente por falta de área)
data_temp2

data_area_preco = data_temp2.groupby(['Municipality_Name'],as_index=False).sum()[['Municipality_Name','BuyingPrice','House_LivingArea']]

area=data_area_preco.BuyingPrice/data_area_preco.House_LivingArea

data_area_preco['preco_por_m2']=area.where(data_area_preco.BuyingPrice > 0)
data_area_preco = data_area_preco[['Municipality_Name','preco_por_m2']]

data_area_preco = data_area_preco.merge(df_ine,'left',on=None,left_on='Municipality_Name',right_on='GM_NAAM')
data_area_preco

data_area_preco_final = municipio.merge(data_area_preco,'left',on=None,left_on='GM_NAAM',right_on='Municipality_Name')[['geometry','preco_por_m2']]
#data_area_preco_final = data_area_preco_final.dropna(subset=['Municipality_Name']) ## corrigir este merge para manter todos os GM_NAAM (sao 355)
data_area_preco_final['preco_por_m2']=data_area_preco_final['preco_por_m2'].replace(np.nan,0)
data_area_preco_final
```



	geometry_x	preco_por_m2
0	MULTIPOLYGON (((4.27163 52.03425, 4.27090 52.0...	0.000000
1	POLYGON ((5.29037 51.67270, 5.29040 51.67280, ...	0.000000
2	MULTIPOLYGON (((6.63111 52.95175, 6.63120 52.9...	0.000000
3	POLYGON ((4.72229 52.22401, 4.72271 52.22709, ...	0.000000
4	MULTIPOLYGON (((6.47732 51.87260, 6.47740 51.8...	0.000000
...	...	...
350	POLYGON ((4.62464 51.42560, 4.62461 51.42561, ...	0.000000
351	POLYGON ((6.22107 52.09926, 6.22107 52.09926, ...	0.000000
352	POLYGON ((6.07087 52.55739, 6.07061 52.55731, ...	2637.883646
353	POLYGON ((4.59469 51.80565, 4.59327 51.80578, ...	0.000000
354	POLYGON ((6.09842 52.46963, 6.09821 52.46985, ...	2637.214379

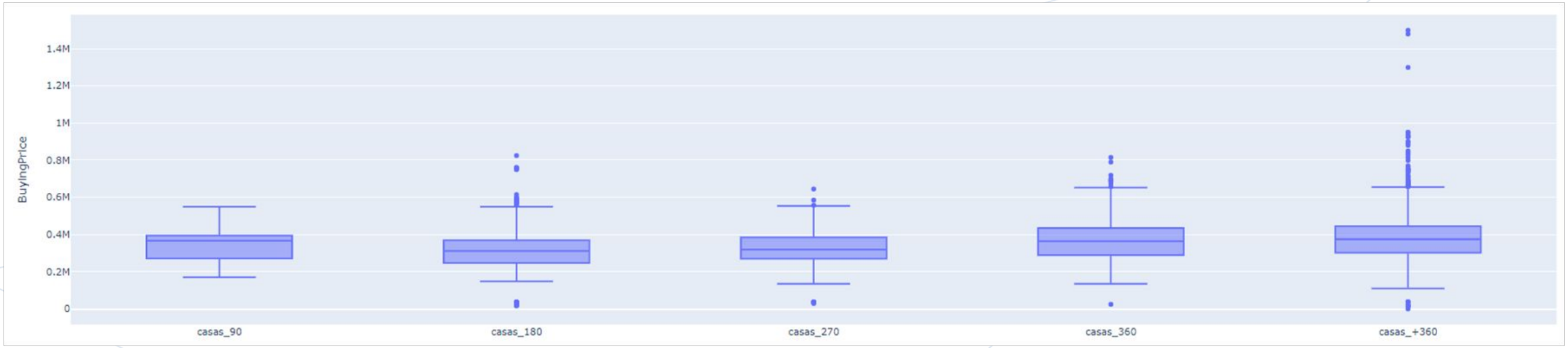
Demonstração de Zonas geográficas do país



Data understanding | Data Exploration – Casas

Venda das casas ao longo do tempo | 90 dias | 180 dias | 270 dias | 360 dias | +360 dias

Boxplot das casas vendidas por intervalo de tempo

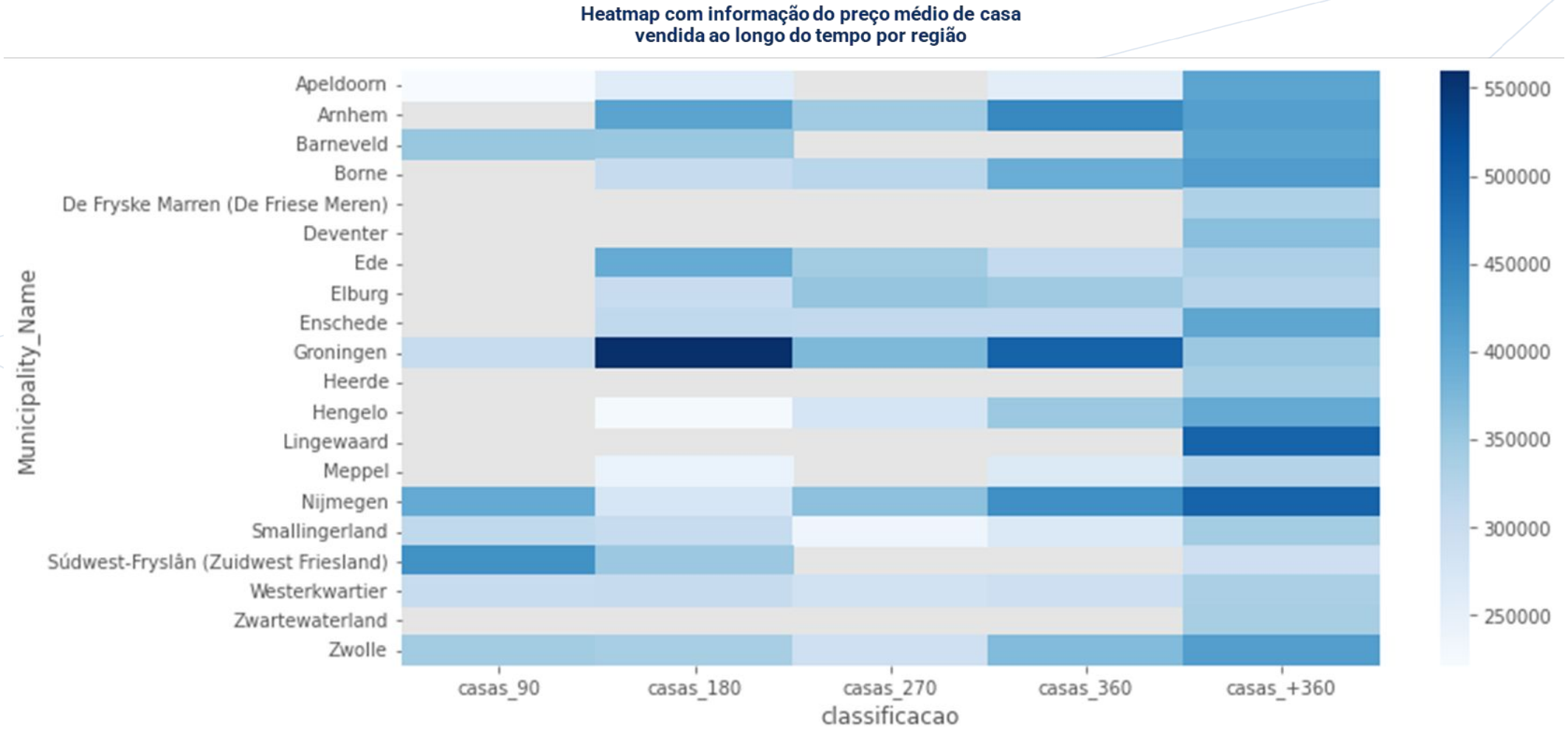


Outlier máx.	-	825	645	815	1499
Máximo	550	550	554	653	655
Q3	395	370	385	435	445
Mediana	367	312	319	365	375
Q1	271	248	270	290	303
Mínimo	170	149	135	135	110
Outlier mín.	-	17	30	25	-

Valores em k€

Data understanding | Data Exploration – Casas

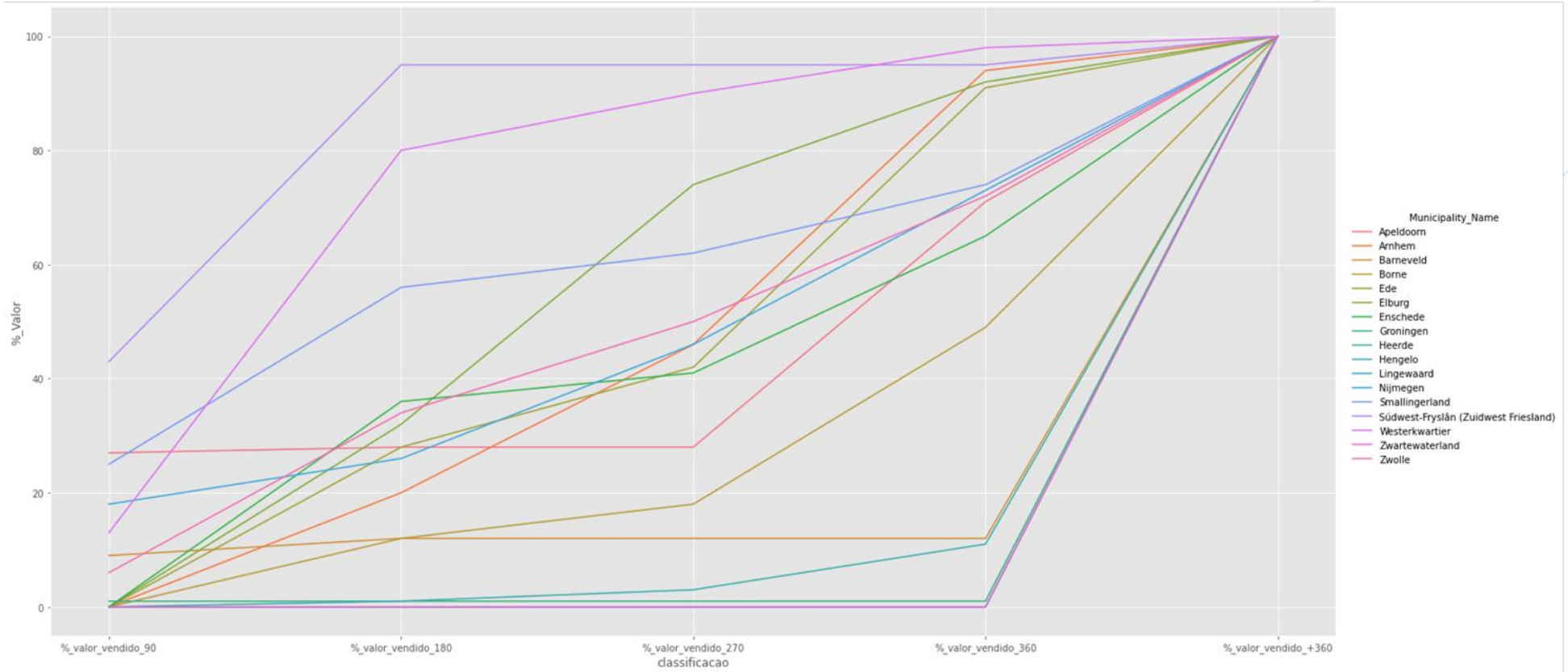
Venda das casas ao longo do tempo | 90 dias | 180 dias | 270 dias | 360 dias | +360 dias



Data understanding | Data Exploration – Casas

Venda das casas ao longo do tempo | 90 dias | 180 dias | 270 dias | 360 dias | +360 dias

Gráfico evolutivo de casas vendidas por região
(% do € valor das casas – Apenas para projetos com 100% das casas vendidas)



Regiões em que 90% das casas são vendidas no primeiro ano de venda:

- **Zwartewaterland**
- **Sudwest-Fryslan**
- **Arnhem**

Regiões que, no primeiro ano de venda, vendem menos de 15% dos apartamentos:

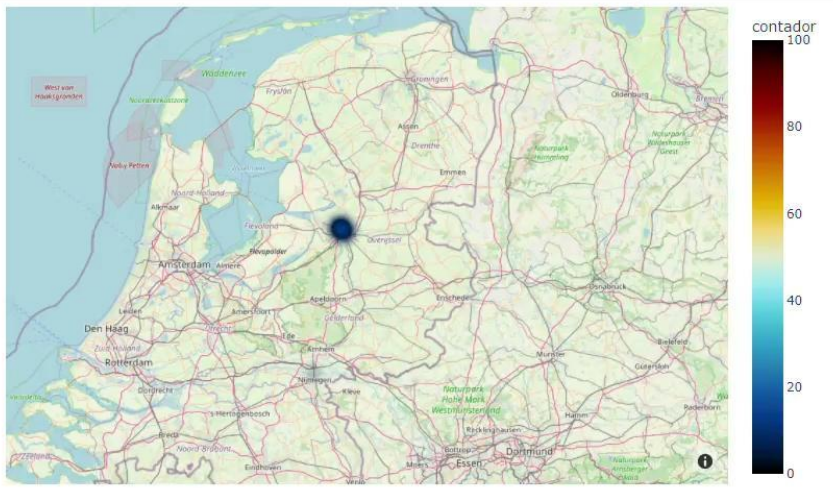
- **Westerkwartier**
- **Groningen**
- **Barneveld**

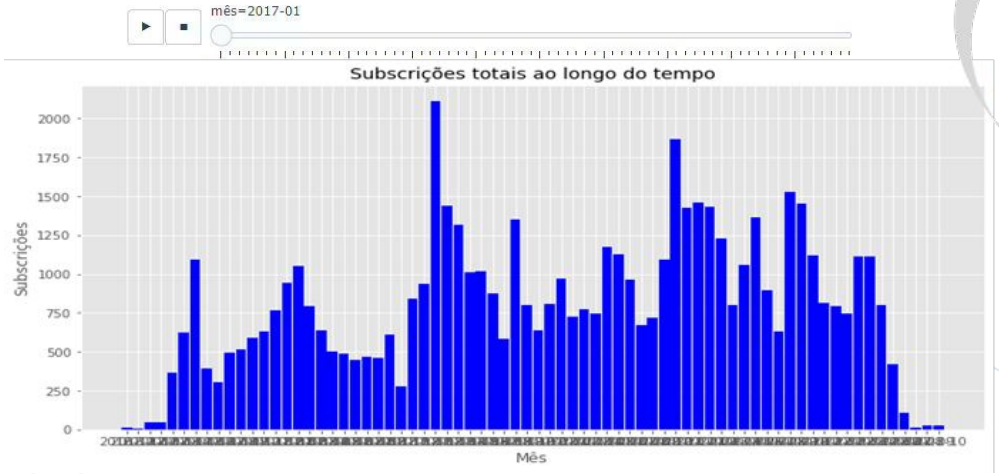
Data Understanding | Data Exploration – Elementos do negócio

Exemplos de exploração de dados

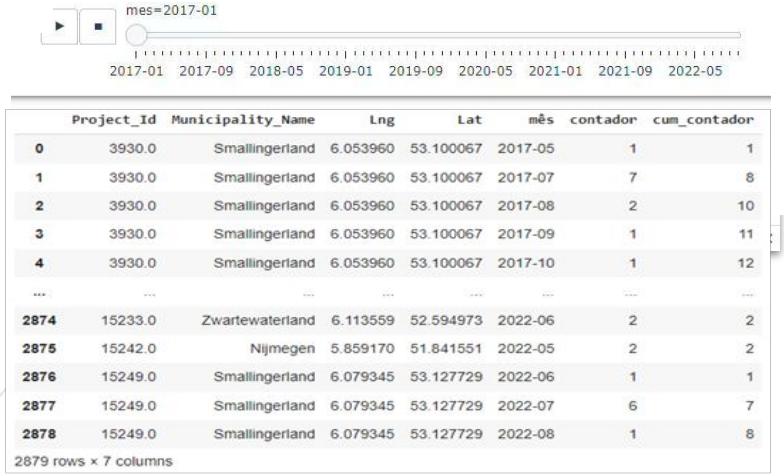
Data understanding | Data Exploration – Users

Evolução das subscrições dos users com os diferentes projetos da VanWonen





Contador mensal



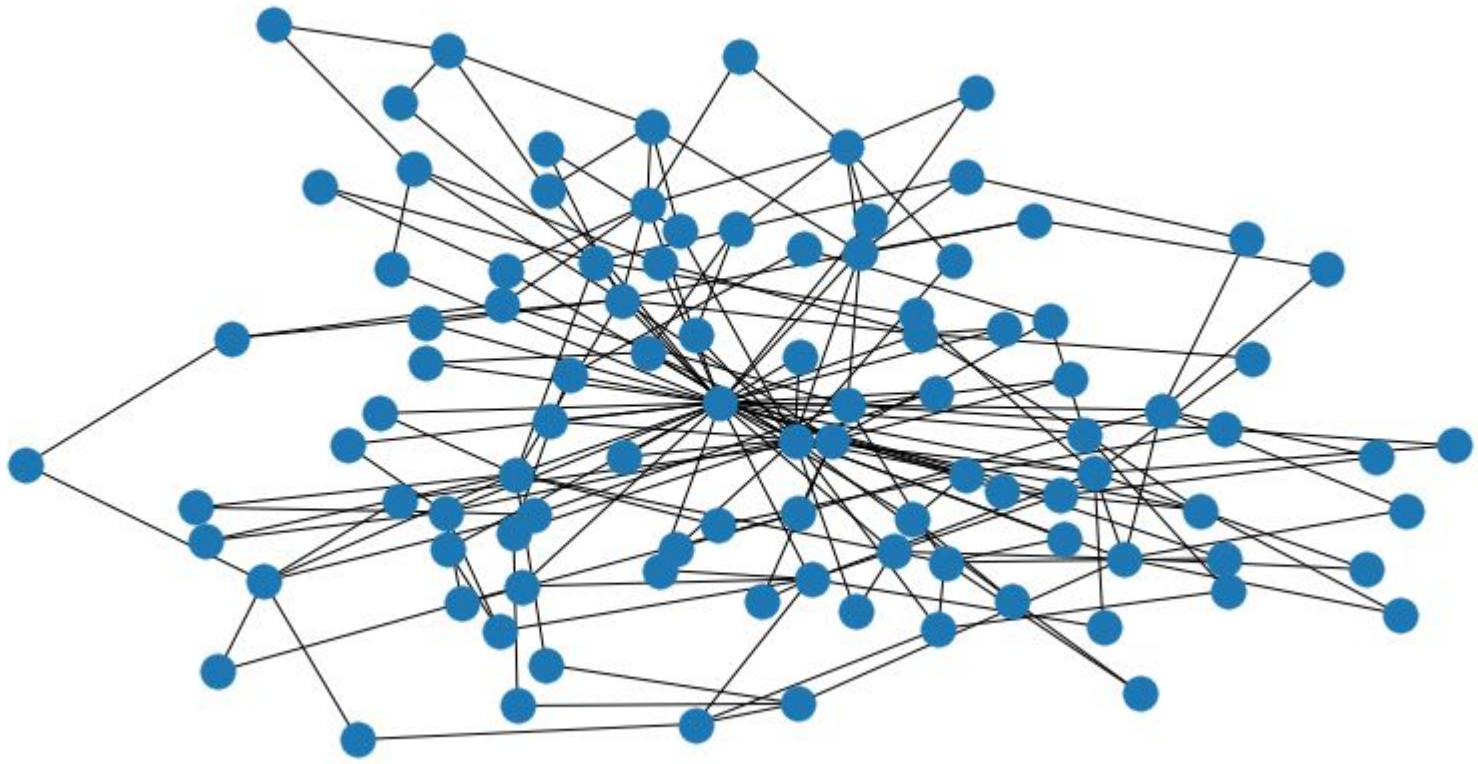
	Project_Id	Municipality_Name	Lng	Lat	mês	contador	cum_contador
0	3930.0	Smallingerland	6.053960	53.100067	2017-05	1	1
1	3930.0	Smallingerland	6.053960	53.100067	2017-07	7	8
2	3930.0	Smallingerland	6.053960	53.100067	2017-08	2	10
3	3930.0	Smallingerland	6.053960	53.100067	2017-09	1	11
4	3930.0	Smallingerland	6.053960	53.100067	2017-10	1	12
...	...	...	...	...	...	...	...
2874	15233.0	Zwartewaterland	6.113559	52.594973	2022-06	2	2
2875	15242.0	Nijmegen	5.859170	51.841551	2022-05	2	2
2876	15249.0	Smallingerland	6.079345	53.127729	2022-06	1	1
2877	15249.0	Smallingerland	6.079345	53.127729	2022-07	6	7
2878	15249.0	Smallingerland	6.079345	53.127729	2022-08	1	8

2879 rows × 7 columns

Contador acumulativo

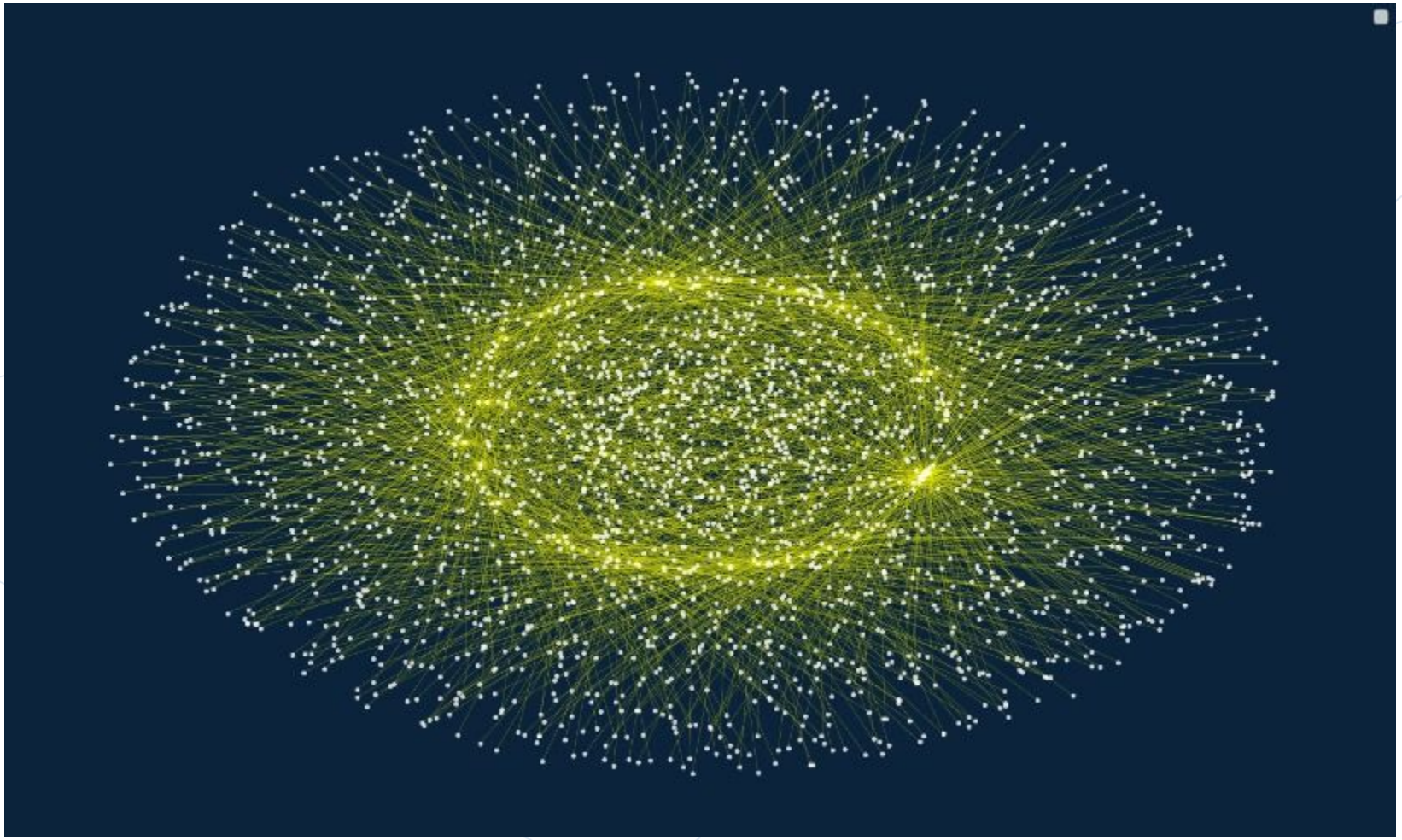
SNA | Users vs Income

Amostra com o Dataset com subscritores a todos os projetos vs rendimento



SNA | Users vs Project

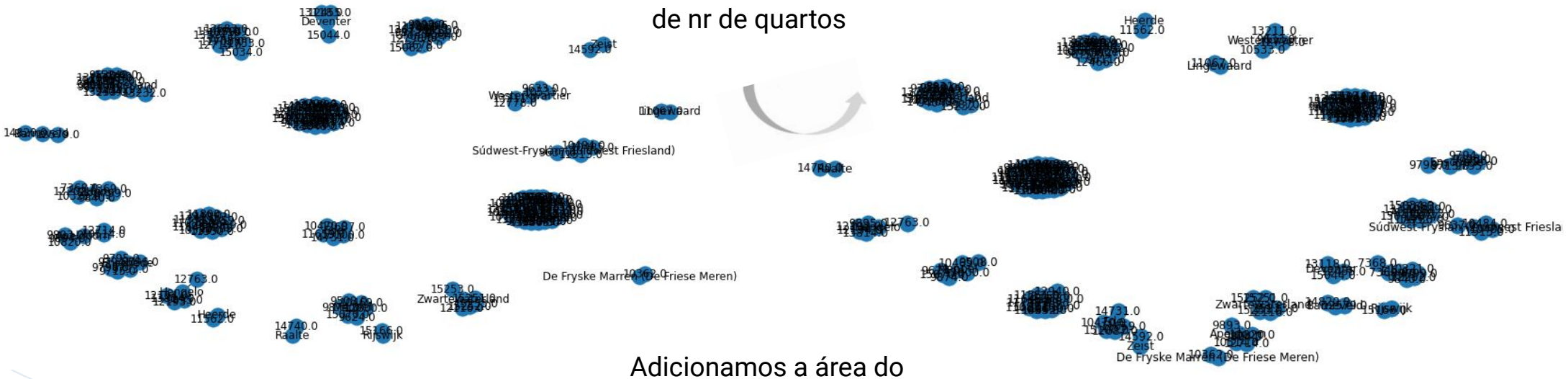
Amostra do último mês completo de dados: subscritores vs projetos



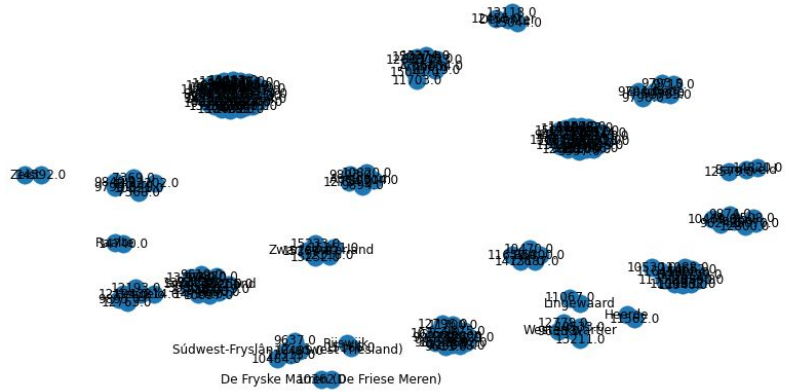
SNA | Users vs Income

Amostra com o Dataset com subscritores a todos os projetos vs rendimento

Adicionamos o máximo de nr de quartos



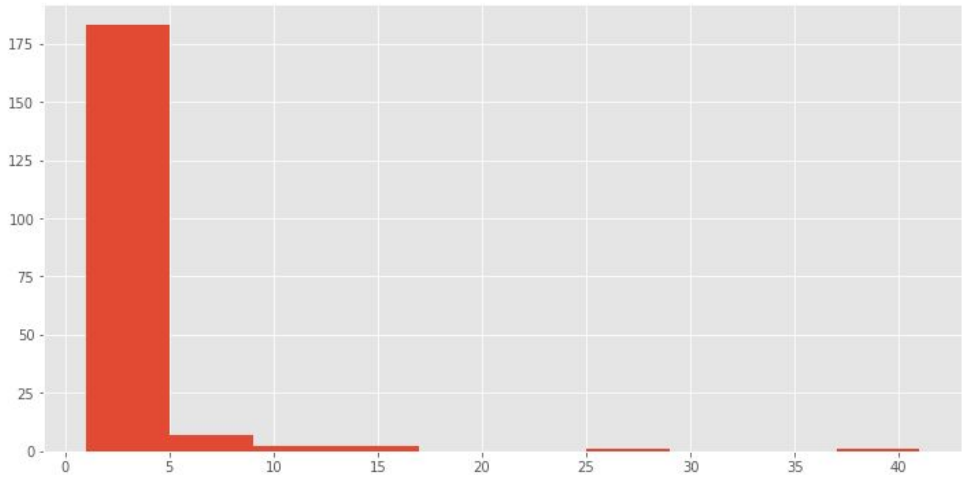
Adicionamos a área do lote



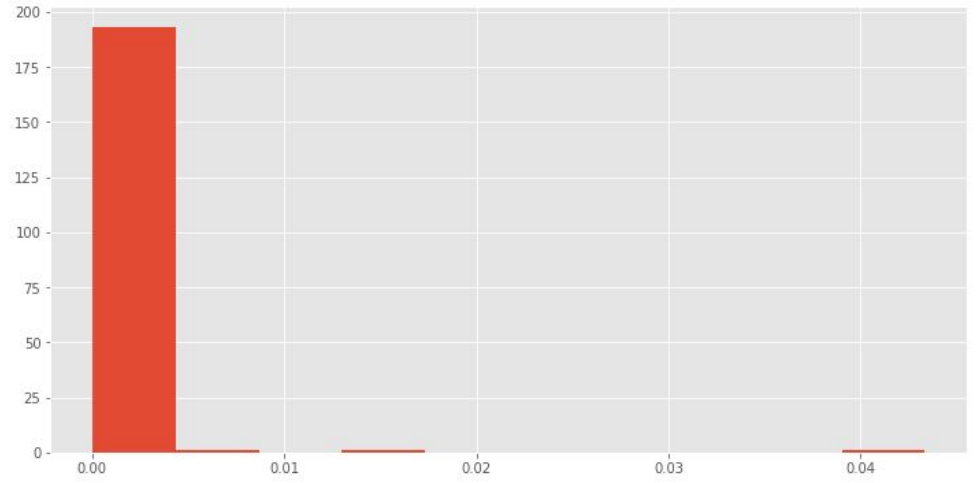
SNA | Projetos

Graph-theoretic measure

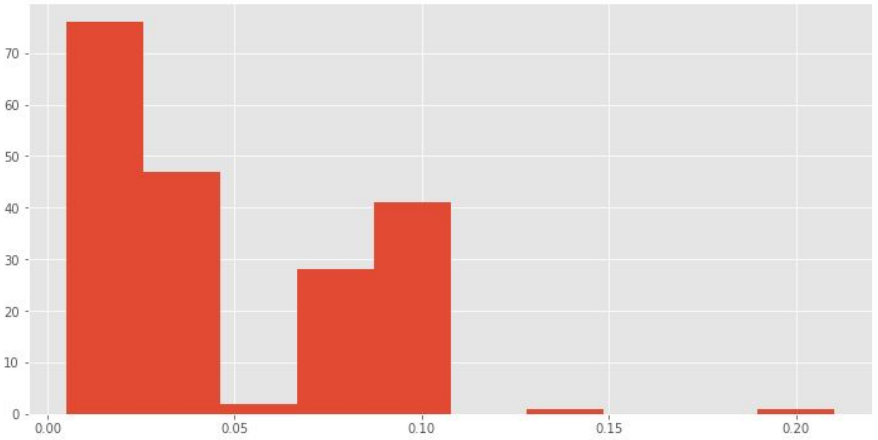
```
plt.hist([v for k, v in nx.degree(G)]);
```




```
plt.hist(nx.centrality.betweenness_centrality(G).values());
```

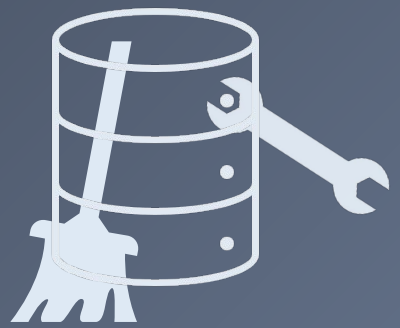



```
[108] plt.hist(nx.centrality.closeness_centrality(G).values());
```



SNA | Rede Bipartida

Amostra com o Dataset com subscritores a todos os projetos vs rendimento



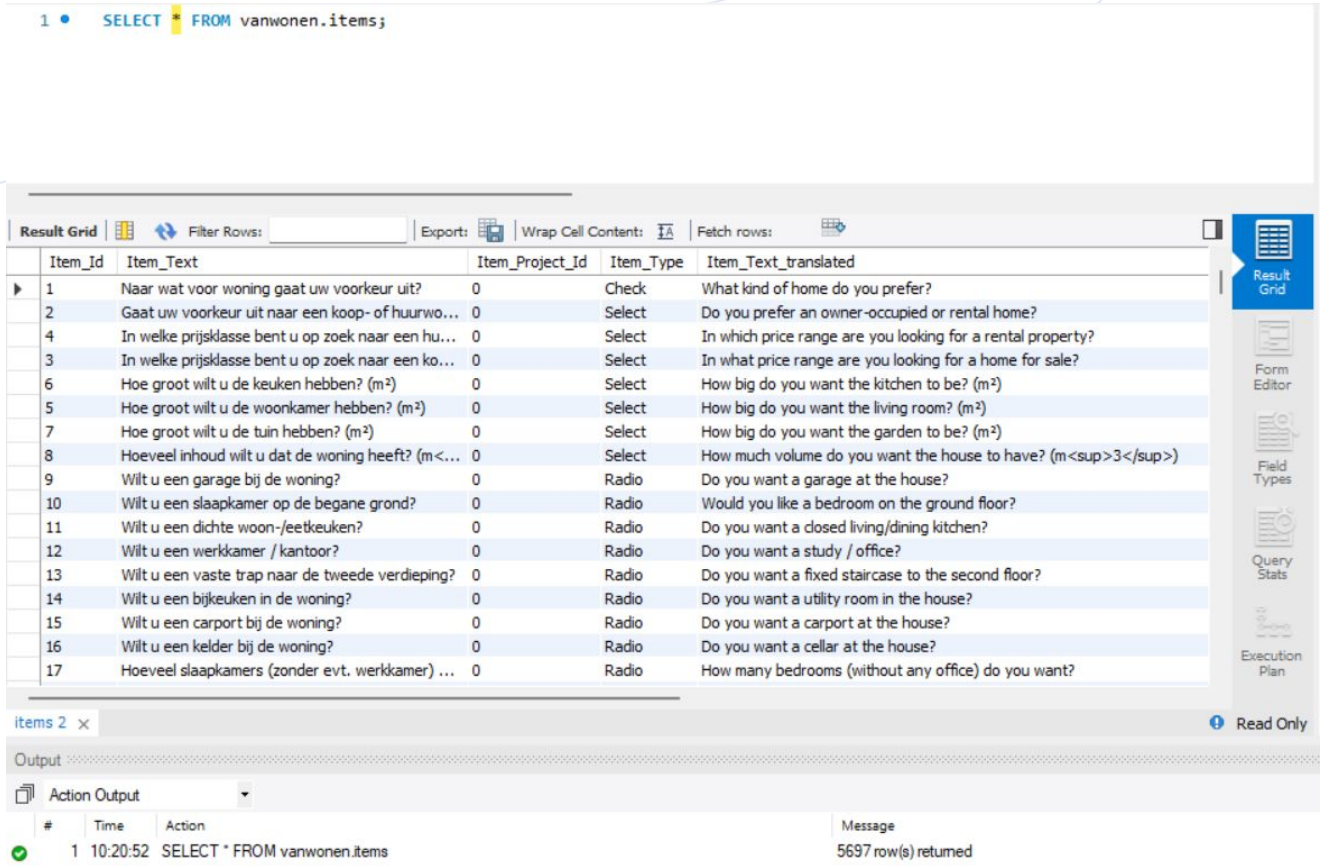
3. Data Preparation

Data Preparation | Inquéritos

- **O que queremos retirar destes questionários?**
  Features para resolver os nossos modelos de previsão e recomendação.

- **Quantas perguntas foram feitas nos inquéritos?**
  5697

- **Qual o processo para perceber o conteúdo dos questionários?**
  Numa primeira abordagem iremos fazer uma wordcloud às perguntas.



```
1 •  SELECT * FROM vanwonen.items;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

Item_Id	Item_Text	Item_Project_Id	Item_Type	Item_Text_translated
1	Naar wat voor woning gaat uw voorkeur uit?	0	Check	What kind of home do you prefer?
2	Gaat uw voorkeur uit naar een koop- of huurwo...	0	Select	Do you prefer an owner-occupied or rental home?
4	In welke prijsklasse bent u op zoek naar een hu...	0	Select	In which price range are you looking for a rental property?
3	In welke prijsklasse bent u op zoek naar een ko...	0	Select	In what price range are you looking for a home for sale?
6	Hoe groot wilt u de keuken hebben? (m²)	0	Select	How big do you want the kitchen to be? (m²)
5	Hoe groot wilt u de woonkamer hebben? (m²)	0	Select	How big do you want the living room? (m²)
7	Hoe groot wilt u de tuin hebben? (m²)	0	Select	How big do you want the garden to be? (m²)
8	Hoeveel inhoud wilt u dat de woning heeft? (m<...	0	Select	How much volume do you want the house to have? (m³)
9	Wilt u een garage bij de woning?	0	Radio	Do you want a garage at the house?
10	Wilt u een slaapkamer op de begane grond?	0	Radio	Would you like a bedroom on the ground floor?
11	Wilt u een dichte woon-/eetkeuken?	0	Radio	Do you want a closed living/dining kitchen?
12	Wilt u een werkkamer / kantoor?	0	Radio	Do you want a study / office?
13	Wilt u een vaste trap naar de tweede verdieping?	0	Radio	Do you want a fixed staircase to the second floor?
14	Wilt u een bijkeuken in de woning?	0	Radio	Do you want a utility room in the house?
15	Wilt u een carport bij de woning?	0	Radio	Do you want a carport at the house?
16	Wilt u een kelder bij de woning?	0	Radio	Do you want a cellar at the house?
17	Hoeveel slaapkamers (zonder evt. werkkamer) ...	0	Radio	How many bedrooms (without any office) do you want?

Result Grid | Form Editor | Field Types | Query Stats | Execution Plan

items 2 | Read Only

Output

Action Output

#	Time	Action	Message
1	10:20:52	SELECT * FROM vanwonen.items	5697 row(s) returned

Data Preparation | Inquéritos - Wordcloud

```
#Testar a BD
cur.execute("SELECT Item_Text_translated FROM vanwonen.items")
record = cur.fetchall()
record

#Converter a lista numa string
my_lst_str = ''.join(map(str, record))
print(my_lst_str)
```

```
#remover stop-words
text_tokens = word_tokenize(my_lst_str)

tokens_without_sw = [word for word in text_tokens if not word in stopwords.words()]

print(tokens_without_sw)
```

```
['(', "'What", 'kind', 'home', 'prefer', '?', "'", ',', ')', '(', "'Do", 'prefer', 'owne
```

```
[ ]
```

```
#import necessary libraries

#for plotting images & adjusting colors
import matplotlib.pyplot as plt
import matplotlib as mpl
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
from PIL import Image
# join the list and lowercase all the words
text = ' '.join(tokens_without_sw).lower()

#create the wordcloud object
wordcloud = WordCloud(stopwords = STOPWORDS,
                      collocations=True).generate(text)

#plot the wordcloud object
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

Esta wordcloud teve como objetivo entender o tipo de perguntas que foram feitas aos users nos questionários da Vanwonen. Infelizmente não conseguimos extrair informação relevante da wordcloud, por isso optamos por fazer um brainstorm, onde discutimos que dados seriam interessantes ter sobre os users para responder aos problemas que queremos resolver. E a partir dai, fazer queries à base de dados para verificar se essa informação existe (foi questionada ao user).

Data Preparation | Inquéritos - Wordcloud

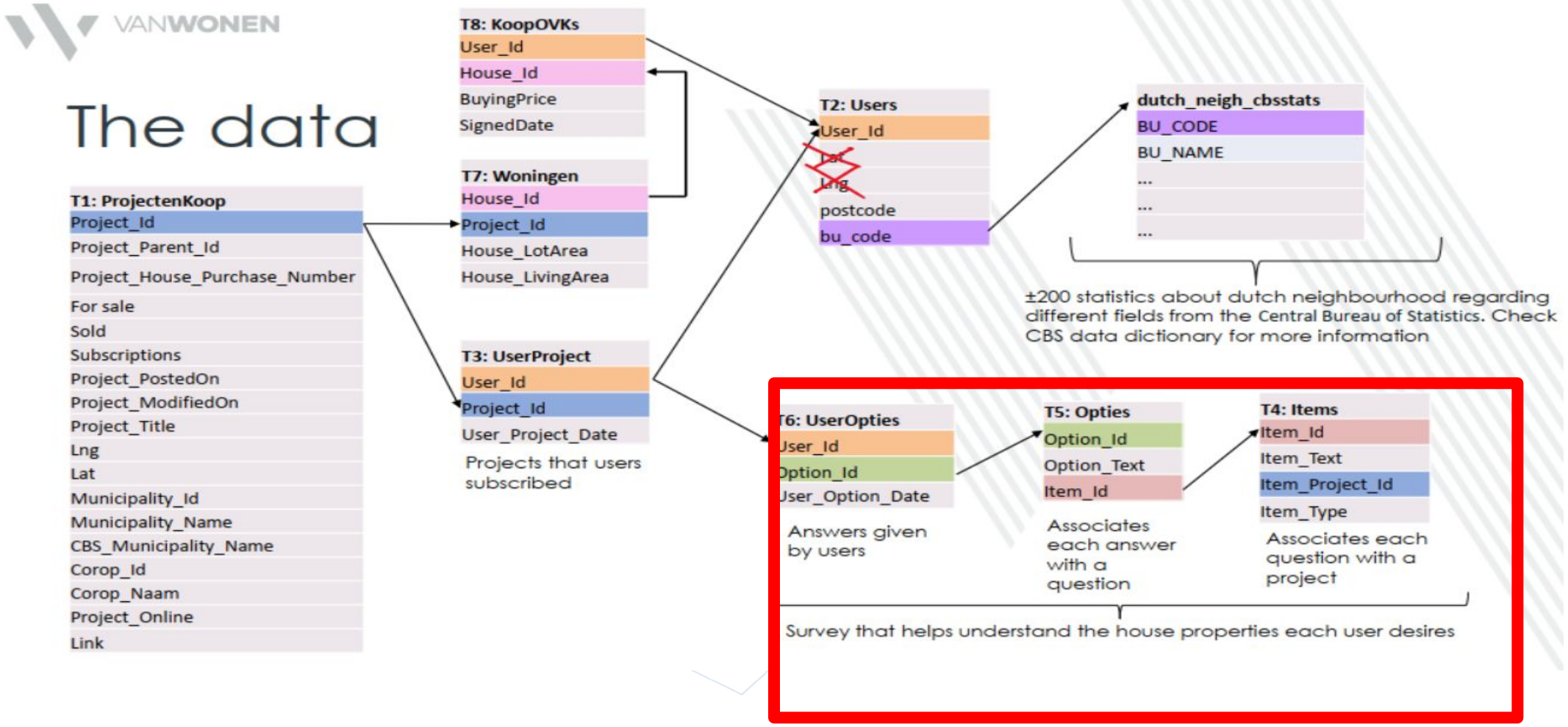
Data Preparation | Inquéritos - Brainstorm

- **Qual o processo para perceber o conteúdo dos questionários?**
  Brainstorm para descobrir quais as variáveis interessantes para resolver os nossos problemas de previsão e recomendação.

- **User features**
  - Idade
  - Estado Civil
  - Tem filhos?
  - Income
  - Deseja lugar de garagem?
  - Quanto está disposto a pagar por uma casa?
  - Quantos quartos deseja?
  - Qual a área (da habitação) desejada?

Data Preparation | Inquéritos

- **Juntar as tabelas com as respostas dos user às questões de um determinado projeto**



Data Preparation | Inquéritos

- **Juntar as tabelas com as respostas dos user às questões de um determinado projeto**

	Item_Id	Item_Text_translated	Item_Project_Id	Option_Id	Option_Text_translated	User_Id
0	1.0	What kind of home do you prefer?	0.0	3.0	Detached	238686.0
1	1.0	What kind of home do you prefer?	0.0	4.0	Free lot	238686.0
2	5.0	How big do you want the living room? (m²)	0.0	36.0	25 to 35	238686.0
3	6.0	How big do you want the kitchen to be? (m²)	0.0	44.0	from 25	238686.0
4	7.0	How big do you want the garden to be? (m²)	0.0	46.0	25 to 50	238686.0
...	...	...	...	...	...	...
647695	29128.0	Question 3 We are already thinking (in co...	12800.0	149201.0	I am interested after purchase	1324845.0
647696	29155.0	Question 7 For many of the terraced house...	12800.0	149317.0	My preference is not for a row house, back-to-...	1324845.0
647697	29086.0	How many people does your household consist of?	15044.0	149018.0	3	649621.0
647698	29088.0	What type of home are you looking for?	15044.0	149027.0	Semi-detached house	649621.0
647699	29089.0	Which price category do you prefer?	15044.0	149030.0	€300,000-400,000	649621.0

647700 rows × 6 columns

Data Preparation | Inquéritos

Dataset com todas as respostas dadas pelos subscritores a todos os projetos

```
mydb = mysql.connector.connect(**config)

query = "SELECT  items.Item_Id,
items.Item_Text_translated,items.Item_Project_Id,
opties.Option_Id, opties.Option_Text_translated,
useropties.User_Id FROM vanwonen.useropties left join
opties on useropties.Option_Id=opties.Option_Id left
join items on items.Item_Id=opties.Item_Id"

df = pd.read_sql(query, con = mydb)
df
```

	Item_Id	Item_Text_translated	Item_Project_Id	Option_Id	Option_Text_translated	User_Id
0	1.0	What kind of home do you prefer?	0.0	3.0	Detached	238686.0
1	1.0	What kind of home do you prefer?	0.0	4.0	Free lot	238686.0
2	5.0	How big do you want the living room? (m²)	0.0	36.0	25 to 35	238686.0
3	6.0	How big do you want the kitchen to be? (m²)	0.0	44.0	from 25	238686.0
4	7.0	How big do you want the garden to be? (m²)	0.0	46.0	25 to 50	238686.0
...	...	...	...	...	...	...
647695	29128.0	Question 3 We are already thinking (in co...	12800.0	149201.0	I am interested after purchase	1324845.0
647696	29155.0	Question 7 For many of the terraced house...	12800.0	149317.0	My preference is not for a row house, back-to-...	1324845.0
647697	29086.0	How many people does your household consist of?	15044.0	149018.0	3	649621.0
647698	29088.0	What type of home are you looking for?	15044.0	149027.0	Semi-detached house	649621.0
647699	29089.0	Which price category do you prefer?	15044.0	149030.0	€300,000-400,000	649621.0

647700 rows × 6 columns

Este dataset foi retirado da base de dados fornecida pela Vanwonen através de queries que fazem a ligação das respostas dadas pelos users a cada pergunta.
Cada pergunta está associada a um projeto, sendo que a mesma pergunta pode ter sido feita em diferentes projetos. O trabalho que se segue, com o objetivo de recolher informação sobre o user, envolveu tratamento de texto para conseguir identificar perguntas similares mas com IDs diferentes e agrupá-las num único dataset.

Data Preparation | Inquéritos

Exemplo do pipeline criado para extrair a variável “**Max_value**”:

1) Criar uma variável que extraia os números das respostas dadas pelos users.
2) Quanto custa a casa mais barata? 10.000Eur
3) Remover todas as linhas cujo número seja inferior ao valor da casa mais barata.
4) Observar as perguntas distintas que temos e remover as perguntas que não interessam.
5) Criar variavel Max_value.
6) Criar DataFrame para guardar as variaveis (User_Id, Project_Id, Max_value)

Data Preparation | Inquéritos

- Exemplo do pipeline criado para extrair a variável "**Max_value**":
  1) Criar uma variável que extraia os números das respostas dadas pelos users.

	Item_Id	Item_Text_translated	Item_Project_Id	Option_Id	Option_Text_translated	User_Id	User_Option_Date	Extract_number	Max_value
4	7.0	How big do you want the garden to be? (m²)	0.0	46.0	25 to 50	238686.0	2017-06-13 16:11:55	[25, 50]	50
5	8.0	How much volume do you want the house to have?...	0.0	52.0	400 to 550	238686.0	2017-06-13 16:11:55	[400, 550]	550
18	21.0	How do you want the finishing level of the house?	0.0	97.0	Basic (kitchen provisional € 3000 bathroom € 4...	238686.0	2017-06-13 16:11:55	[3000, 4000]	4000
45	21.0	How do you want the finishing level of the house?	0.0	98.0	Normal (supply post kitchen € 7500 bathroom € ...	173103.0	2017-06-13 20:51:21	[7500, 10000]	10000
46	22.0	Which living style appeals to you the most?	0.0	102.0	The 30's	173103.0	2017-06-13 20:51:21	[30]	30
...	...	...	...	...	...	...	...	...	...
647654	29089.0	Which price category do you prefer?	15044.0	149030.0	€300000-400000	389767.0	2022-05-31 15:49:24	[300000, 400000]	400000
647657	29089.0	Which price category do you prefer?	15044.0	149030.0	€300000-400000	389827.0	2022-05-31 18:31:25	[300000, 400000]	400000
647660	29089.0	Which price category do you prefer?	15044.0	149030.0	€300000-400000	389956.0	2022-05-31 22:46:14	[300000, 400000]	400000
647688	29089.0	Which price category do you prefer?	15044.0	149030.0	€300000-400000	393138.0	2022-06-07 17:05:38	[300000, 400000]	400000
647699	29089.0	Which price category do you prefer?	15044.0	149030.0	€300000-400000	649621.0	2022-06-09 18:56:19	[300000, 400000]	400000

86413 rows × 9 columns

Data Preparation | Inquéritos

- Exemplo do pipeline criado para extrair a variável “**Max_value**”:
  2) Quanto custa a casa mais barata? 10.000Eur
  3) Remover todas as linhas cuja resposta seja inferior à casa mais barata

```
1 •  SELECT * FROM vanwonen.koopovks;
```

Result Grid | Filter Rows: | Expo

User_Id	House_Id	BuyingPrice	SignedDate
867178	108711	12500	2021-07-02
340721	108712	12500	2021-06-29
431917	108710	15000	2021-06-28
862036	108713	15000	2021-07-02
468602	91804	17500	2021-12-14
468602	91805	17500	2021-12-14
404527	91806	17500	2020-09-10
122434	91797	19500	2020-10-17
380140	91800	19500	2020-09-28
350857	91801	19500	2020-09-21

Perguntas distintas resultantes da remoção de respostas (sem número) e inferiores ao valor da casa mais barata:

'In which price range are you looking for a home?'
'What is your annual income?'
'In what price range are you looking for a home for sale?'
'How do you want the finishing level of the house?'

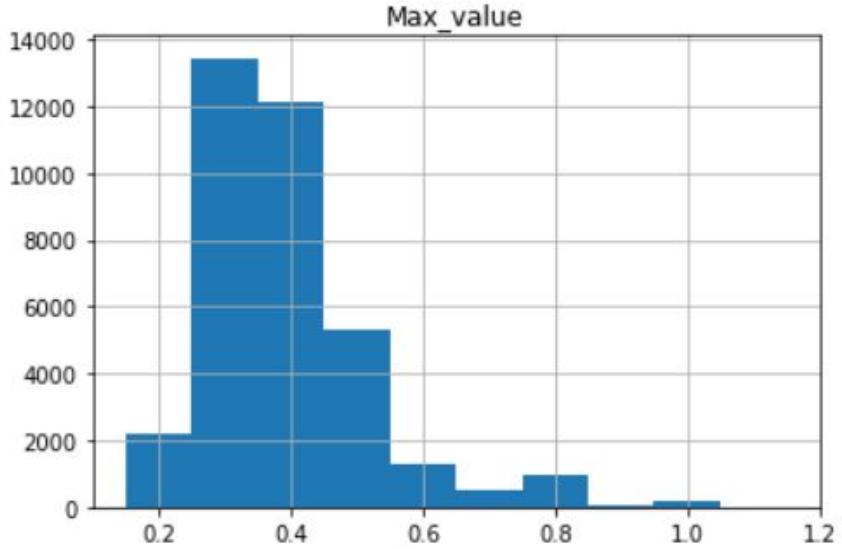
Data Preparation | Inquéritos

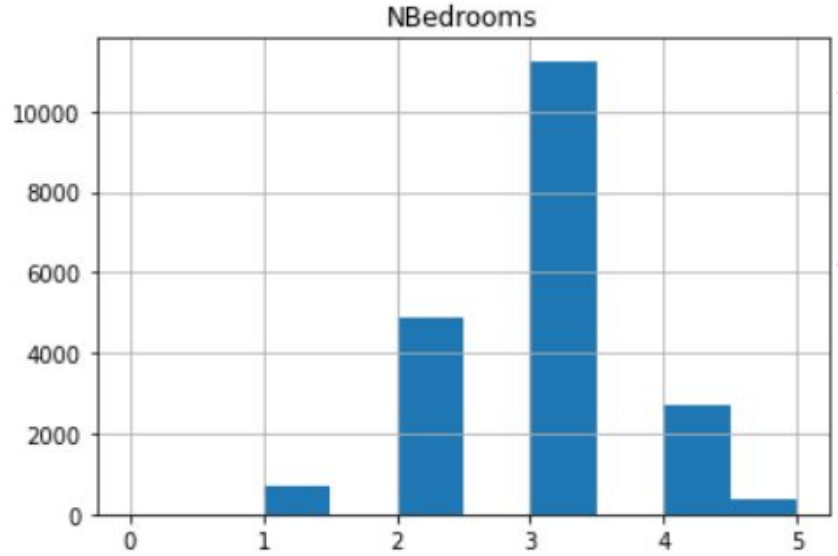
- Exemplo do pipeline criado para extrair a variável “**Max_value**”:
    4) Observar as perguntas distintas que temos e remover as perguntas que não interessam.
    5) Criar o Dataframe (User_Id, Item_Project_Id, User_Option_Date,Max_value)

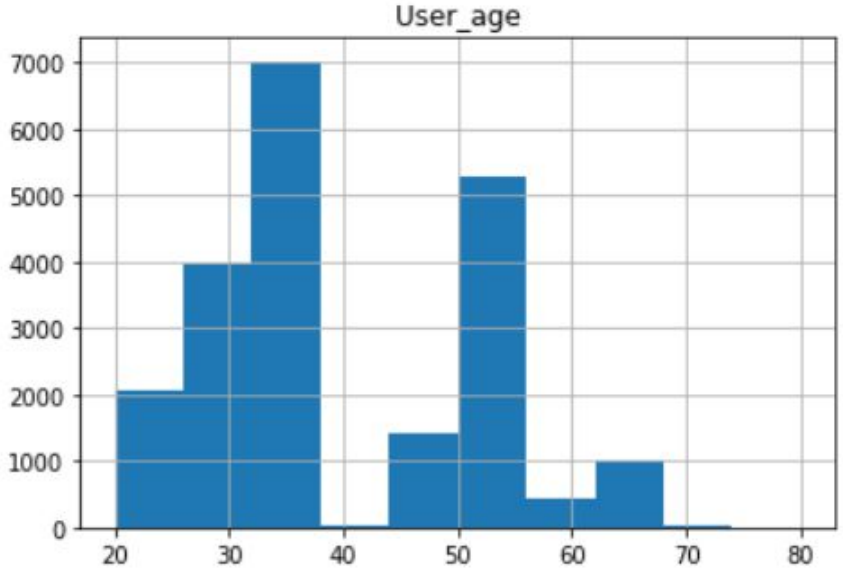
	User_Id	Item_Project_Id	User_Option_Date	Max_value
83	355677	9675	2017-06-14 09:45:32	250000
101	355872	0	2017-06-14 10:08:47	350000
130	335361	0	2017-06-14 10:38:51	200000
189	355956	9675	2017-06-14 14:40:37	250000
239	172382	0	2017-06-14 18:26:52	400000
...	...	...	...	...
647654	1389767	15044	2022-05-31 15:49:24	400000
647657	1389827	15044	2022-05-31 18:31:25	400000
647660	1389956	15044	2022-05-31 22:46:14	400000
647688	1393138	15044	2022-06-07 17:05:38	400000
647699	649621	15044	2022-06-09 18:56:19	400000

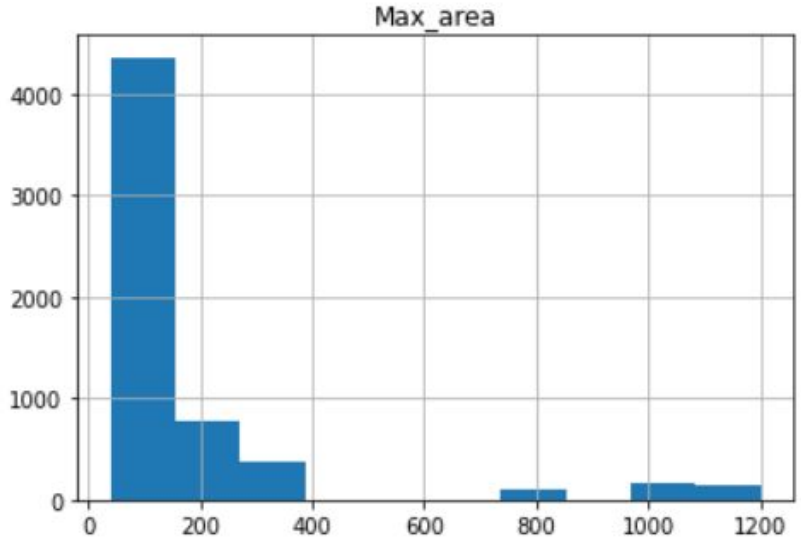
36013 rows × 4 columns

Data Preparation | Inquéritos



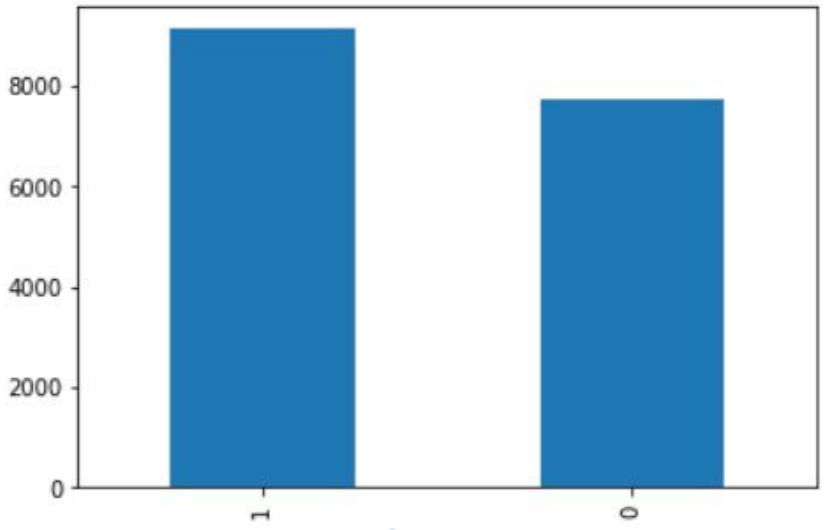






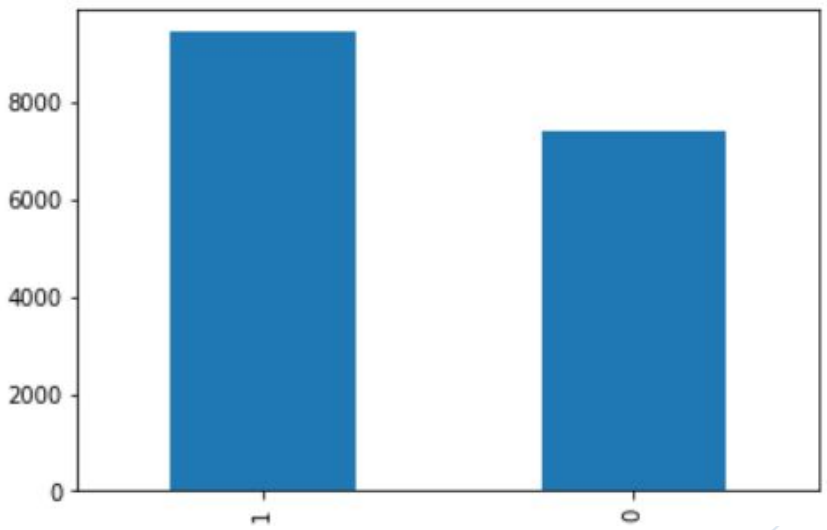
Data Preparation | Inquéritos

Parent



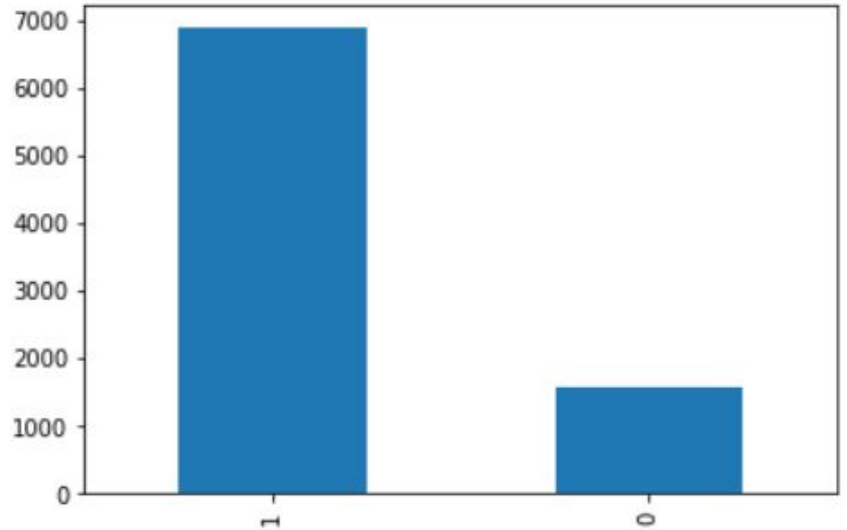
1: Tem filhos
0: Não tem filhos

Marital_Status



1: Casado
0: Solteiro

User_garage



1: Quer lugar de garagem
0: Não quer lugar de garagem

Data Preparation | Inquéritos

NBedrooms – Número de quartos desejado pelo user

```
#Extract numbers

df['Extract_number'] = df.Option_Text_translated.str.findall(r'(\d+(?:\.\d+)?)')

#Extract numbers not empty

df=df[df['Extract_number'].apply(lambda x: len(x)) > 0]

df['NBedrooms']=df['Extract_number'].apply(lambda x: x[-1])

#Convert to int

df['NBedrooms'] = df['NBedrooms'].astype('int')

#Remove unwanted answer

df = df[df["Item_Text_translated"].str.contains("room") == True]

df = df[df["Item_Text_translated"].str.contains("Options") == False]

df = df[df["Item_Text_translated"].str.contains("large") == False]

df = df[df["Item_Text_translated"].str.contains("m²") == False]
```

```
print(df['Item_Text_translated'].unique())

['How many bedrooms (without any office) do you want?'

 'How many bedrooms do you want?' 'Number of rooms on the 1st floor:'

 'Number of rooms in the attic:' 'How many bedrooms do you prefer?'

 'How many bedrooms do you want in your future home?'

 'How many bedrooms do you want on the ground floor?'

 'How many (bed)rooms do you want at least next to your living room, kitchen and bathroom?

On the 1st floor:'

 'How many bedrooms do you want upstairs?'

 'How many rooms do you want? This includes the living, sleeping and study or work rooms.']
```

Data Preparation | Inquéritos

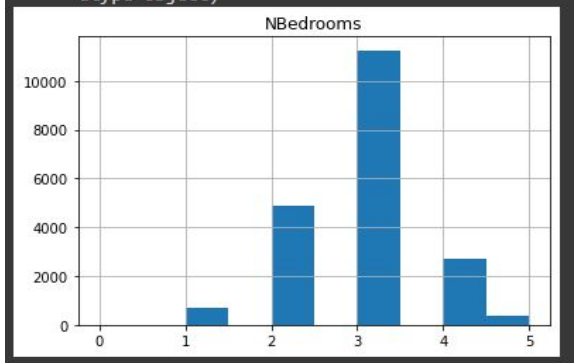
NBedrooms – Número de quartos desejado pelo user

	User_Id	Item_Project_Id	NBedrooms
14	238686	0	3
41	173103	0	3
73	355677	9675	3
111	355872	0	4
155	355935	0	3
...	...	...	...
631637	617131	11703	4
631657	621155	11703	4
631661	618377	11703	4
631673	566415	11703	4
631685	626170	11703	4

19947 rows × 3 columns

```
#User Max value graphs
df.hist(column='NBedrooms')
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f4e162b0850>]],
      dtype=object)
```




```
boxplot = df.boxplot(column=['NBedrooms'])
```

Data Preparation | Inquéritos

Porto Business School / University of Porto

Max_area – Área desejada pelo user

```
#Extract numbers

df['Extract_number'] = df.Option_Text_translated.str.findall(r'(\d+(?:\.\d+)?)')

#Extract numbers not empty

df=df[df['Extract_number'].apply(lambda x: len(x)) > 0]

df['Max_area']=df['Extract_number'].apply(lambda x: x[-1])

#Convert Max_value to int

df['Max_area'] = df['Max_area'].astype('int')

#Remove Max_area lesser than 1500

df=df[df.Max_area < 1500]

df=df[df.Max_area > 35]

print(df['Item_Text_translated'].unique())

#Remove unwanted answer

df = df[df["Item_Text_translated"].str.contains("garden") == False]

df = df[df["Item_Text_translated"].str.contains("volume") == False]
```

▪ . .

. . .

▪ . .

. . .

```
#Remove unwanted answer

df = df[df["Item_Text_translated"].str.contains("age") == False]

df = df[df["Item_Text_translated"].str.contains("rental") == False]

df = df[df["Item_Text_translated"].str.contains("outdoor") == False]

df = df[df["Item_Text_translated"].str.contains("room") == False]

df = df[df["Item_Text_translated"].str.contains("depth") == False]

df = df[df["Item_Text_translated"].str.contains("number") == False]

df = df[df["Item_Text_translated"].str.contains("independently") == False]

df = df[df["Item_Text_translated"].str.contains("number") == False]

df = df[df["Item_Text_translated"].str.contains("price") == False]

df = df[df["Item_Text_translated"].str.contains("energy") == False]

print(df['Item_Text_translated'].unique())
```

['How much m2 of living space do you want your future home to have?' 'How big do you want to live (living space)?' 'How big can your plot be?'

'How is your household composed?' 'What type of home are you looking for?'

'Do you have specific wishes that your apartment must meet?"Question 5b
 How big do you want to live?'

'What size/usable surface should your new home or apartment be?' 'What size / usable surface should your new home be?']

Data Preparation | Inquéritos

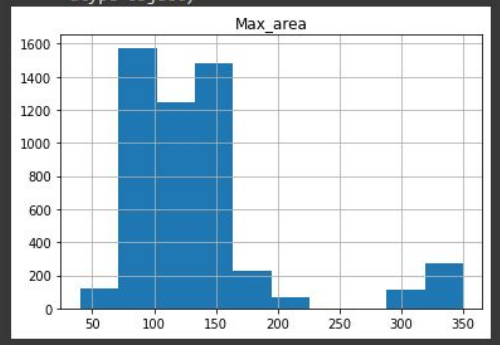
Max_area – Área desejada pelo user

	User_Id	Item_Project_Id	Max_area
11635	51226	9592	120
11656	224019	9592	60
12032	58159	9592	120
12256	385970	9592	120
12685	387352	9592	120
...	...	...	...
646434	1344688	12800	110
646491	1332608	12800	110
646718	210800	12800	110
646786	1356151	12800	110
646814	1356923	12800	110

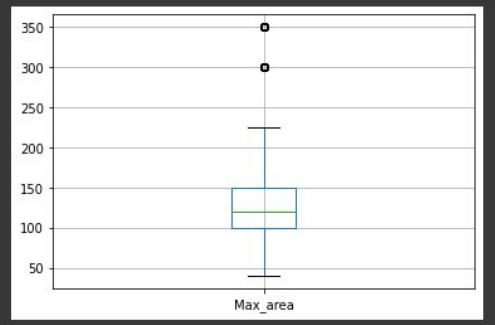
5100 rows × 3 columns

```
#User Max value graphs
df.hist(column='Max_area')
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f7d82a89f40>]],
      dtype=object)
```




```
[63] boxplot = df.boxplot(column=['Max_area'])
```



Data Preparation | Inquéritos

Porto Business School / University of Porto

User_age – Idade do user

```
#Extract numbers

df['Extract_number'] = df.Option_Text_translated. str.findall(r'(\d+(?:\.\d+)?)')

#Extract numbers not empty

df=df[df['Extract_number'].apply(lambda x: len(x)) > 0]

df['User_age_max']=df['Extract_number'].apply(lambda x: x[-1])

df['User_age_min']=df['Extract_number'].apply(lambda x: x[0])

#Remove numbers not in [18,100]

df['User_age_max'] = df['User_age_max'].astype('int')

df['User_age_min'] = df['User_age_min'].astype('int')

df=df[df.User_age_min > 18]

df=df[df.User_age_min < 100]

df=df[df.User_age_max > 18]

df=df[df.User_age_max < 100]
```

```
#Remove unwanted answer

df = df[df["Item_Text_translated"].str.contains("age") == True]

print(df['Item_Text_translated'].unique())

['What is your age?' 'What age category are you in?'

 '<b>1. Which of the following age categories do you belong to? </b>'

 'What is your age:' "What's your age?"

 'Question 14 <br> What is your age?']

#Get middle number

df['User_age'] = df['User_age_max']/2+df['User_age_min']/2

df['User_age'] = df['User_age'].astype('int')
```

Data Preparation | Inquéritos

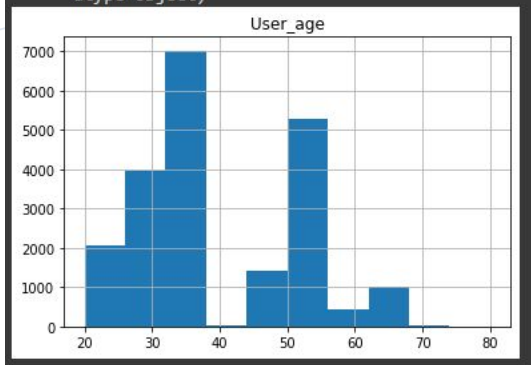
User_age – Idade do user

	User_Id	Item_Project_Id	User_age
22	238686	0	55
50	173103	0	37
78	355677	9675	58
119	355872	0	30
132	335361	0	30
...	...	...	...
631459	381057	12851	55
631467	1201160	12851	55
631521	396307	12851	55
631525	1392089	12851	55
631530	1391082	12851	55

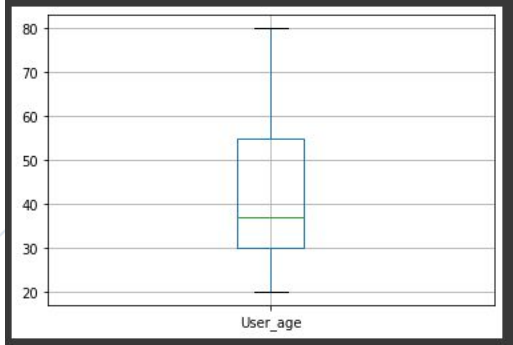
21264 rows × 3 columns

```
df.hist(column='User_age')
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f560279de50>]],
      dtype=object)
```




```
boxplot = df.boxplot(column=['User_age'])
```



Data Preparation | Inquéritos

Porto Business School / University of Porto

Dataset com todas as respostas dadas pelos subscritores a todos os projetos – Estado civil do user + O user tem filhos?

```
#Create variable for marital_status (1=married, 0=single)

df.loc[df['Option_Text_translated'] == 'Single', 'Marital_status'] = 0
df.loc[df['Option_Text_translated'] == 'Living with children', 'Marital_status'] = 0
df.loc[df['Option_Text_translated'] == 'Single with children', 'Marital_status'] = 0
df.loc[df['Option_Text_translated'] == 'Single household', 'Marital_status'] = 0
df.loc[df['Option_Text_translated'] == 'single parent', 'Marital_status'] = 0
df.loc[df['Option_Text_translated'] == 'Only with children', 'Marital_status'] = 0
df.loc[df['Option_Text_translated'] == 'Single with child(ren)', 'Marital_status'] = 0
df.loc[df['Option_Text_translated'] == 'Only with children', 'Marital_status'] = 0
df.loc[df['Option_Text_translated'] == 'Single with child(ren)', 'Marital_status'] = 0
df.loc[df['Option_Text_translated'] == 'alone with children', 'Marital_status'] = 0
df.loc[df['Option_Text_translated'] == 'Single-parent family with children (living at home) aged 12 years and older', 'Marital_status'] = 0
df.loc[df['Option_Text_translated'] == 'Single-parent family with children (living at home) aged 6 to 12 years', 'Marital_status'] = 0
df.loc[df['Option_Text_translated'] == 'Single-parent family with (living at home) children up to 6 years old', 'Marital_status'] = 0
df.loc[df['Option_Text_translated'] == 'Living together without children', 'Marital_status'] = 1
df.loc[df['Option_Text_translated'] == 'Couple without children', 'Marital_status'] = 1
df.loc[df['Option_Text_translated'] == 'Couple with children', 'Marital_status'] = 1
df.loc[df['Option_Text_translated'] == 'Living together', 'Marital_status'] = 1
df.loc[df['Option_Text_translated'] == 'live together', 'Marital_status'] = 1
df.loc[df['Option_Text_translated'] == 'Couple with (living at home) children up to 6 years old', 'Marital_status'] = 1
df.loc[df['Option_Text_translated'] == 'Couple with (living at home) children aged 6 to 12 years', 'Marital_status'] = 1
df.loc[df['Option_Text_translated'] == 'Couple without (living at home) children', 'Marital_status'] = 1
df.loc[df['Option_Text_translated'] == 'Couple with (living at home) children aged 12 years and older', 'Marital_status'] = 1
```

```
mydb = mysql.connector.connect (**config)

query = "SELECT  items.Item_Id,
items.Item_Text_translated,items.Item_Project_Id, opties.Option_Id,
opties.Option_Text_translated, useropties.User_Id FROM
vanwonen.useropties left join opties on
useropties.Option_Id=opties.Option_Id left join items on
items.Item_Id=opties.Item_Id"
HAVING items.Item_Text_translated LIKE '%fam%'"
```

Data Preparation | Inquéritos

Marital_status + Parent – Estado civil do user + O user tem filhos?

```
#Create variable for parents (1=children, 0=no children)

df.loc[df['Option_Text_translated'] == 'Single', 'Parent'] = 0
df.loc[df['Option_Text_translated'] == 'Single household', 'Parent'] = 0
df.loc[df['Option_Text_translated'] == 'Alone', 'Parent'] = 0
df.loc[df['Option_Text_translated'] == 'Living together', 'Parent'] = 0
df.loc[df['Option_Text_translated'] == 'live together', 'Parent'] = 0
df.loc[df['Option_Text_translated'] == 'Couple without (living at home) children', 'Parent'] = 0
df.loc[df['Option_Text_translated'] == 'Living with children', 'Parent'] = 1
df.loc[df['Option_Text_translated'] == 'Single with children', 'Parent'] = 1
df.loc[df['Option_Text_translated'] == 'single parent', 'Parent'] = 1
df.loc[df['Option_Text_translated'] == 'Only with children', 'Parent'] = 1
df.loc[df['Option_Text_translated'] == 'Single with child(ren)', 'Parent'] = 1
df.loc[df['Option_Text_translated'] == 'alone with children', 'Parent'] = 1
df.loc[df['Option_Text_translated'] == 'Single-parent family with children (living at home) aged 12 years and older', 'Parent'] = 1
df.loc[df['Option_Text_translated'] == 'Single-parent family with children (living at home) aged 6 to 12 years', 'Parent'] = 1
df.loc[df['Option_Text_translated'] == 'Single-parent family with (living at home) children up to 6 years old', 'Parent'] = 1
df.loc[df['Option_Text_translated'] == 'Living together without children', 'Parent'] = 1
df.loc[df['Option_Text_translated'] == 'Couple without children', 'Parent'] = 1
df.loc[df['Option_Text_translated'] == 'Couple with children', 'Parent'] = 1
df.loc[df['Option_Text_translated'] == 'Family', 'Parent'] = 1
df.loc[df['Option_Text_translated'] == 'Couple with (living at home) children up to 6 years old', 'Parent'] = 1
df.loc[df['Option_Text_translated'] == 'Couple with (living at home) children aged 6 to 12 years', 'Parent'] = 1
df.loc[df['Option_Text_translated'] == 'Couple with (living at home) children aged 12 years and older', 'Parent'] = 1
```

Tanto na identificação do estado civil do user como para saber se o user tem filhos ou não, surgiram casos que suscitaram dúvida. Como por exemplo “Living together” não diz claramente se o user é pai ou não, mas através da análise das opções de resposta a esta pergunta conseguimos perceber que o user não é pai, pois havia a opção de resposta “Couple with kids”. Este exercício foi feito para todos os casos onde a resposta não era clara.

Data Preparation | Inquéritos

Marital_status + Parent – Estado civil do user + O user tem filhos?

	User_Id	Item_Project_Id	Marital_status	Parent
0	238686	0	1	1
1	173103	0	0	0
3	355872	0	0	1
4	279765	0	0	0
5	335361	0	1	1
...	...	...	...	...
23128	1369902	11868	0	0
23129	1356427	11868	0	0
23130	999441	11868	0	0
23131	1186515	12800	0	1
23132	1193623	12800	0	1

16856 rows × 4 columns

Data Preparation | Inquéritos

User_garage – O user pretende lugar de garagem?

```
mydb = mysql.connector.connect(**config)

query = "SELECT  items.Item_Id, items.Item_Text_translated,items.Item_Project_Id, opties.Option_Id,
opties.Option_Text_translated, useropties.User_Id FROM vanwonen.useropties left join opties on
useropties.Option_Id=opties.Option_Id left join items on items.Item_Id=opties.Item_Id HAVING
Item_Text_translated LIKE '%garage%'"

df = pd.read_sql(query, con = mydb)
df
print(df['Item_Text_translated'].unique())
['Do you want a garage at the house?' 'Garage/Storage:'
 'Do you experience the use of the Boterdiep parking garage as positive? And why?'
 'Do you use the Boterdiep parking garage?' 'Do you want a garage?'
 'Do you prefer a garage or rather a carport?'
 'Do you think it would be desirable that an option be available to realize a passage from the house to the
storage room/garage (this question relates to semi-detached houses and detached houses)?'
 'Would you like another addition? (such as a garage or carport)']
```

Data Preparation | Inquéritos

User_garage – O user pretende lugar de garagem?

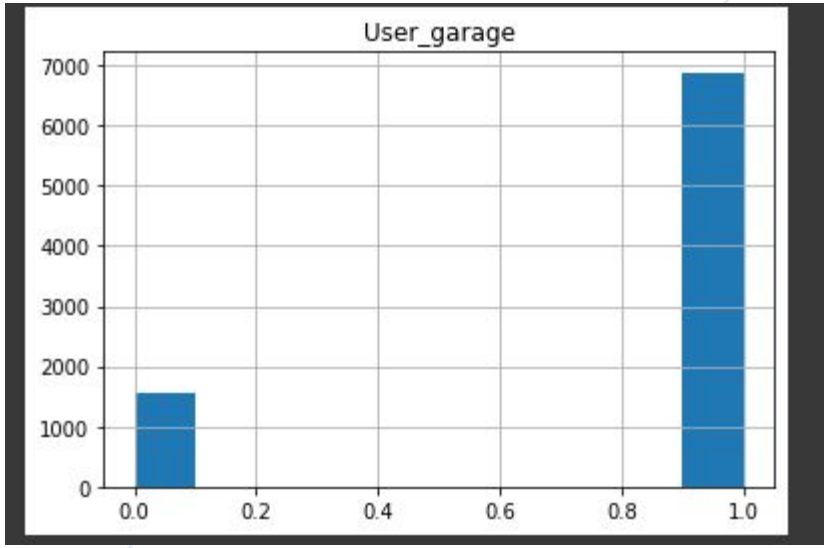
```
df = df[df["Item_Text_translated"].str.contains("Boterdiep") == False]
df = df[df["Item_Text_translated"].str.contains("semi-detached") == False]
df = df[df["Item_Text_translated"].str.contains("Garage/Storage") == False]

#Create variable for users desire for garage (1=want garage, 0=don't want garage)

df.loc[df['Option_Text_translated'] == 'Possibly', 'User_garage'] = 1
df.loc[df['Option_Text_translated'] == 'Yes', 'User_garage'] = 1
df.loc[df['Option_Text_translated'] == 'no', 'User_garage'] = 0
df.loc[df['Option_Text_translated'] == 'Garage', 'User_garage'] = 1
df.loc[df['Option_Text_translated'] == 'carport', 'User_garage'] = 0
df.loc[df['Option_Text_translated'] == 'Neither, my preference is for a larger driveway.', 'User_garage'] = 0
df.loc[df['Option_Text_translated'] == 'Addition of carport € 15,000 v.o.n.', 'User_garage'] = 0
df.loc[df['Option_Text_translated'] == 'Addition garage € 25,000 v.o.n.', 'User_garage'] = 1

#Final dataset
df=df[['User_Id', 'Item_Project_Id','User_garage']]
```

Data Preparation | Inquéritos

Dataset com todas as respostas dadas pelos subscritores a todos os projetos – O user pretende lugar de garagem?

	User_Id	Item_Project_Id	User_garage
0	238686	0	1
1	355872	0	1
2	355935	0	1
3	356001	0	1
4	172382	0	1
...	...	...	...
8602	1210026	12851	1
8603	1178258	12851	1
8604	1230303	12851	1
8605	976870	12851	1
8606	272015	12851	1

8454 rows × 3 columns



Data Preparation | Inquéritos

Dataset com todas as respostas dadas pelos subscritores a todos os projetos – Valor máximo que o user está disposto a gastar na casa

```
# filtering the rows where Option_Text_translated contains "0"

df =df[df['Option_Text_translated' ].str.contains ("0")]

#Extract numbers

df['Extract_number' ] = df.Option_Text_translated. str.findall(r'(\d+(?:\.\d+)?)' )

#Extract last numbers of the list

df['Max_value' ]=df['Extract_number' ].apply(lambda x: x[-1])

#Convert Max_value to int

df['Max_value' ] = df['Max_value' ].astype('int')

#Remove Max_value lesser than 10000eur

df=df[df.Max_value > 10000]

print(df['Item_Text_translated' ].unique())

['In which price range are you looking for a home?'

 'What is your annual income? (incl. holiday allowance and partner income)'

 'In what price range are you looking for a home for sale?'

 'How do you want the finishing level of the house?'

 'Which price category do you prefer?'

 'How do you want the finishing level of the house?' 'Which price category do you prefer?'

 'Which price category do you prefer/'  'What type of home are you looking for?'

 'In which price range should the house fall?' 'Sustainability measures require an extra investment,
which you earn back through lower monthly costs.Which of the options below applies to you?'

 'In which price category are you looking for a home?' 'In which price category are you looking
(several options possible)?' 'In which price range are you looking for a life-resistant home?'

 'Which price category is preferred?' 'In which price range are you looking for an apartment?'

 'In which price range* (more options)' 'Can you indicate in which price segment you are looking for a
home?' 'Which housing type do you prefer?' 'Can you indicate in which price segment you are looking
for a home? (Maximum 2 preferences)' 'Which house do you prefer?' '17. In which price category are you
looking for a house / what do you want to spend on a house?' 'Question 5c <br> In which price category
are you looking for a home?' 'Question 5c <BR> In which price category are you looking for a home?'

 'In which price range do you want to buy a home?' 'Would you like another addition? (such as a garage
or carport)']
```

Data Preparation | Inquéritos

Dataset com todas as respostas dadas pelos subscritores a todos os projetos – Valor máximo que o user está disposto a gastar na casa

```
#Remove unwanted answer

print(df['Item_Text_translated'].unique())

df = df[df["Item_Text_translated"].str.contains("income") == False]

df = df[df["Item_Text_translated"].str.contains("finishing") == False]

df = df[df["Item_Text_translated"].str.contains("investment") == False]

df = df[df["Item_Text_translated"].str.contains("garage") == False]

print(df['Item_Text_translated'].unique())

['In which price range are you looking for a home?'

 'In what price range are you looking for a home for sale?'

 'Which price category do you prefer?'

 'Which price category do you prefer/'

 'What type of home are you looking for?'

 'In which price range should the house fall?'

 'In which price category are you looking for a home?'
. . .
. . .
. . .

 'In which price category are you looking (several options possible)?'

 'In which price range are you looking for a life-resistant home?'

 'Which price category is preferred?'

 'In which price range are you looking for an apartment?'

 'In which price range* (more options)'

 'Can you indicate in which price segment you are looking for a home?'

 'Which housing type do you prefer?'

 'Can you indicate in which price segment you are looking for a home? (Maximum 2 preferences)'

 'Which house do you prefer?'

 '17. In which price category are you looking for a house / what do you want to spend on a house?'

 'Question 5c <br> In which price category are you looking for a home?'

 'Question 5c <BR> In which price category are you looking for a home?'

 'In which price range do you want to buy a home?']
```

Data Preparation | Inquéritos

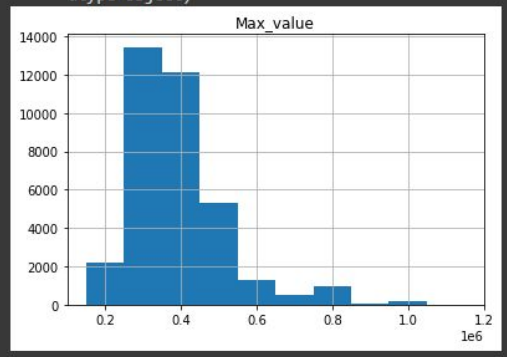
Max_value – Valor máximo que o user está disposto a pagar

	User_Id	Item_Project_Id	User_Option_Date	Max_value
83	355677	9675	2017-06-14 09:45:32	250000
101	355872	0	2017-06-14 10:08:47	350000
130	335361	0	2017-06-14 10:38:51	200000
189	355956	9675	2017-06-14 14:40:37	250000
239	172382	0	2017-06-14 18:26:52	400000
...	...	...	...	...
647654	1389767	15044	2022-05-31 15:49:24	400000
647657	1389827	15044	2022-05-31 18:31:25	400000
647660	1389956	15044	2022-05-31 22:46:14	400000
647688	1393138	15044	2022-06-07 17:05:38	400000
647699	649621	15044	2022-06-09 18:56:19	400000

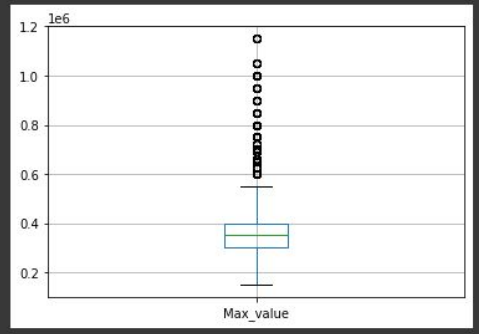
37427 rows × 4 columns

```
#User Max value graphs
df.hist(column='Max_value')
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f55258389d0>]],
      dtype=object)
```




```
[ ] boxplot = df.boxplot(column=['Max_value'])
```



Data Preparation | Inquéritos

Dataset com todas as respostas dadas pelos subscritores a todos os projetos – Income dos users

```
mydb = mysql.connector.connect(**config)

query = "SELECT  items.Item_Id,
items.Item_Text_translated,items.Item_Project_Id,
opties.Option_Id, opties.Option_Text_translated,
useropties.User_Id FROM vanwonen.useropties left join opties on
useropties.Option_Id=opties.Option_Id left join items on
items.Item_Id=opties.Item_Id HAVING items.Item_Text_translated
LIKE '%income%'"

df = pd.read_sql(query, con = mydb)
df
```

	Item_Id	Item_Text_translated	Item_Project_Id	Option_Id	Option_Text_translated	User_Id
0	3826.0	What is your annual income? (incl. holiday all...	9675.0	18032.0	1.5 times the average (approximately €50,000 /...	355677.0
1	3826.0	What is your annual income? (incl. holiday all...	9675.0	18030.0	Below average (less than €33,000 / year)	355956.0
2	3826.0	What is your annual income? (incl. holiday all...	9675.0	18032.0	1.5 times the average (approximately €50,000 /...	356238.0
3	3826.0	What is your annual income? (incl. holiday all...	9675.0	18031.0	Modal (approximately €33,000 / year)	356527.0
4	3826.0	What is your annual income? (incl. holiday all...	9675.0	18030.0	Below average (less than €33,000 / year)	356601.0
...	...	...	...	...	...	...
60	3826.0	What is your annual income? (incl. holiday all...	9675.0	18034.0	More than 2 times the average (more than €66,0...	401984.0
61	3826.0	What is your annual income? (incl. holiday all...	9675.0	18031.0	Modal (approximately €33,000 / year)	403760.0
62	3826.0	What is your annual income? (incl. holiday all...	9675.0	18032.0	1.5 times the average (approximately €50,000 /...	404180.0
63	3826.0	What is your annual income? (incl. holiday all...	9675.0	18033.0	2 times the average (approximately €66,000 / y...	405141.0
64	3826.0	What is your annual income? (incl. holiday all...	9675.0	18031.0	Modal (approximately €33,000 / year)	405194.0

65 rows × 6 columns

Acreditamos que a variável “Income” seria bastante interessante para prever o máximo que o utilizador está disposto a pagar por um apartamento (regressão) ou até mesmo para prever se o user quer lugar de garagem ou não (classificação). Mas infelizmente esta pergunta tem apenas 65 respostas, por isso decidimos ignorar esta feature.

Data Preparation | Tabelas

Tarefas gerais de preparação dos dados

Porto Business School / University of Porto

Identificação de quantidade de dados e duplicados

```
select * from woningen;
select count(*) from woningen;
select count(distinct(House_Id)),count(distinct(Project_Id)),count(distinct(House_LotArea)),count(distinct(House_LivingArea)) from woningen;
select count(*) from woningen where House_Id = Null;
select count(*) from woningen where Project_Id = Null;
select count(*) from woningen where House_LotArea = Null;
select count(*) from woningen where House_LivingArea = Null;

select * from userproject;
select count(*) from userproject;
select count(distinct(User_Id)),count(distinct(Project_Id)),count(distinct(User_Project_Date)) from userproject;
select count(*) from userproject where User_Id = Null;
select count(*) from userproject where Project_Id = Null;
select count(*) from userproject where User_Project_Date = Null;

select * from opties;
select * from items;

select * from useropties;
select count(*) from useropties where Option_Id=1;
select * from cbs_data_dictionary;
```

Passagem de tabelas originais do servidor para grupo 3

```
CREATE TABLE Inq_Questions_NEW AS SELECT Item_Id, Item_Project_Id, Item_Type, Item_Text_translated FROM vanwonen.items;
CREATE TABLE House_Sales_NEW AS SELECT * FROM vanwonen.koopovks;
CREATE TABLE AnswerOptions_NEW AS SELECT * FROM vanwonen.opties;
CREATE TABLE Projects_NEW AS SELECT * FROM vanwonen.projectenkoop;
CREATE TABLE Given_Answers_NEW AS SELECT * FROM vanwonen.useropties;
CREATE TABLE Subs_Projects_NEW AS SELECT * FROM vanwonen.userproject;
CREATE TABLE Users_NEW AS SELECT * FROM vanwonen.users;
CREATE TABLE Houses_FromProject_NEW AS SELECT * FROM vanwonen.woningen;
CREATE TABLE Dictionary1_NEW AS SELECT * FROM vanwonen.xitres_data_dictionary;
CREATE TABLE Dictionary2_NEW AS SELECT * FROM vanwonen.cbs_data_dictionary;
CREATE TABLE INE_NL_NEW AS SELECT * FROM vanwonen.dutch_neigh_cbsstats;
```

```
ALTER TABLE Subs_Projects_NEW RENAME TO Subs_Projects_T3;
```

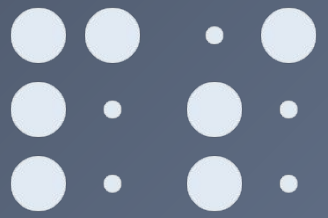
```
UPDATE INE_NL_T0 SET P_00_14_JR = NULL WHERE P_00_14_JR = '-99999999';
```

```
ALTER TABLE INE_NL_V3_T0 RENAME TO INE_NL_T0;
```

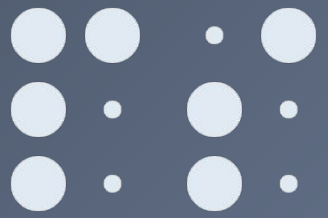
Limpeza de campos sem significado de -99999999 para null

```
select * from Dictionary2;
select distinct(Category) from Dictionary2;
select * from Dictionary2 where Category = "Population";
select * from Dictionary2 where Category = "Businesses";
select * from Dictionary2 where Category = "Living";
select * from Dictionary2 where Category = "Labour";
select * from Dictionary2 where Category = "Energy consumption private homes";
select * from Dictionary2 where Category = "Income";
select * from Dictionary2 where Category = "Social concern";
select * from Dictionary2 where Category = "Motor Vehicles";
select * from Dictionary2 where Category = "Surface";
select * from Dictionary2 where Category = "Services";
```

Interpretação das categorias do dicionário do “INE” Holandês



4. Modeling



4.1 Modeling - Regression

Modeling | Regression

Problema: Prever o preço máximo que o utilizador está disposto a pagar por um apartamento num determinado projeto.

Variáveis preditivas:
User_age
NBedrooms
Marital_status
Parent

Target:
Max_value

Algoritmos:
Regression Tree
Random Forest
Linear Regression
Multilinear Regression

Modeling | Regression (Resumo)

1ª Iteração

1) Definir as variáveis preditivas e o target

2) Dividir os dados em treino e teste (80/20)

3) Treinar o modelo

4) Avaliar o modelo:
   a) Mean Absolute Error
   b) Mean Square Error
   c) Root Mean Squared Error
   d) R squared

2ª Iteração

<u>Optimizar Hiperparametros:</u>

- GridSearchCV()

3ª Iteração

<u>Novas Variáveis preditivas:</u>

- User_age
- NBedrooms
- Marital_status
- Parent
- **Minimal_wage**
- **Municipality_Id**
- **Year**

Modeling | Regression | Dataset preparation (1/3)

Porto Business School
/ University of Porto

```
#Import tables
import io
df1 = pd.read_csv(io.BytesIO(uploaded['Age.csv']))
df2 = pd.read_csv(io.BytesIO(uploaded['Area.csv']))
df3 = pd.read_csv(io.BytesIO(uploaded['Family.csv']))
df4 = pd.read_csv(io.BytesIO(uploaded['Max_value.csv']))
df5 = pd.read_csv(io.BytesIO(uploaded['NBedrooms.csv']))
df6 = pd.read_csv(io.BytesIO(uploaded['Garage.csv']))

#Concatenate User_Id with Item_Project_Id to use as key to joining dataframes
df1['Concate']=df1['User_Id'].astype(str) + '_' +
df1['Item_Project_Id'].astype(str)
df2['Concate']=df2['User_Id'].astype(str) + '_' +
df2['Item_Project_Id'].astype(str)
df3['Concate']=df3['User_Id'].astype(str) + '_' +
df3['Item_Project_Id'].astype(str)
df4['Concate']=df4['User_Id'].astype(str) + '_' +
df4['Item_Project_Id'].astype(str)
df5['Concate']=df5['User_Id'].astype(str) + '_' +
df5['Item_Project_Id'].astype(str)
df6['Concate']=df6['User_Id'].astype(str) + '_' +
df6['Item_Project_Id'].astype(str)
```

```
df_1=pd.merge(df1,df2,on='Concate',how='outer')
df_2=pd.merge(df_1,df3,on='Concate',how='outer')
df_3=pd.merge(df_2,df4,on='Concate',how='outer')
df_4=pd.merge(df_3,df5,on='Concate',how='outer')
df_5=pd.merge(df_4,df6,on='Concate',how='outer')

df=df_5
df
df=df[['Concate','User_age','Max_area','NBedrooms','
Marital_status','Parent','User_garage','Max_value']]
df
```

	Concate	User_age	Max_area	NBedrooms	Marital_status	Parent	User_garage	Max_value
0	238686_0	55.0	NaN	3.0	1.0	1.0	1.0	NaN
1	173103_0	37.0	NaN	3.0	0.0	0.0	NaN	NaN
2	355677_9675	58.0	NaN	3.0	NaN	NaN	NaN	250000.0
3	355872_0	30.0	NaN	4.0	0.0	1.0	1.0	350000.0
4	335361_0	30.0	NaN	NaN	1.0	1.0	NaN	200000.0
...	...	...	...	...	...	...	...	...
44032	923684_12851	NaN	NaN	NaN	NaN	NaN	1.0	NaN
44033	48_12851	NaN	NaN	NaN	NaN	NaN	1.0	NaN
44034	968583_12851	NaN	NaN	NaN	NaN	NaN	1.0	NaN
44035	287961_12851	NaN	NaN	NaN	NaN	NaN	1.0	NaN
44036	923698_12851	NaN	NaN	NaN	NaN	NaN	0.0	NaN

44037 rows × 8 columns

Modeling | Regression | Dataset preparation (2/3)

```
df.isna().sum()
```

```
Concate              0
User_age         16880
Max_area         35667
NBedrooms        17008
Marital_status   23623
Parent           23623
User_garage      34440
Max_value         4626
dtype: int64
```

```
# Default configuration drops rows having at least 1 missing value

df=df.dropna()
df
```

	Concate	User_age	Max_area	NBedrooms	Marital_status	Parent	User_garage	Max_value
23661	568082_12851	37.0	100.0	3.0	0.0	1.0	1.0	375000.0
23662	568082_12851	37.0	100.0	3.0	0.0	1.0	1.0	425000.0
23663	568082_12851	37.0	100.0	3.0	0.0	1.0	1.0	340000.0
23665	713538_12851	37.0	120.0	4.0	0.0	1.0	0.0	500000.0
23666	713538_12851	37.0	120.0	4.0	0.0	1.0	0.0	500000.0
...	...	...	...	...	...	...	...	...
27116	295009_12851	55.0	120.0	3.0	1.0	1.0	0.0	625000.0
27117	295009_12851	55.0	120.0	3.0	1.0	1.0	0.0	625000.0
27118	1177726_12851	55.0	120.0	3.0	0.0	1.0	1.0	375000.0
27119	1177726_12851	55.0	120.0	3.0	0.0	1.0	1.0	325000.0
27120	1177726_12851	55.0	120.0	3.0	0.0	1.0	1.0	340000.0

186 rows × 8 columns

Construção do dataset:

1) Full outer join para juntarmos todas as variáveis criadas anteriormente;
2) Noção do problema dos NaNs.

Concluímos que as variáveis “Max_area” e “User_garage” têm muitos NaNs, por isso decidimos descartar estas duas variáveis para o problema de regressão.

Modeling | Regression | Dataset preparation (3/3)

	User_Id	Item_Project_Id	User_age	NBedrooms	Marital_status	Parent	Max_value
3	355872	0	30	4	0	1	350000
8	172382	0	65	2	1	1	400000
9	356094	0	55	2	0	0	350000
10	356301	0	37	4	0	1	200000
13	356508	0	55	3	1	1	350000
...	...	...	...	...	...	...	...
26519	295009	12851	55	3	1	1	500000
26520	295009	12851	55	3	1	1	625000
26521	1177726	12851	55	3	0	1	375000
26522	1177726	12851	55	3	0	1	325000
26523	1177726	12851	55	3	0	1	340000

13840 rows × 7 columns

Dataset final para o problema de regressão

```
#   Column           Non-Null Count  Dtype
0   User_Id          13840 non-null  category
1   Item_Project_Id  13840 non-null  category
2   User_age         13840 non-null  int64
3   NBedrooms        13840 non-null  int64
4   Marital_status   13840 non-null  category
5   Parent           13840 non-null  category
6   Max_value        13840 non-null  int64
```

4.2 Modeling - Regression - Regression Tree

Modeling | Regression - Decision Tree - 1 iteração

1) Definir as variáveis preditivas e o target

2) Dividir os dados em treino e teste (80/20)

3) Treinar o modelo com os seguintes hiperparâmetros:
   a) Max_depth = 10
   b) Min_samples_split = 20
   c) Critério = MSE

4) Avaliar o modelo:
   a) Mean Absolute Error: 75.008
   b) Mean Square Error: 12.174.524.136
   c) Root Mean Squared Error: 110.338
   d) R squared: 0,34

Modeling | Regression - Decision Tree - 1 iteração

```
#set preditors and target
preditors=df.iloc[:,:-1]
target=df.iloc[:,-1]
x=preditors
y=target
#Split data into 80% train and 20% test
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,
                                               y,
                                               test_size=0.2,
                                               random_state=1)
#Train model
from sklearn.tree import DecisionTreeRegressor
reg = DecisionTreeRegressor(criterion="mse",
                            max_depth=10,
                            min_samples_split=20,
                            random_state=1)
reg.fit(x_train,y_train)
y_pred=reg.predict(x_test)
y_pred
```

```
#Evaluate model
from sklearn.metrics import mean_absolute_error,
mean_squared_error, r2_score
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)
# print the results
print("Mean Absolute Error:", mae)
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
print("R-squared:", r2)
```

```
Mean Absolute Error: 75007.5731939463
Mean Squared Error: 12174524136.369076
Root Mean Squared Error: 110338.22608855499
R-squared: 0.33596970947340943
```

Modeling | Regression - Decision Tree - 2 iteração

1) Definir as variáveis preditivas e o target

2) Dividir os dados em treino e teste (80/20)

3) **Optimizar o modelo** recorrendo à função GridSearchCV()
   a) Max_depth = 8
   b) Min_samples_split = 185
   c) Critério = MSE

4) Treinar o modelo

5) Avaliar o modelo:
   a) Mean Absolute Error: 75.543
   b) Mean Square Error: 12.175.653.970
   c) Root Mean Squared Error: 110.343
   d) R squared: 0,34

Verificamos que o erro do algoritmo aumentou depois de termos optimizado os hiperparametros. Talvez seja um problema de overfitting e estamos a aproximar o algoritimo demasiado aos dados de treino. No sentido de melhor os nossos resultados, numa segunda iteração vamos recorrer a novas variaveis "**Minimal_wage**" e "**Municipality_Id**".

Modeling | Regression - Decision Tree - 2 iteração

Porto Business School / University of Porto

```
#Optimize Hyperparameters
from sklearn.model_selection import GridSearchCV
model = DecisionTreeRegressor()
gs = GridSearchCV(model,
                  param_grid = {'max_depth': range(1, 50),
                                'min_samples_split': range(10, 200, 5)},
                  cv=5,
                  n_jobs=1,
                  scoring='neg_mean_squared_error')
gs.fit(x_train, y_train)

print(gs.best_params_)
print(-gs.best_score_)

{'max_depth': 14, 'min_samples_split': 185}
```

Verificamos que o erro do algoritmo aumentou depois de termos optimizado os hiperparametros. Talvez seja um problema de overfitting e estamos a aproximar o algoritimo demasiado aos dados de treino. No sentido de melhor os nossos resultados, numa segunda iteração vamos recorrer a novas variaveis “Minimal_wage” e “Municipality_Id”.

```
#Train model
reg = DecisionTreeRegressor(criterion="mse",
                            max_depth=8,
                            min_samples_split=185,
                            random_state=1)
reg.fit(x_train,y_train)
y_pred=reg.predict(x_test)
y_pred
#Evaluate model
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)
# print the results
print("Mean Absolute Error:", mae)
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
print("R-squared:", r2)
```

New

```
Mean Absolute Error: 75543.22674521306
Mean Squared Error: 12175653970.180445
Root Mean Squared Error: 110343.34583553484
R-squared: 0.335908085391383
```

Old

```
Mean Absolute Error: 75007.573193946
Mean Squared Error: 12174524136.369076
Root Mean Squared Error: 110338.22608855499
R-squared: 0.3359697094734094
```

Modeling | Regression - Decision Tree - 3 iteração

1) Adicionar novas features ao dataset

2) Definir as variáveis preditivas e o target

3) Dividir os dados em treino e teste (80/20)

4) Optimizar o modelo recorrendo à função GridSearchCV()
   a) Max_depth = 8
   b) Min_samples_split = 185
   c) Critério = MSE

5) Treinar o modelo

6) Avaliar o modelo:
   a) Mean Absolute Error: 69.191
   b) Mean Square Error: 10.459.579.825
   c) Root Mean Squared Error: 102.272
   d) R squared: 0,35

Variáveis preditivas:

User_age
NBedrooms
Marital_status
Parent
Minimal_wage
Municipality_Id
Year

Modeling | Regression - Decision Tree - 3 iteração

Porto Business School / University of Porto

Municipality_Id e Minimal_wage

```
mydb = mysql.connector.connect(**config)

query = "SELECT distinct(Municipality_Id),Project_Id FROM
vanwonen.projectenkoop"

df = pd.read_sql(query, con = mydb)
df
```

	Municipality_Id	Project_Id
0	98.0	3930.0
1	98.0	3931.0
2	321.0	7368.0
3	321.0	7369.0
4	28.0	9395.0
...	...	...
168	21.0	15242.0
169	98.0	15249.0
170	106.0	15251.0
171	106.0	15252.0
172	106.0	15253.0

173 rows × 2 columns

```
data = {'Year':  [2022, 2021,2020,2019,2018,2017],
        'Minimal_wage': [1756, 1701,1680,1636, 1594,1565]
        }

df = pd.DataFrame(data)
df
```

	Year	Minimal_wage
0	2022	1756
1	2021	1701
2	2020	1680
3	2019	1636
4	2018	1594
5	2017	1565

Modeling | Regression - Decision Tree - 3 iteração

```
#set preditors and target
preditors=df[['Year','User_age','NBedrooms','Marital_status','Parent
','Minimal_wage','Municipality_Id']]
target=df[['Max_value']]
x=preditors
y=target

#Split data into 80% train and 20% test
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,
                                               y,
                                               test_size=0.2,
                                               random_state=1)

#Train model
from sklearn.tree import DecisionTreeRegressor
reg = DecisionTreeRegressor(criterion="mse",
                            max_depth=10,
                            min_samples_split=20,
                            random_state=1)
reg.fit(x_train,y_train)
y_pred=reg.predict(x_test)
y_pred
```

```
#Evaluate model
from sklearn.metrics import
mean_absolute_error,
mean_squared_error, r2_score
mae = mean_absolute_error(y_test,
y_pred)
mse = mean_squared_error(y_test,
y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)
# print the results
print("Mean Absolute Error:", mae)
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:",
rmse)
print("R-squared:", r2)
```

```
Mean Absolute Error:
69101.20218019342
Mean Squared Error:
10459579825.011389
Root Mean Squared Error:
102272.08722330541
R-squared: 0.346270901570482
```

New

```
Mean Absolute Error:
75543.22674521306
Mean Squared Error:
12175653970.180445
Root Mean Squared Error:
110343.34583553484
R-squared: 0.3359080853913833
```

Old

Modeling | Regression - Decision Tree

1ª Iteração - Modelo preliminar

a) Mean Absolute Error: 75.008
b) Mean Square Error: 12.174.524.136
c) Root Mean Squared Error: 110.338
d) R squared: 0,34

2ª Iteração - Optimização dos hiperparâmetros

a) Mean Absolute Error: 75.543 (+535)
b) Mean Square Error: 12.175.653.970 (+1.129.834)
c) Root Mean Squared Error: 110.343 (+5)
d) R squared: 0,34 (0)

3ª Iteração - Novas features

a) Mean Absolute Error: 69.191 (-6352)
b) Mean Square Error: 10.459.579.825 (-1.716.074.145)
c) Root Mean Squared Error: 102.272 (-8071)
d) R squared: 0,35 (+0.01)

4.3 Modeling - Regression - Random Forest

Modeling | Regression - Random Forest - 1 iteração

1) Dividir os dados em treino e teste (80/20)

2) Treinar o modelo com os seguintes hiperparâmetros:
   a) n_estimator = 100

3) Avaliar o modelo:
   a) Mean Absolute Error: 69.399
   b) Mean Square Error: 10.523.851.521
   c) Root Mean Squared Error: 102.586
   d) R squared: 0,34

Modeling | Regression - RandomForest - 1 iteração

```
#set preditors and target
preditors=df[['Year','User_age','NBedrooms','Marital_status','Parent
','Minimal_wage','Municipality_Id']]
target=df[['Max_value']]
x=preditors
y=target

#Split data into 80% train and 20% test
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,
                                               y,
                                               test_size=0.2,
                                               random_state=1)

#Train model
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(n_estimators=100,random_state=1)
rf.fit(x_train,y_train)
y_pred =rf.predict(x_test)
```

```
# calculate the evaluation metrics
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

# print the results
print("Mean Absolute Error:", mae)
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
print("R-squared:", r2)
```

```
Mean Absolute Error:
69398.61507972967
Mean Squared Error:
10523851520.77756
Root Mean Squared Error:
102585.82514547299
R-squared: 0.34225388765303144
```

Modeling | Regression - Random Forest - 2 iteração

1) Dividir os dados em treino e teste (80/20)

2) **Optimizar o modelo** recorrendo à função GridSearchCV()
   a) n_estimator = 56
   b) max_features = ‘sqrt’
   c) max_depth = 20
   d) min_samples_split = 10
   e) min_samples_leaf = 2
   f) bootstrap = True

3) Avaliar o modelo:
   a) Mean Absolute Error: 69.023
   b) Mean Square Error: 10.425.656.896
   c) Root Mean Squared Error: 102.106
   d) R squared: 0,35

Modeling | Regression - RandomForest - 2 iteração

```
#Optimize Hyperparameters

# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 10, stop = 80,
num = 10)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [1,20]
# Minimum number of samples required to split a node
min_samples_split = [10, 200]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2]
# Method of selecting samples for training each tree
bootstrap = [True, False]
# Create the param grid
param_grid = {'n_estimators': n_estimators,
              'max_features': max_features,
              'max_depth': max_depth,
              'min_samples_split': min_samples_split,
              'min_samples_leaf': min_samples_leaf,
              'bootstrap': bootstrap}
```

```
print(param_grid)
rf = RandomForestRegressor()
from sklearn.model_selection import GridSearchCV
rf_Grid = GridSearchCV(estimator = rf, param_grid = param_grid, cv =
3, verbose=2, n_jobs = 4)
rf_Grid.fit(x_train, y_train)
```

```
{'bootstrap': True,
 'max_depth': 20,
 'max_features': 'sqrt',
 'min_samples_leaf': 2,
 'min_samples_split': 10,
 'n_estimators': 56}
```

Modeling | Regression - RandomForest - 2 iteração

```
#Train model
rf = RandomForestRegressor(bootstrap = True, max_depth = 20,
max_features = "sqrt", min_samples_leaf =2, min_samples_split=10,
n_estimators=56, random_state=1)
rf.fit(x_train,y_train)
y_pred =rf.predict(x_test)

#Evaluate model
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)
```

```
# print the results
print("Mean Absolute Error:", mae)
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
print("R-squared:", r2)
```

```
Mean Absolute Error: 69023.8051635946
Mean Squared Error: 10425656896.162098
Root Mean Squared Error: 102106.10606698357
R-squared: 0.34839110200527457
```

New

```
Mean Absolute Error:
69023.8051635946
Mean Squared Error:
10425656896.162098
Root Mean Squared Error:
102106.10606698357
R-squared: 0.34839110200527457
```

Old

```
Mean Absolute Error:
69398.61507972967
Mean Squared Error:
10523851520.77756
Root Mean Squared Error:
102585.82514547299
R-squared: 0.34225388765303144
```

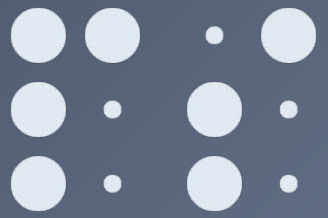
Modeling | Regression - Random Forest

1ª Iteração - Modelo preliminar

a) Mean Absolute Error: 69.399
b) Mean Square Error: 10.523.851.521
c) Root Mean Squared Error: 102.586
d) R squared: 0,34

2ª Iteração - Optimização dos hiperparâmetros

a) Mean Absolute Error: 69.023 (-376)
b) Mean Square Error: 10.425.656.896(-98.194.625)
c) Root Mean Squared Error: 102.106 (-480)
d) R squared: 0,35 (+0,01)



4.4 Modeling - Regression - Linear Regression

Modeling | Regression - Linear Regression

1) Dividir os dados em treino e teste (80/20)

2) Avaliar o modelo:
   a) Mean Absolute Error: 80.558
   b) Mean Square Error: 13.982.772.137
   c) Root Mean Squared Error: 118.248
   d) R squared: 0,12

Modeling | Regression - Linear Regression

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

# Assume data is in a pandas dataframe called 'data'

# Assign predictor and target variables
X =
df[['Year','User_age','NBedrooms','Marital_status','Parent','Minimal
_wage','Municipality_Id']]
y = df[['Max_value']]

# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2)

# Create and fit the model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)
```

```
# calculate the evaluation metrics
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

# print the results
print("Mean Absolute Error:", mae)
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
print("R-squared:", r2)
```

```
Mean Absolute Error: 80558.88563837457
Mean Squared Error: 13982772137.006016
Root Mean Squared Error: 118248.77224312317
R-squared: 0.11630905376302658
```

4.5 Modeling - Regression - MultiLinear Regression

Modeling | Regression - Mutilinear Regression

1) Dividir os dados em treino e teste (80/20)

2) Avaliar o modelo:
   a) Mean Absolute Error: 79.176
   b) Mean Square Error: 13.384.386.986
   c) Root Mean Squared Error: 115.691
   d) R squared: 0,09

Modeling | Regression - MultiLinear Regression

Porto Business School / University of Porto

```
import statsmodels.api as sm
import pandas as pd
from sklearn.model_selection import train_test_split

# Assign predictor and target variables
X =
df[['Year','User_age','NBedrooms','Marital_status','Parent','Minimal
_wage','Municipality_Id']]
y = df[['Max_value']]

# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2)

# Add a constant to the predictor variable of training set
X_train = sm.add_constant(X_train)

# Create the model
model = sm.OLS(y_train, X_train).fit()

# Make predictions on the test set
X_test = sm.add_constant(X_test)
y_pred = model.predict(X_test)
```

```
# calculate the evaluation metrics
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

# print the results
print("Mean Absolute Error:", mae)
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
print("R-squared:", r2)
```

```
Mean Absolute Error: 79175.56132907128
Mean Squared Error: 13384386985.698694
Root Mean Squared Error: 115690.91142219727
R-squared: 0.08948410228963732
```

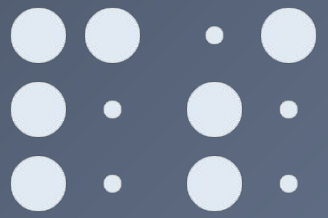
Modeling | Regression - Comparação dos modelos

Error	RegressionTree	RandomForest	LinearRegression	MultiLinearRegression
Mean Absolute Error	69101.20218	69023.80516	80558.88564	79175.56133
Mean Squared Error	10459579825	10425656896	13982772137	13384386986
Root Mean Squared Error	102272.0872	102106.1061	118248.7722	115690.9114
R-squared	0.346270902	0.348391102	0.116309054	0.089484102

```
#Feature Importance
for importance, name in sorted(zip(rf.feature_importances_, x_train.columns),reverse=True):
 print (name, importance)
```

```
Municipality_Id 0.3052697270637107
User_age 0.222737046170669
Year 0.15897506623037017
Minimal_wage 0.14235976268477757
NBedrooms 0.10126768469428857
Marital_status 0.040857824902605346
Parent 0.02853288825357869
```

Através da análise da feature importance percebemos que a localização é a variável que influencia mais o preço que o utilizador está disposto a pagar. Seguida da idade do utilizador, é natural que o user mais velho tenha uma situação financeira mais estável que um utilizador mais novo. De seguida encontramos as variáveis ano e salário mínimo. O salário mínimo pode estar relacionado com o poder de compra do utilizador, enquanto o ano poderá ser influenciado pela inflação e valorização do preço dos imóveis com o tempo.



4.6 Modeling - Classification

Modeling | Classification

Problema: Prever se o user quer lugar de garagem

Variáveis preditivas:
User_age
NBedrooms
Marital_status
Parent

Target:
User_garage

Algoritmos:
Classification Tree
Support Vector Machine
XGBoost

Modeling | Classification (Resumo)

1ª Iteração

1) Definir as variáveis preditivas e o target

2) Dividir os dados em treino e teste (70/30)

3) Treinar o modelo

4) Avaliar o modelo:
   a) Accuracy: 0,85
   b) Precision: 0,82
   c) Recall: 0,85
   d) F1-score: 0,80

2ª Iteração

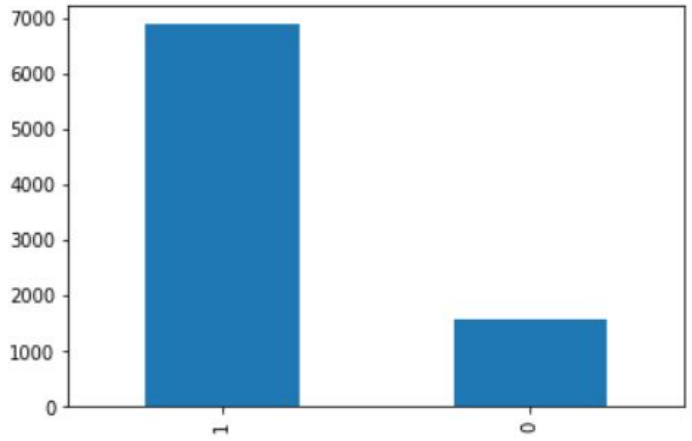
Optimizar Hiperparametros:

- GridSearchCV()

3ª Iteração

Dados desbalanceados: SMOTE

User_garage



1: Quer lugar de garagem
0: Não quer lugar de garagem

4.7 Modeling - Classification - Classification Tree

Modeling | Classification - Decision Tree - 1 iteração

1) Definir as variáveis preditivas e o target

2) Dividir os dados em treino e teste (70/30)

3) Treinar o modelo

4) Avaliar o modelo:
   a) Accuracy: 0,85
   b) Precision: 0,82
   c) Recall: 0,85
   d) F1-score: 0,80

Modeling | Classification - Classification Tree - 1 iteração

```
#Import library required
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

#Import tables
from google.colab import files
uploaded = files.upload()
import io
df = pd.read_csv(io.BytesIO(uploaded['Dataset_Classification.csv']))

# load your data into X and y
preditors=df[['User_age','NBedrooms','Marital_status','Parent']]
target=df[['User_garage']]
X=preditors
y=target
```

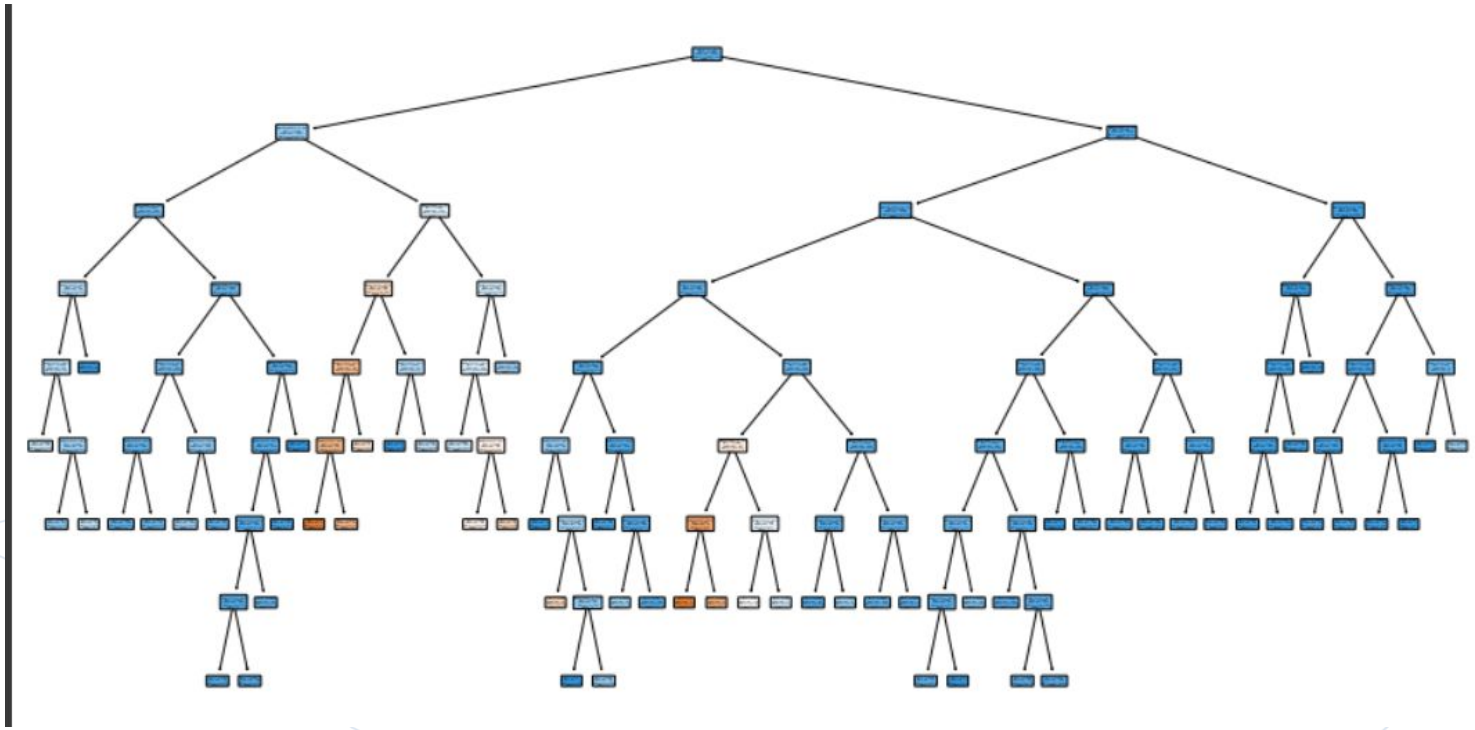
Modeling | Classification - Classification Tree - 1 iteração

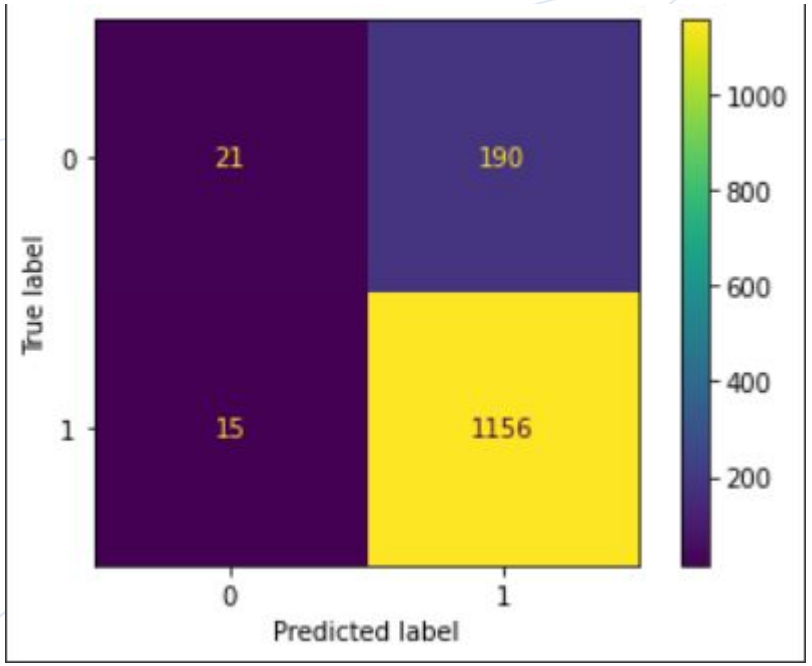
Porto Business School / University of Porto

```
# split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)

#create a decision tree and fit it to the training data
clf_dt = DecisionTreeClassifier(random_state=42)
clf_dt = clf_dt.fit(X_train, y_train)

#plot the decision tree (training)
plt.figure(figsize=(15,7.5))
plot_tree(clf_dt,
          filled=True,
          rounded=True,
          class_names=["0","1"],
          feature_names=X.columns);

#plot_confusion_matrix() will run the test down the tree and draw a
confusion matrix
plot_confusion_matrix(clf_dt, X_test, y_test, display_labels = ["0", "1"])
```

```
# make predictions on the test data
y_pred = clf_dt.predict(X_test)

# evaluate the classifier using various metrics
acc = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred, average='weighted')
rec = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

# print the results
print("Accuracy: ", acc)
print("Precision: ", prec)
print("Recall: ", rec)
print("F1-Score: ", f1)
```

Modeling | Classification - Classification Tree - 1 iteração






```
Accuracy:  0.8516642547033285
Precision:  0.8167772478387303
Recall:  0.8516642547033285
F1-Score:  0.8042728524177347
```

Modeling | Classification - Decision Tree - 2 iteração

1) Definir as variáveis preditivas e o target

2) Dividir os dados em treino e teste (70/30)

3) Optimizar os hiperparâmetros recorrendo ao “**cost complexity pruning**”

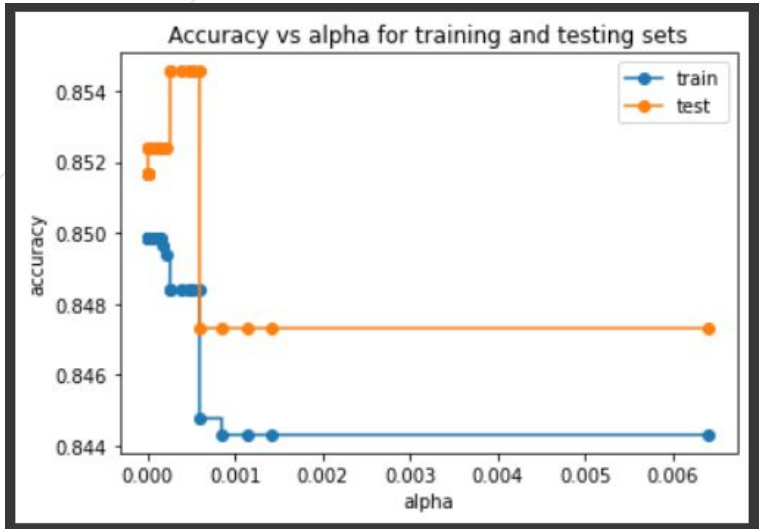
4) Treinar o modelo

5) Avaliar o modelo:
   a) Accuracy: 0,85
   b) Precision: 0,84 (+0,02)
   c) Recall: 0,86 (+0,01)
   d) F1-score: 0,80

Modeling | Classification - Classification Tree - 2 iteração

```
path = clf_dt.cost_complexity_pruning_path(X_train, y_train) #determines values for alpha
ccp_alphas = path.ccp_alphas # extract different values for alpha
ccp_alphas = ccp_alphas[:-1] # exclude the maximum value for alpha

clf_dts = [] #create an array that will put the decision trees into

##now create one decision tree per value for alpha and store it in the array
for ccp_alpha in ccp_alphas:
  clf_dt = DecisionTreeClassifier(random_state=0, ccp_alpha=ccp_alpha)
  clf_dt.fit(X_train,y_train)
  clf_dts.append(clf_dt)

train_scores = [clf_dt.score(X_train, y_train) for clf_dt in clf_dts]
test_scores = [clf_dt.score(X_test, y_test) for clf_dt in clf_dts]

fig, ax = plt.subplots()
ax.set_xlabel("alpha")
ax.set_ylabel('accuracy')
ax.set_title('Accuracy vs alpha for training and testing sets')
ax.plot(ccp_alphas, train_scores, marker='o', label='train', drawstyle='steps-post')
ax.plot(ccp_alphas, test_scores, marker='o', label='test', drawstyle='steps-post')
ax.legend()
plt.show()
```

As árvores de decisão tendem a sofrer overfitting do dataset de treino e existem muitos parâmetros tais como “max_depth” e “min_samples” que podem ser utilizados para reduzir o “overfitting”. Contudo, se “podarmos” a árvore recorrendo ao “ cost complexity pruning” conseguimos simplificar esse processo de encontrar árvores mais pequenas e ainda melhorar a precisão do dataset de teste.

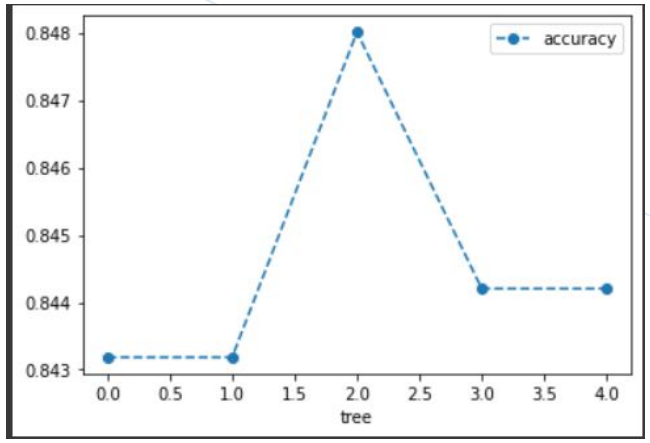


Modeling | Classification - Classification Tree - 2 iteração

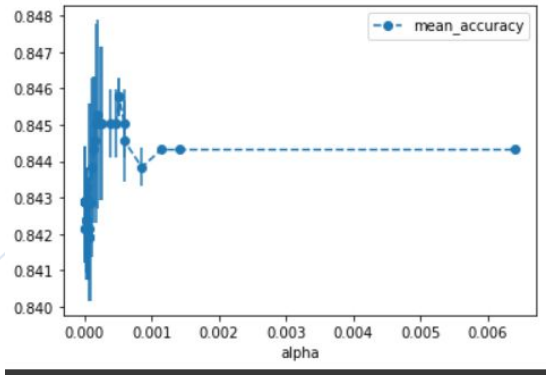
Porto Business School / University of Porto

Vamos encontrar o melhor alpha usando Cross Validation.

```
clf_dt = DecisionTreeClassifier(random_state=42, ccp_alpha=0.00015)
#create the tree with ccp_alpha=0.00015

## now user 5-fold cross validation create 5 different training and
testing dataset that are then used to train and test the tree
scores = cross_val_score(clf_dt, X_train, y_train, cv=5)
df = pd.DataFrame(data={'tree': range(5), 'accuracy': scores})

df.plot(x='tree', y='accuracy', marker='o', linestyle='--')
```




```
alpha_loop_values = []

for ccp_alpha in ccp_alphas:
  clf_dt = DecisionTreeClassifier(random_state=0, ccp_alpha=ccp_alpha)
  scores = cross_val_score(clf_dt,X_train,y_train,cv=5)
  alpha_loop_values.append([ccp_alpha, np.mean(scores), np.std(scores)])

alpha_results = pd.DataFrame(alpha_loop_values,
                             columns=['alpha', 'mean_accuracy','std'])
alpha_results.plot(x='alpha',
                   y='mean_accuracy',
                   yerr='std',
                   marker='o',
                   linestyle='--')
```



Modeling | Classification - Classification Tree - 2 iteração

```
alpha_results[(alpha_results['alpha']>0.0002)
              &
              (alpha_results['alpha']<0.003)]
```

	alpha	mean_accuracy	std
35	0.000202	0.845281	0.002606
36	0.000244	0.845039	0.002131
37	0.000261	0.845039	0.002131
38	0.000378	0.845040	0.000931
39	0.000474	0.845040	0.000931
40	0.000515	0.845764	0.000526
41	0.000516	0.845764	0.000526
42	0.000591	0.845040	0.000931
43	0.000594	0.844557	0.001123
44	0.000853	0.843833	0.000534
45	0.001142	0.844316	0.000092
46	0.001416	0.844316	0.000092

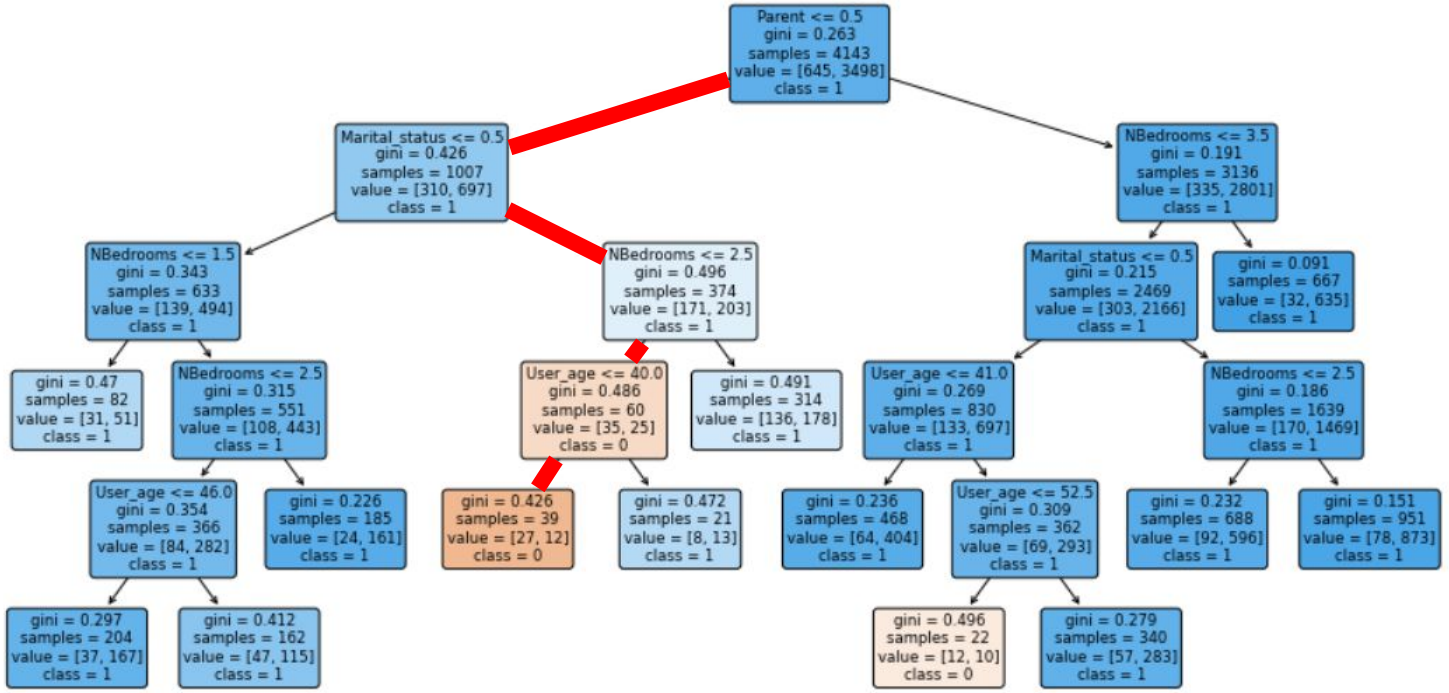
```
ideal_ccp_alpha = float(0.000474)
ideal_ccp_alpha
```

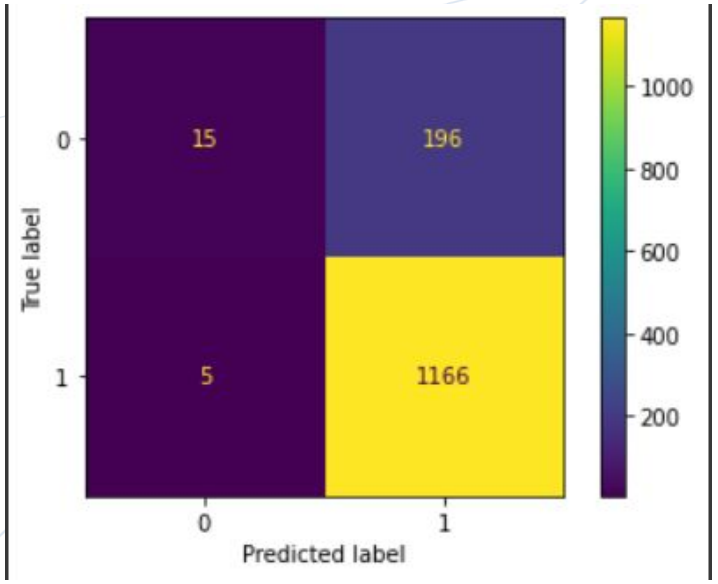
```
##Build and train a new decision tree, only this time use the optimal value
of alpha
```

```
clf_dt_prunned = DecisionTreeClassifier(random_state=42,
                                        ccp_alpha=ideal_ccp_alpha)
clf_dt_prunned = clf_dt_prunned.fit(  X_train,y_train)
```

```
 #plot_confusion_matrix() will run the test down the tree and draw a
confusion matrix
plot_confusion_matrix(clf_dt_prunned,
                      X_test,
                      y_test,
                      display_labels = ["0", "1"])
#plot the decision tree (training)
plt.figure(figsize=(15,7.5))
plot_tree(clf_dt_prunned,
          filled=True,
          rounded=True,
          class_names=["0","1"],
          feature_names=X.columns);
```

Modeling | Classification - Classification Tree - 2 iteração



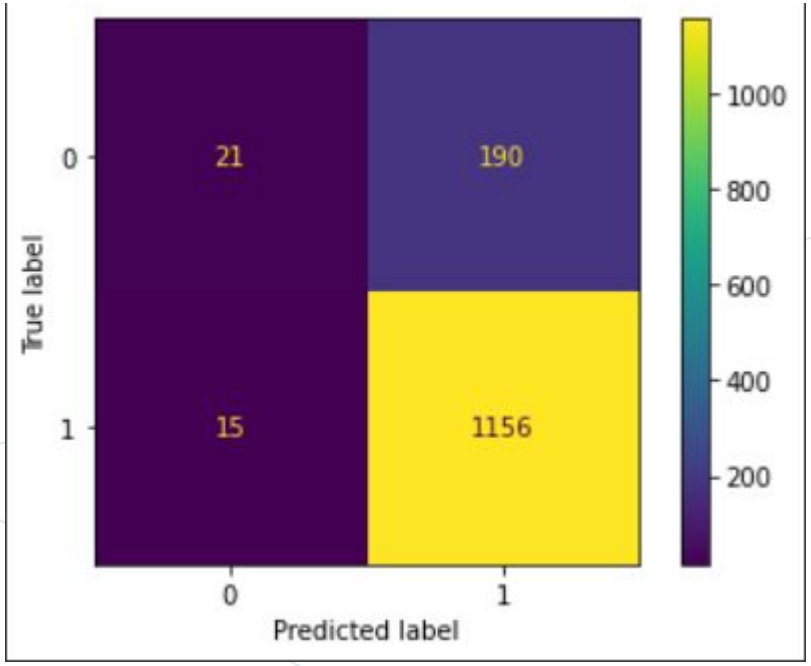



```
Accuracy:  0.8545586107091172
Precision:  0.8398958393101147
Recall:  0.8545586107091172
F1-Score:  0.7999137232869906
```

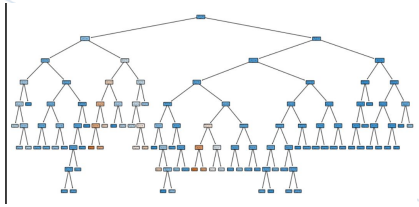
Users sem filhos, solteiros, que desejem T2, T1 ou T0 e tenham 40 anos ou menos é provável que não queiram lugar de garagem.

Modeling | Classification - Classification Tree - 2 iteração

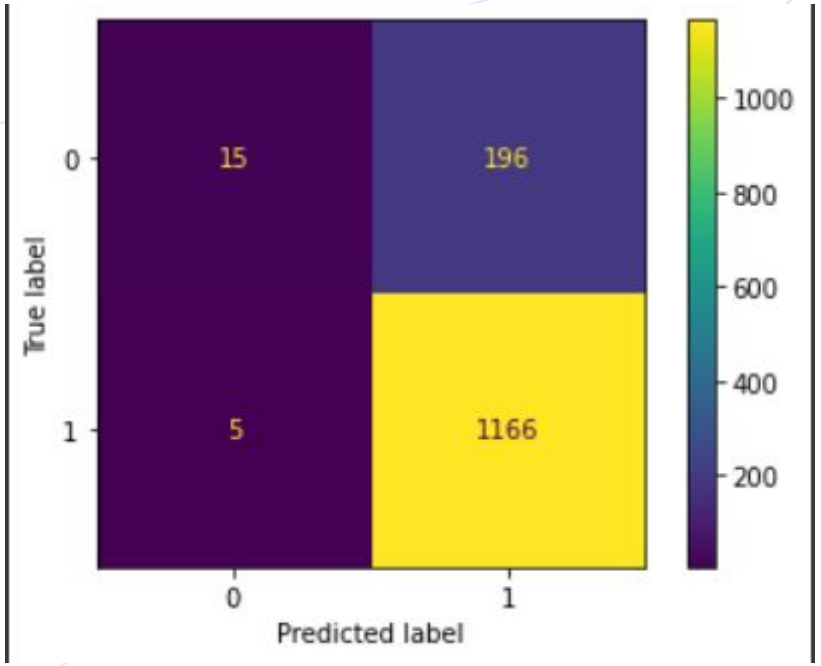
1ª iteração



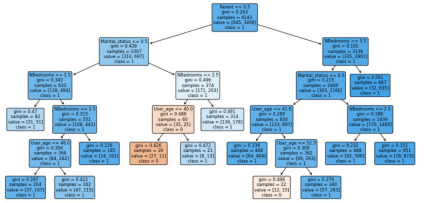

```
Accuracy:  0.852
Precision:  0.817
Recall:  0.852
F1-Score:  0.804
```



2ª iteração

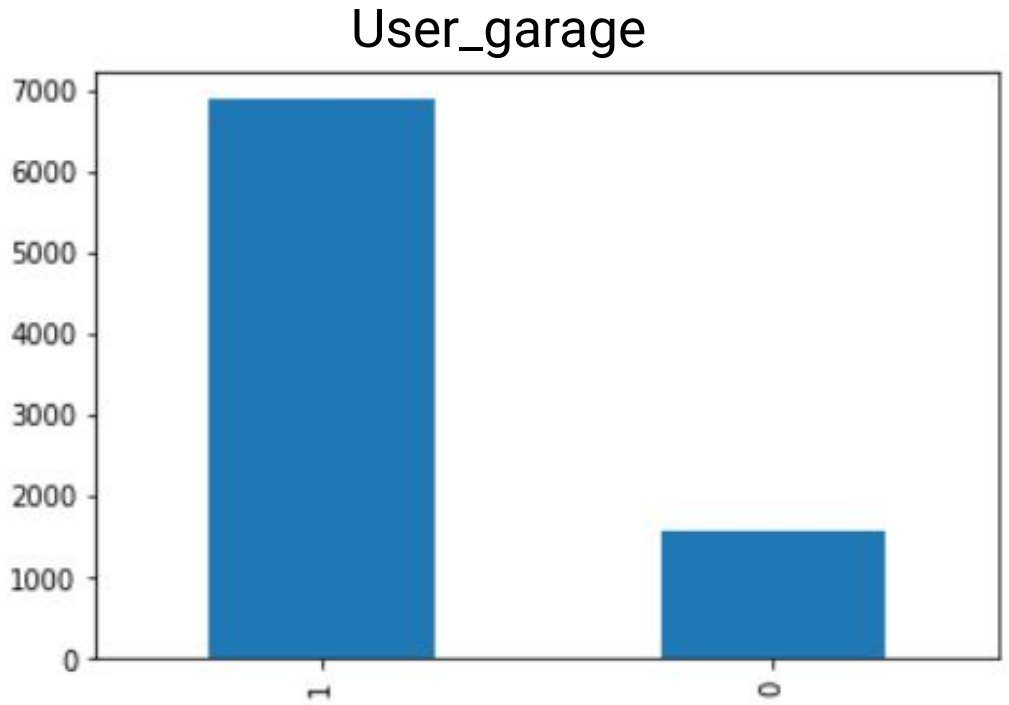



```
Accuracy:  0.854
Precision:  0.840
Recall:  0.855
F1-Score:  0.800
```



Comparação da 1ª iteração com a 2ª iteração

Modeling | Classification - Classification Tree - 3 iteração



1: Quer lugar de garagem
0: Não quer lugar de garagem

Como podemos verificar pela análise do histograma, já apresentado anteriormente, as nossas **classes estão desbalanceadas**. Por isso vamos usar o Smote para lidar com essa situação.

0	1571 (19%)
1	6883 (81%)

Modeling | Classification - Decision Tree - 3 iteração

1) Definir as variáveis preditivas e o target

2) Dividir os dados em treino e teste (70/30)

3) Optimizar os hiperparâmetros recorrendo ao “cost complexity pruning”

4) Lidar com os dados desbalanceados recorrendo ao **SMOTE**

5) Treinar o modelo

6) Avaliar o modelo:
   a) Accuracy: 0,74 (-0,11)
   b) Precision: 0,82
   c) Recall: 0,74 (-0,11)
   d) F1-score: 0,77 (-0,03)

Modeling | Classification - Classification Tree - 3 iteração

Without Smote

	precision	recall	f1-score	support
0	0.30	0.61	0.40	211
1	0.91	0.75	0.82	1171
accuracy			0.73	1382
macro avg	0.61	0.68	0.61	1382
weighted avg	0.82	0.73	0.76	1382

With Smote

	precision	recall	f1-score	support
0	0.31	0.59	0.41	211
1	0.91	0.77	0.83	1171
accuracy			0.74	1382
macro avg	0.61	0.68	0.62	1382
weighted avg	0.82	0.74	0.77	1382

Modeling | Classification - Classification Tree - 3 iteração

```
# Create an instance of the decision tree classifier
clf = DecisionTreeClassifier()

# Perform cross-validation using the function cross_val_score()
# Here we use 5-fold cross-validation
scores = cross_val_score(clf, X, y, cv=5)

#print report
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.30      0.61      0.40       211
           1       0.91      0.75      0.82      1171

    accuracy                           0.73      1382
   macro avg       0.61      0.68      0.61      1382
weighted avg       0.82      0.73      0.76      1382
```

```
# Create an instance of the SMOTE class
sm = SMOTE()

# Perform oversampling using the fit_resample() method of SMOTE
X_sm, y_sm = sm.fit_resample(X_train, y_train)

# Create an instance of the decision tree classifier
clf_sm = DecisionTreeClassifier()

# Perform cross-validation using the function cross_val_score()
# Here we use 5-fold cross-validation
scores = cross_val_score(clf_sm, X_sm, y_sm, cv=5)

# Fit the classifier on the oversampled data
clf_sm.fit(X_sm, y_sm)

# Use the classifier to predict on the test set
y_pred = clf_sm.predict(X_test)

#print report
print(classification_report(y_test,y_pred))
```

Modeling | Classification - Classification Tree - 3 iteração

Porto Business School / University of Porto

Without Smote

	precision	recall	f1-score	support
0	0.30	0.61	0.40	211
1	0.91	0.75	0.82	1171
accuracy			0.73	1382
macro avg	0.61	0.68	0.61	1382
weighted avg	0.82	0.73	0.76	1382

With Smote

	precision	recall	f1-score	support
0	0.31	0.59	0.41	211
1	0.91	0.77	0.83	1171
accuracy			0.74	1382
macro avg	0.61	0.68	0.62	1382
weighted avg	0.82	0.74	0.77	1382

```
# evaluate the classifier using various metrics
acc = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred, average='weighted')
rec = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

# print the results
print("Accuracy: ", acc)
print("Precision: ", prec)
print("Recall: ", rec)
print("F1-Score: ", f1)
```

```
Accuracy:  0.7387843704775687
Precision:  0.819974787903018
Recall:  0.7387843704775688
F1-Score:  0.7675554191809674
```

4.8 Modeling - Classification - Support Vector Machine

Modeling | Classification - Support Vector Machines

1) Definir as variáveis preditivas e o target

2) Dividir os dados em treino e teste (70/30)

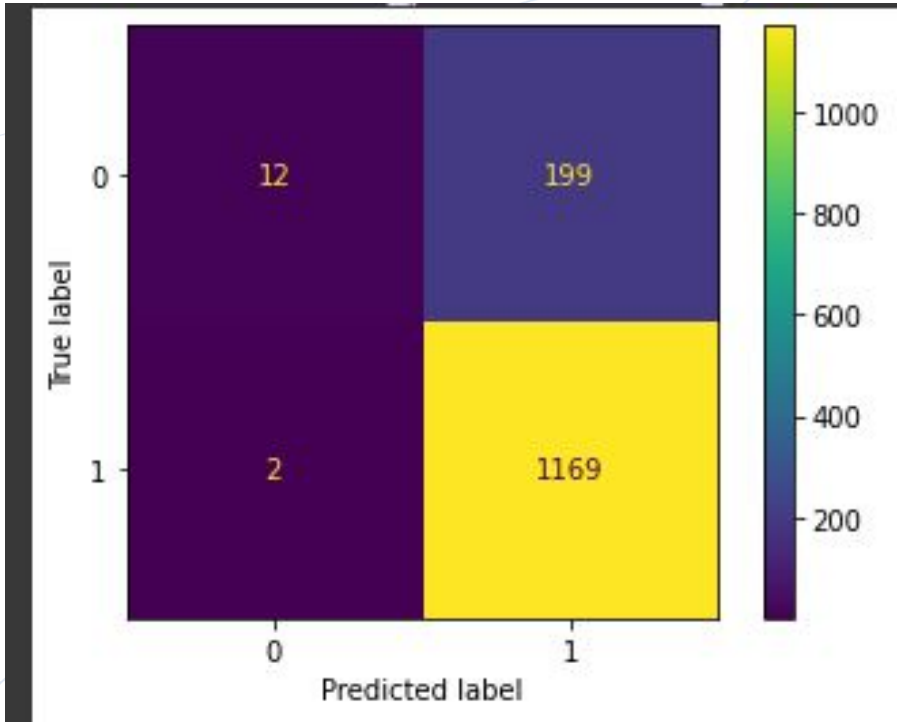
3) Centrar e escalar os dados

4) Optimizar os hiperparâmetros recorrendo ao GridSearchCV()
   a) C
   b) gamma
   c) kernel

5) Lidar com os dados desbalanceados recorrendo ao SMOTE

6) Treinar o modelo

7) Avaliar o modelo:
   a) Accuracy: 0,85
   b) Precision: 0,85
   c) Recall: 0,85
   d) F1-score: 0,80



Modeling | Classification - Support Vector Machines

Porto Business School / University of Porto

Centering and scaling: Radial Basis Function assumes that the data is centered and scaled (mean=0 and sd=1)

```
# split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
X_train_scaled=scale(X_train)
X_test_scaled=scale(X_test)
```

Build a preliminary SVM

```
clf_svm = SVC(random_state=42)
clf_svm.fit(X_train_scaled,y_train)
```

Optimize Parameters with Cross Validation and GridSearchCV()

```
param_grid = [
    {'C': [0.5, 1, 10, 100],
     'gamma': ['scale', 1, 0.1, 0.01, 0.001, 0.0001],
     'kernel':['rbf']},
]
optimal_params = GridSearchCV(
    SVC(),
    param_grid,
    cv=5,
    scoring='accuracy',
    verbose=0
)
```

```
optimal_params.fit(X_train_scaled,y_train)
print(optimal_params.best_params_)
clf_svm = SVC(random_state=42, C=0.5, gamma='scale', kernel='rbf')
clf_svm.fit(X_train_scaled,y_train)
```

```
{'C': 0.5, 'gamma': 'scale', 'kernel': 'rbf'}
```

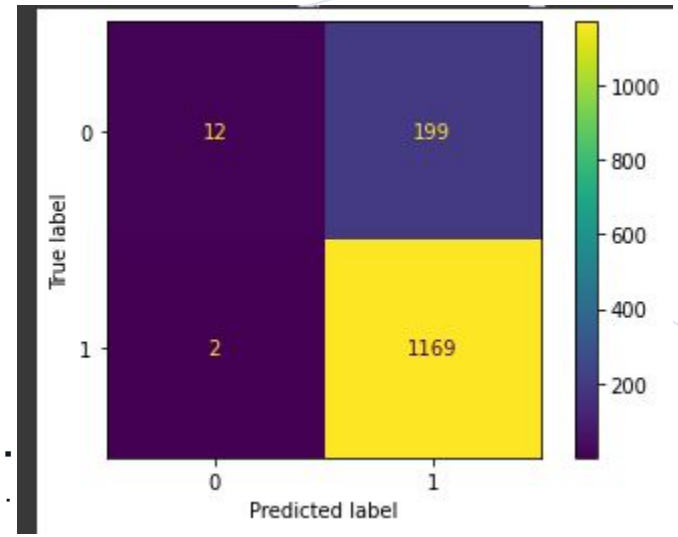
SMOTE

```
# Create an instance of the SMOTE class
sm = SMOTE()
# Perform oversampling using the fit_resample() method of SMOTE
X_sm, y_sm = sm.fit_resample(X_train_scaled, y_train)
# Create an instance of the decision tree classifier
clf_svm_sm = SVC(random_state=42, C=0.5, gamma='scale', kernel='rbf')
# Perform cross-validation using the function cross_val_score()
# Here we use 5-fold cross-validation
scores = cross_val_score(clf_svm_sm, X_sm, y_sm, cv=5)
# Fit the classifier on the oversampled data
clf_svm_sm.fit(X_train_scaled,y_train)
# Use the classifier to predict on the test set
y_pred = clf_svm_sm.predict(X_test_scaled)
```

Modeling | Classification - Support Vector Machines

Porto Business School / University of Porto

```
plot_confusion_matrix(clf_svm_sm,
                      X_test_scaled,
                      y_test,
                      values_format='d',
                      display_labels=['0', '1'])
```




```
# evaluate the classifier using various metrics
acc = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred, average='weighted')
rec = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

# print the results
print("Accuracy: ", acc)
print("Precision: ", prec)
print("Recall: ", rec)
print("F1-Score: ", f1)
```

```
Accuracy:  0.8545586107091172
Precision:  0.8549307573081281
Recall:  0.8545586107091172
F1-Score:  0.796529972278096
```

4.9 Modeling - Classification - XGBoost

Modeling | Classification - XGBoost

1) Definir as variáveis preditivas e o target

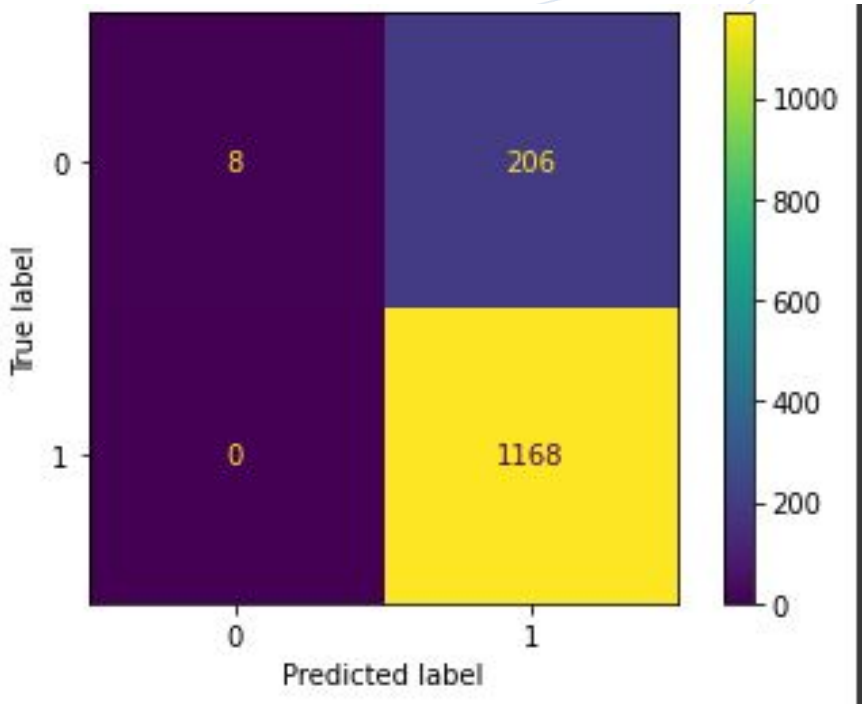
2) Dividir os dados em treino e teste (70/30)

3) Centrar e escalar os dados

4) Optimizar os hiperparâmetros recorrendo ao GridSearchCV()
   a) max_depth
   b) learning_rate
   c) gamma
   d) reg_lambda
   e) scale_pos_weight

5) Treinar o modelo

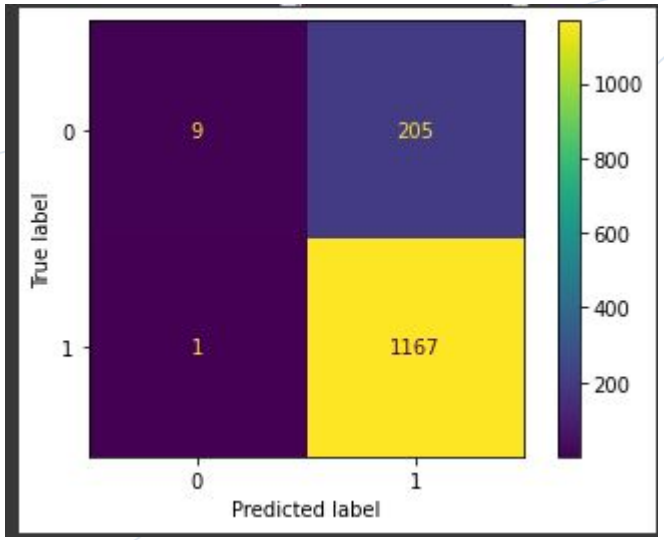
6) Avaliar o modelo:
   a) Accuracy: 0,85
   b) Precision: 0,87
   c) Recall: 0,85
   d) F1-score: 0,78



Quando os dados são desbalanceados, o algoritmo compensa isso com o scale_pos_weight e usa o AUC para avaliar.

Modeling | Classification - XGBoost

```
# split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42, stratify=y)
# Build XGBoost Model
clf_xgb = xgb.XGBClassifier(objective='binary:logistic', missing=None, seed=42)
clf_xgb.fit(X_train,
            y_train,
            verbose=True,
            early_stopping_rounds=10,
            eval_metric='aucpr',
            eval_set=[(X_test,y_test)])
# Plot Confusion Matrix
plot_confusion_matrix(clf_xgb,
                      X_test,
                      y_test,
                      values_format='d',
                      display_labels=["0","1"])
```



Modeling | Classification - XGBoost

Optimize Parameters using Cross Validation and GridSearch(). When data are imbalanced, balance the positive and negative weights via scale_pos_weight and use AUC for evaluation

```
#Round 1
param_grid = {
    'max_depth':[3,4,5],
    'learning_rate': [0.1,0.001,0.05],
    'gamma': [0,0.25,1],
    'reg_lambda': [0,1,10],
    'scale_pos_weight': [1,2,5]
}
optimal_params = GridSearchCV(
    estimator=xgb.XGBClassifier(objective='binary:logistic',
                                seed=42,
                                subsample=0.9,
                                colsample_bytree=0.5),
    param_grid=param_grid,
    scoring='roc_auc',
    verbose=0,
    n_jobs=10,
    cv=5
)
```

```
optimal_params.fit(X_train,
                   y_train,
                   early_stopping_rounds=10,
                   eval_metric='auc',
                   eval_set=[(X_test,y_test)],
                   verbose=False)
print(optimal_params.best_params_)
```

```
{'gamma': 1, 'learning_rate': 0.1, 'max_depth': 4,
'reg_lambda': 1, 'scale_pos_weight': 1}
```

Modeling | Classification - XGBoost

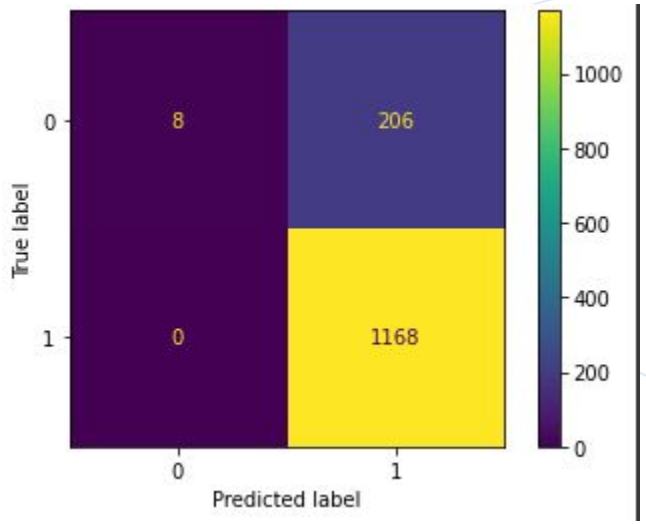
```
#Round 2
param_grid = {
    'max_depth':[4],
    'learning_rate': [0.1],
    'gamma': [1],
    'reg_lambda': [1],
    'scale_pos_weight': [1]
}
optimal_params = GridSearchCV(
    estimator=xgb.XGBClassifier(objective='binary:logistic',
                                seed=42,
                                subsample=0.9,
                                colsample_bytree=0.5),
    param_grid=param_grid,
    scoring='roc_auc',
    verbose=0,
    n_jobs=10,
    cv=5
)
```

. . .
. . .
. . .
. . .

```
optimal_params.fit(X_train,
                   y_train,
                   early_stopping_rounds=10,
                   eval_metric='auc',
                   eval_set=[(X_test,y_test)],
                   verbose=False)
print(optimal_params.best_params)
{'gamma': 1, 'learning_rate': 0.1, 'max_depth': 4, 'reg_lambda': 1, 'scale_pos_weight': 1}
#Train out model
clf_xgb= xgb.XGBClassifier(seed=42,
                           objective='binary:logistic',
                           gamma=1,
                           learn_rate=0.1,
                           max_depth=4,
                           reg_lambda=1,
                           scale_pos_weigth=3,
                           subsample=0.9,
                           colsample_bytree=0.5)
clf_xgb.fit(X_train,
        y_train,
        verbose=True,
        early_stopping_rounds=10,
        eval_metric='aucpr',
        eval_set=[(X_test,y_test)])
```

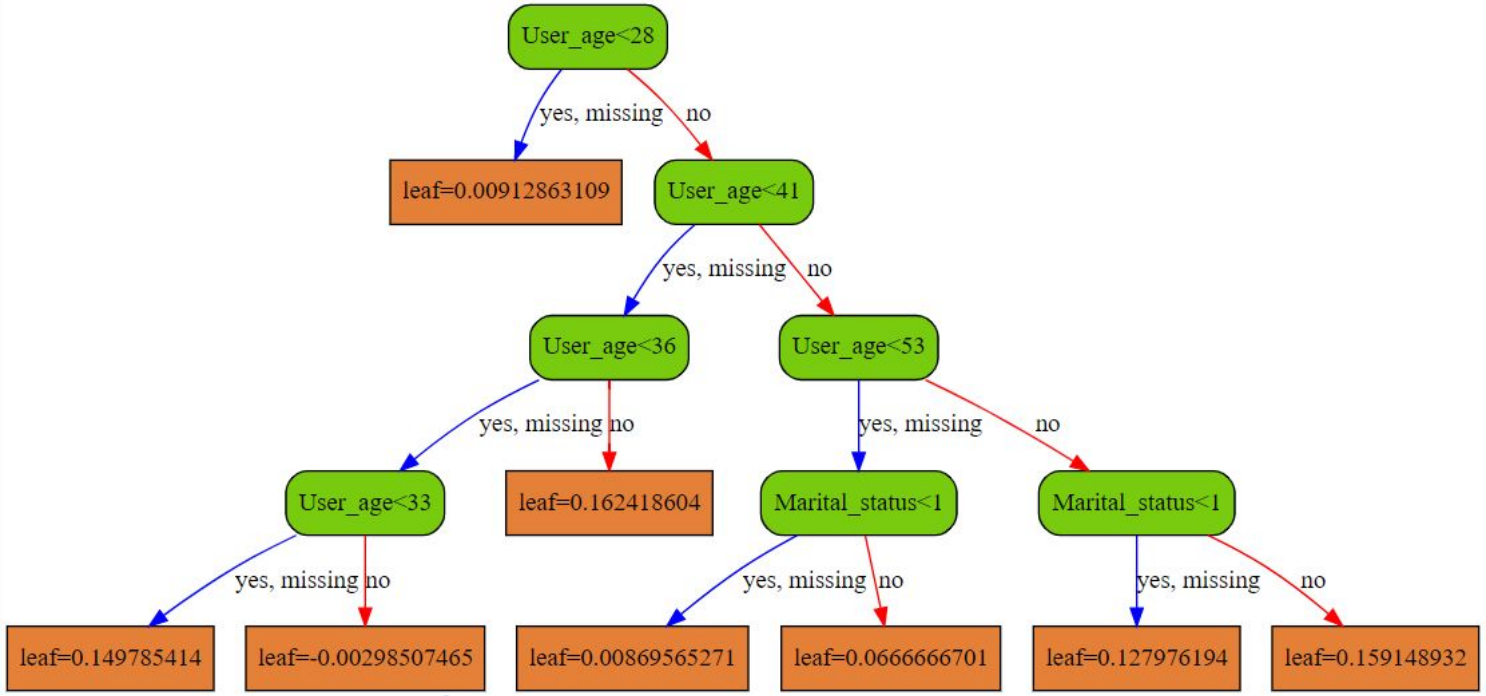
Modeling | Classification - XGBoost

```
plot_confusion_matrix(clf_xgb,
                      X_test,
                      y_test,
                      values_format='d',
                      display_labels=["0","1"])
```



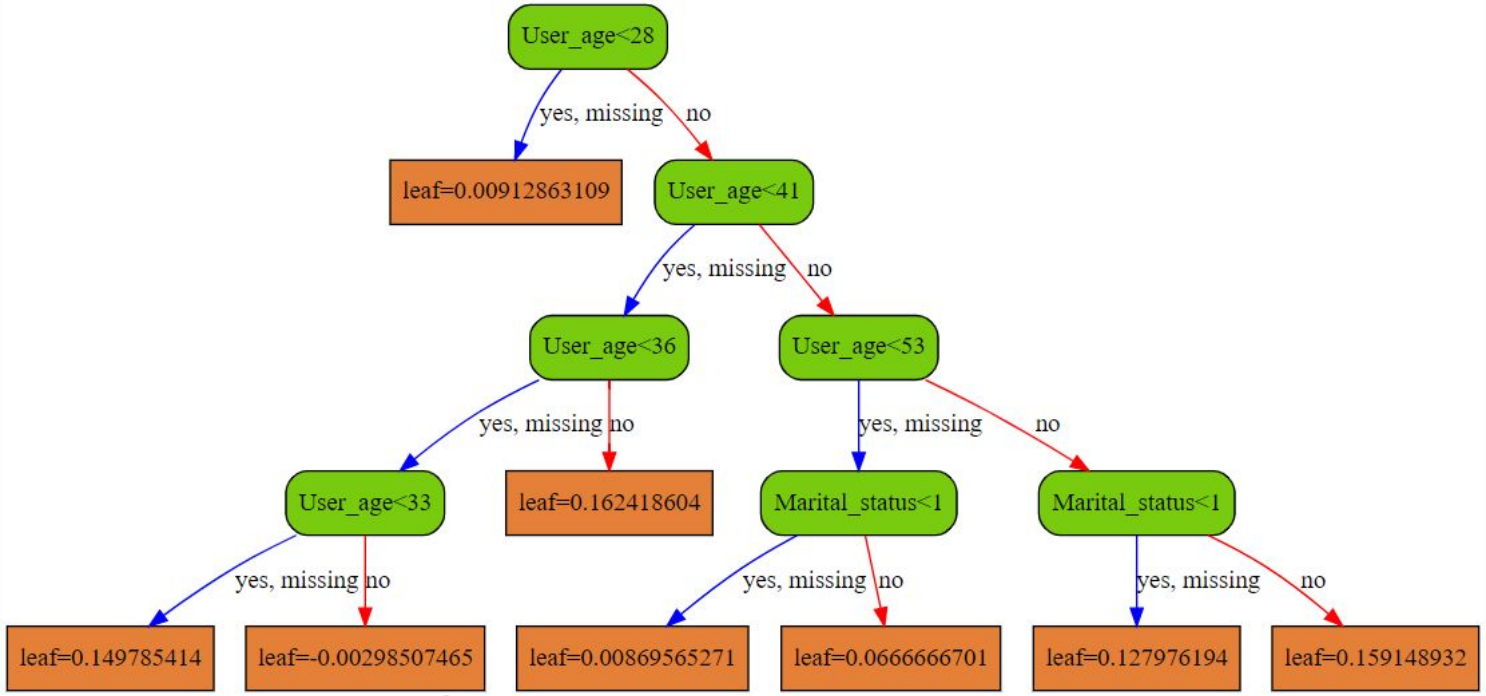

```
#Plot 1 tree
clf_xgb= xgb.XGBClassifier(seed=42,
                           objective='binary:logistic',
                           gamma=1,
                           learn_rate=0.1,
                           max_depth=4,
                           reg_lambda=1,
                           scale_pos_weigth=3,
                           subsample=0.9,
                           colsample_bytree=0.5,
                           n_estimators=1)
clf_xgb.fit(X_train,y_train)
bst = clf_xgb.get_booster()
for importance_type in ('weight', 'gain', 'cover', 'total_gain', 'total_cover'):
  print('%s: '% importance_type, bst.get_score(importance_type=importance_type))
node_params = {'shape': 'box',
               'style': 'filled, rounded',
               'fillcolor': '#78cbe'}
leaf_params ={'shape':'box',
              'style': 'filled',
              'fillcolor': '#e48038'}
xgb.to_graphviz(clf_xgb, num_trees=0, size='50,50',
                condition_node_params=node_params,
                leaf_node_params=leaf_params)
```

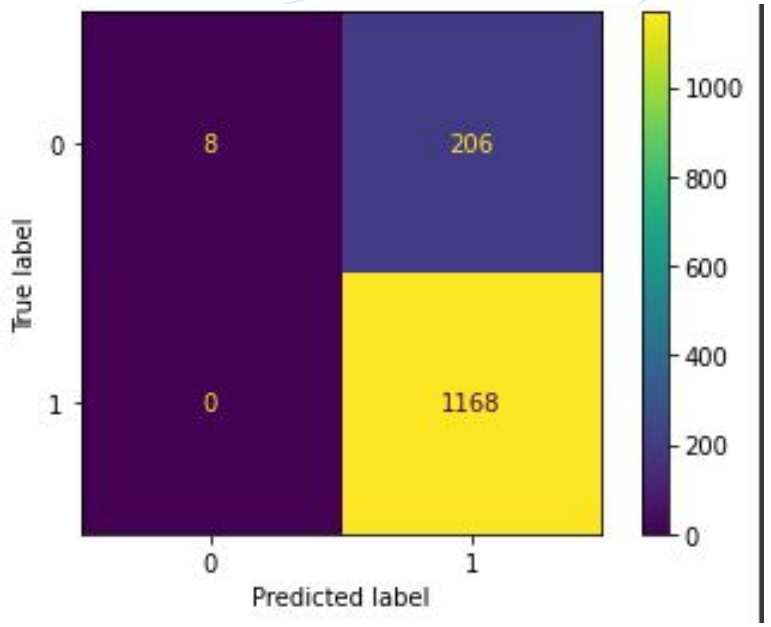
Modeling | Classification - XGBoost




```
y_pred = clf_xgb.predict(X_test)
# evaluate the classifier using various metrics
acc = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred,
average='weighted')
rec = recall_score(y_test, y_pred,
average='weighted')
f1 = f1_score(y_test, y_pred,
average='weighted')

# print the results
print("Accuracy: ", acc)
print("Precision: ", prec)
print("Recall: ", rec)
print("F1-Score: ", f1)
Accuracy:  0.8509406657018813
Precision:  0.8732887172778729
Recall:  0.8509406657018813
F1-Score:  0.7878222824258228
```

Modeling | Classification - XGBoost

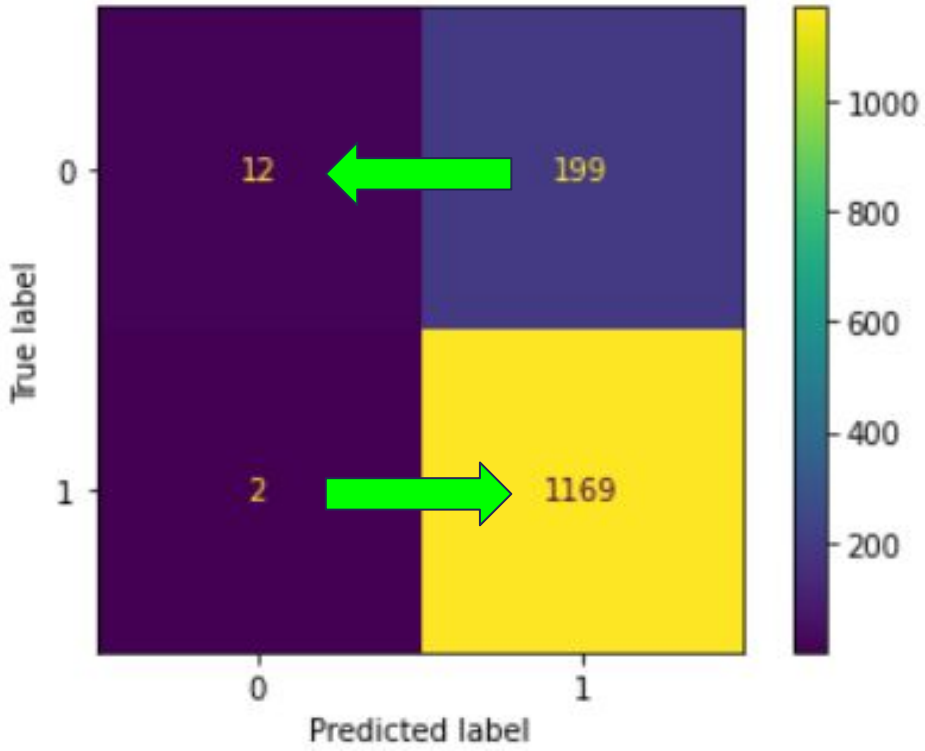




Modeling | Classification - Comparação dos modelos

Error	Classification Tree	SVM	XGBoost
Accuracy	0.74	0.85	0.85
Precision	0.82	0.85	0.87
Recall	0.74	0.85	0.85
F1-score	0.77	0.8	0.78

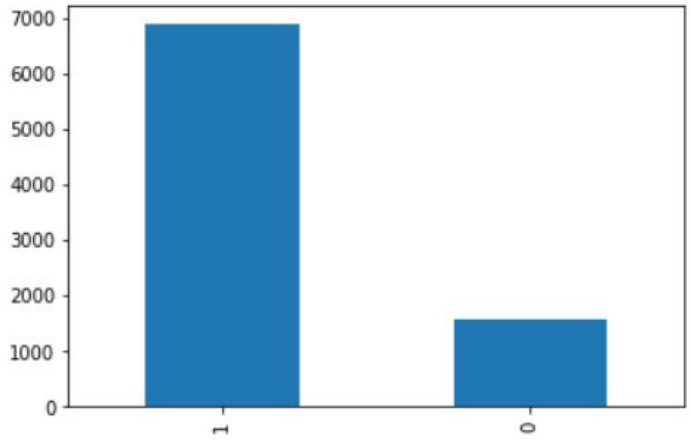
Modeling | Classification - Support Vector Machine



	precision	recall	f1-score	support
0	0.86	0.06	0.11	211
1	0.85	1.00	0.92	1171
accuracy			0.85	1382
macro avg	0.86	0.53	0.51	1382
weighted avg	0.85	0.85	0.80	1382

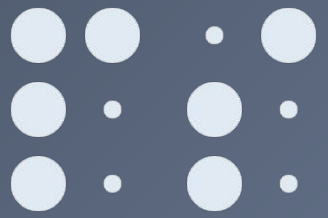
- O modelo é mais preciso que o threshold (85%>81%)
- 2 lugares de estacionamento a menos.
- 199 lugares de estacionamento a mais.

User_garage



1: Quer lugar de garagem (81%)

0: Não quer lugar de garagem (19%)



5. Recommender System

Porto Business School / University of Porto

Recomendação:

Recomendar um projeto a um utilizador usando *collaborative filtering*.

Target: ID de Projeto

user	proj1	proj2	proj3	proj4	proj5
1	1	0	0	1	0
2	1	0	1	1	0
3	1	0	0	1	0
4	1	1	0	1	0

Recommender System - Similar Products

	Project_Id	Corop_Naam	Project_House_LotAreaFrom	Project_House_LotAreaTo	Project_House_LivingAreaFrom	Project_House_LivingAreaTo	Project_House_BedroomsFrom	Project_House_BedroomsTo
0	9395	Noord-Overijssel	81	107	140	185	5	7
1	9404	Zuidoost-Friesland	131	539	126	135	4	4
2	9431	Arnhem/Nijmegen	121	231	118	161	4	5
3	9508	Zuidwest-Drenthe	244	317	131	137	4	4
4	9592	Overig Groningen	87	122	110	179	4	5
...	...	...	...	...	...	...	...	...
62	13800	Arnhem/Nijmegen	28	38	85	89	3	3
63	14714	Noord-Overijssel	142	424	124	175	4	4
64	14715	Noord-Overijssel	136	581	124	189	4	4
65	14726	Zuidoost-Friesland	125	955	122	185	4	5
66	14731	Veluwe	89	285	91	158	2	5

67 rows × 8 columns

```
[156] location_df = pd.get_dummies(df['Corop_Naam'])
```

```
location_df
```

	Arnhem/Nijmegen	Noord-Overijssel	Overig Groningen	Twente	Veluwe	Zuidoost-Friesland	Zuidwest-Drenthe	Zuidwest-Friesland
0	0	1	0	0	0	0	0	0
1	0	0	0	0	0	1	0	0
2	1	0	0	0	0	0	0	0
3	0	0	0	0	0	0	1	0
4	0	0	1	0	0	0	0	0
...	...	...	...	...	...	...	...	...
62	1	0	0	0	0	0	0	0
63	0	1	0	0	0	0	0	0
64	0	1	0	0	0	0	0	0
65	0	0	0	0	0	1	0	0
66	0	0	0	0	1	0	0	0

67 rows × 8 columns

Recommender System - Similar Products

```python
def recommend_similar_projects(vectors, project_id, k):
    # Get the index of the input project in the dataframe
    project_index = df[df['Project_Id'] == project_id].index[0]

    # Get the similarity between projects
    similarities = cosine_similarity(vectors)

    # Get the top k similar projects
    top_k = np.argsort(similarities[project_index])[-(k+1):-1][::-1]

    columns_to_keep = ['Project_Id', 'Corop_Naam', 'Project_House_LotAreaFrom', 'P
    # Get the project IDs of the most similar projects
    similar_project_ids = df[columns_to_keep].iloc[top_k]

    return similar_project_ids
```

1) Transformação do dataframe anterior em vectores
2) Cálculo de similaridade entre projectos
3) Recomendar os top k projectos mais similares

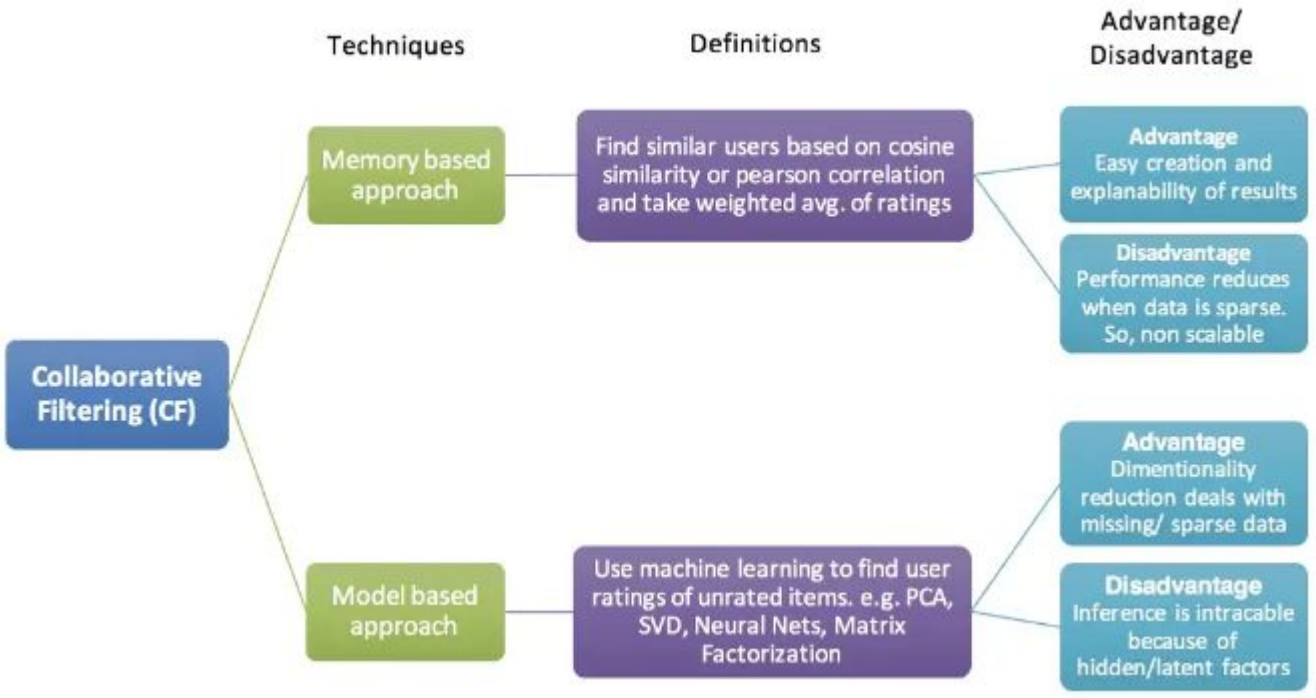
```python
recommend_similar_projects(vectors, 14714, 5)
```

	Project_Id	Corop_Naam	Project_House_LotAreaFrom	Project_House_LotAreaTo	Project_House_LivingAreaFrom	Project_House_LivingAreaTo	Project_House_BedroomsFrom	Project_House_BedroomsTo
41	11733	Arnhem/Nijmegen	140	401	118	163	4	6
32	11502	Noord-Overijssel	124	372	103	157	3	5
47	12187	Arnhem/Nijmegen	122	365	120	168	3	4
21	10471	Noord-Overijssel	145	423	137	155	3	4
17	10362	Zuidwest-Friesland	156	388	114	170	3	5

Recommender System - Similar Products - Next Steps

1) A distância ser um factor a ter em conta no cálculo das similaridades

Recommender System - Collaborative Filtering



Recommender System - Collaborative Filtering - Memory Based Approach

	date	user	item
0	2022-10-26	591845	15174
1	2022-10-24	714137	14937
2	2022-10-23	397221	15174
3	2022-10-19	285850	14891
4	2022-10-18	219494	14937
...	...	...	...
62942	2016-06-30	107232	9431
62943	2015-09-17	281718	9431
62944	2015-08-18	341882	9431
62945	2015-07-07	28479	9431
62946	2014-04-04	341888	9431

62947 rows × 3 columns

1) Rating de 1 se projeto foi subscrito

Para o futuro:

2) Rating ser em função do número de vezes que o projeto foi visto
3) Rating ser função que decai em função da última data de visualização de um projecto por parte do utilizador

Recommender System - Collaborative Filtering - Memory Based Approach

Porto Business School / University of Porto

```python
#matrix.py
import pandas
from scipy import sparse as sp
from typing import List


def table_to_sparse_matrix(ratings: pandas.DataFrame, x : str, y : str, v: str) -> (sp.spmatrix, List, List):
    assert ratings[x].dtype == 'category' and ratings[y].dtype == 'category'

    i = ratings[x].cat.codes
    j = ratings[y].cat.codes
    data = ratings[v].values

    users = ratings[x].cat.categories.values
    items = ratings[y].cat.categories.values

    mat = sp.coo_matrix((data, (i, j)), dtype='float32')
    mat = mat.tocsr()
    return mat, list(users), list(items)
```

Recommender System - Collaborative Filtering - Memory Based Approach

Porto Business School / University of Porto

```python
#algo __init__.py
import numpy as np

from scipy.sparse import spmatrix, csr_matrix
from scipy.spatial.distance import cosine
from typing import List


def similarity(v1: spmatrix, v2: spmatrix) -> float:
    """
        >>> R = csr_matrix([[1,0,1],[1,1,0]])
        >>> similarity(R[0],R[1])
        0.5
    """
    assert v1.shape == v2.shape
    return 1 - cosine(v1.A.flatten(), v2.A.flatten())


def get_items_from_user(i: int, R: spmatrix) -> List:
    """
    >>> R = csr_matrix([[1,0,0],[0,1,0],[1,1,0]])
    >>> get_items_from_user(0, R)
    [0]
    >>> get_items_from_user(2, R)
    [0, 1]
    """
    assert isinstance(R, csr_matrix)
    scores = R[i].data
    idxs = R[i].indices
    return idxs[np.argsort(-scores)]
```

Recommender System - Collaborative Filtering - Memory Based Approach

Porto Business School / University of Porto

```python
def recommend_user_user_cf(i: int, R: spmatrix, min_score=0.5) -> Iterator[int]:
    "recommend_user_user_cf products from similar users that the user never saw"

    for j in range(R.shape[0]):
        if similarity(R[i], R[j]) > min_score:
            for product in get_items_from_user(j, R):
                yield product


def related(k: int, R: spmatrix, min_score=0.5) -> Iterator[int]:
    "show related products based on co-occurency"

    for l in range(R.shape[1]):
        score = similarity(R[:, k], R[:, l])
        if score > min_score:
            yield l


def recommend_item_item_cf(i: int, R: spmatrix) -> Iterator[int]:
    "Recommend things liked by people who rate highly the same things as you"

    for product in get_items_from_user(i, R):
        for other_product in related(product, R):
            yield other_product
```

Recommender System - Collaborative Filtering - Memory Based Approach

Porto Business School / University of Porto

User User Collaborative Filtering

```
--------------------------------
User-User CF for '334493'
--------------------------------
9431
10487
11067
9997
```

Item Item Collaborative Filtering

```
--------------------------------
Item-Item CF for user '341937'
--------------------------------
9431
9997
15077
```

Recommender System - Collaborative Filtering - Model Based Approach

Porto Business School / University of Porto

	User_Id	Project_Id
0	48	14740
1	48	14820
2	48	14891
3	48	14937
4	48	15041
...	...	...
62942	1440619	11209
62943	1440619	15078
62944	1442635	12455
62945	1450708	11209
62946	1450708	15080

62947 rows × 3 columns

1) Divisão dos dados entre treino e teste (75 / 25)

2) Fit do modelo

```
from rankfm.rankfm import RankFM


model = RankFM(factors=50, loss='warp', max_samples=100, learning_schedule='invscaling')
model.fit(interactions_train, sample_weight=sample_weight_train, epochs=40, verbose=True)
```

```
Hit Rate: 0.8856169052488071
Precision: 0.12000000000000002
Recall: 0.8500992039624302
F1 Score: 0.21031231457322655
Discounted Cumulative Gain: 0.6683037868757044
Reciprocal Rank: 0.4265315242206858
```

6. Conclusão

Conclusões

Porto Business School / University of Porto

- Regressão:
  - A localização projeto e idade do user são as variáveis com mais impacto sobre o preço que o user estás disposto a pagar por uma casa;
  - Seria interessante ter mais variaveis relativas ao apartamento que suscitou o interesse do user, em vez do projeto;
  - O salário anual do user ajudaria a fortalecer o algoritmo, infelizmente esta pergunta foi respondida poucas vezes.

- Classificação:
  - Poderiamos ter abordado este problema dos lugares de garagem como um problema multiclasse. Seria interessante saber quantos lugares de garagem o user deseja;
  - Introduzir a localização exata do projeto poderia melhorar o algoritmo, no sentido que consideraria se na envolvência do projeto existem lugares de parqueamento.

Sugestões para a empresa

Dados sobre os apartamentos poderiam enriquecer tanto os algoritmos de previsão como os de recomendação.

Trabalhos Futuros

Make change happen