Ciência de Dados - Modelagem

Luiz Celso Gomes-Jr

Agenda

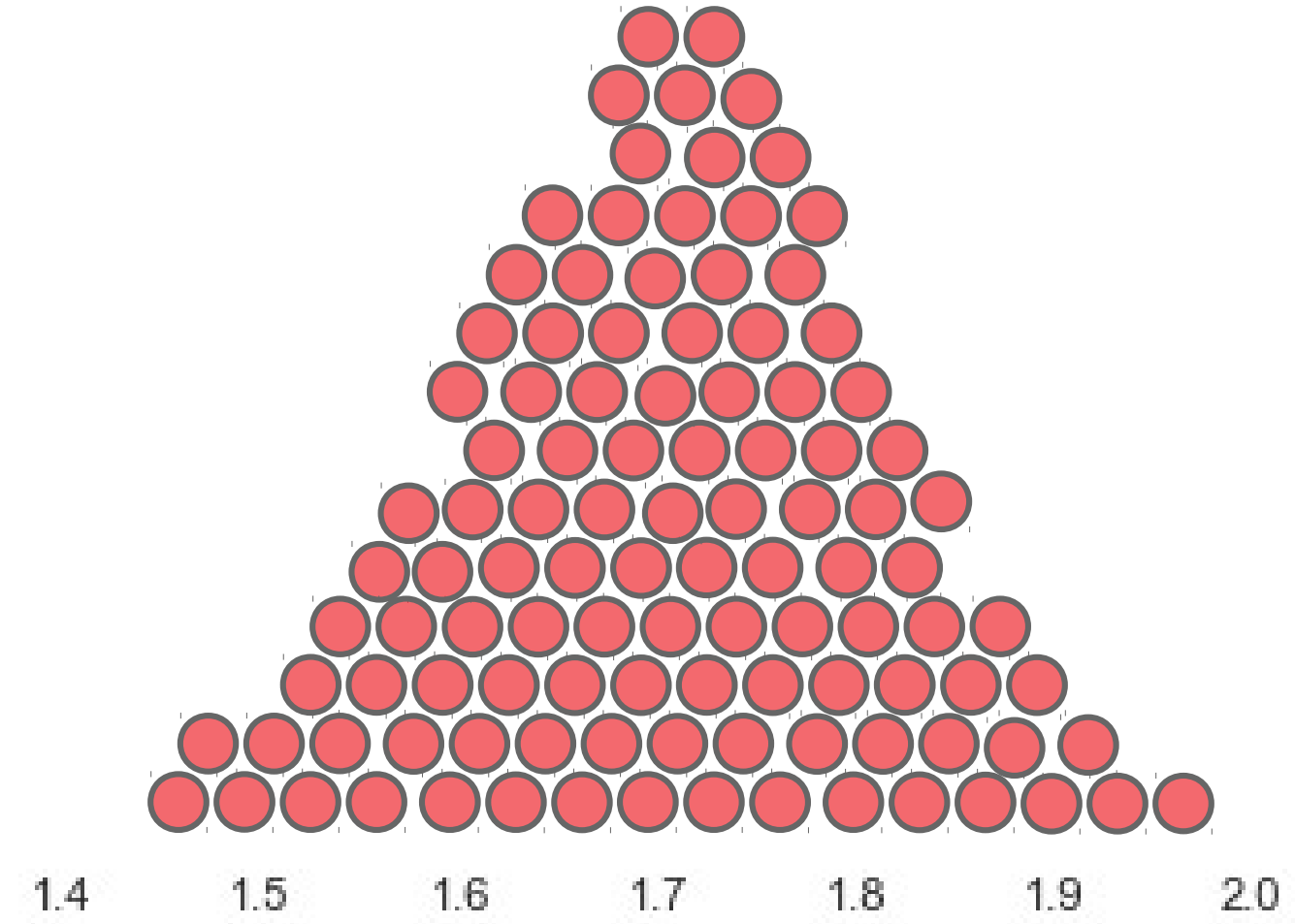
- Modelagem Estatística
- Modelagem baseada em Aprendizado de Máquina
  - Classificação
  - Clusterização
- Modelagem de Grafos
- Modelagem de Texto

Estatística

- A Estatística estuda diversos assuntos relacionados com análise de dados, como coleta, organização, plotagem, análise e interpretação.
- Nesta aula vamos discutir dois tópicos importantes para Ciência de Dados:
  - Intervalos de confiança
  - Regressão linear

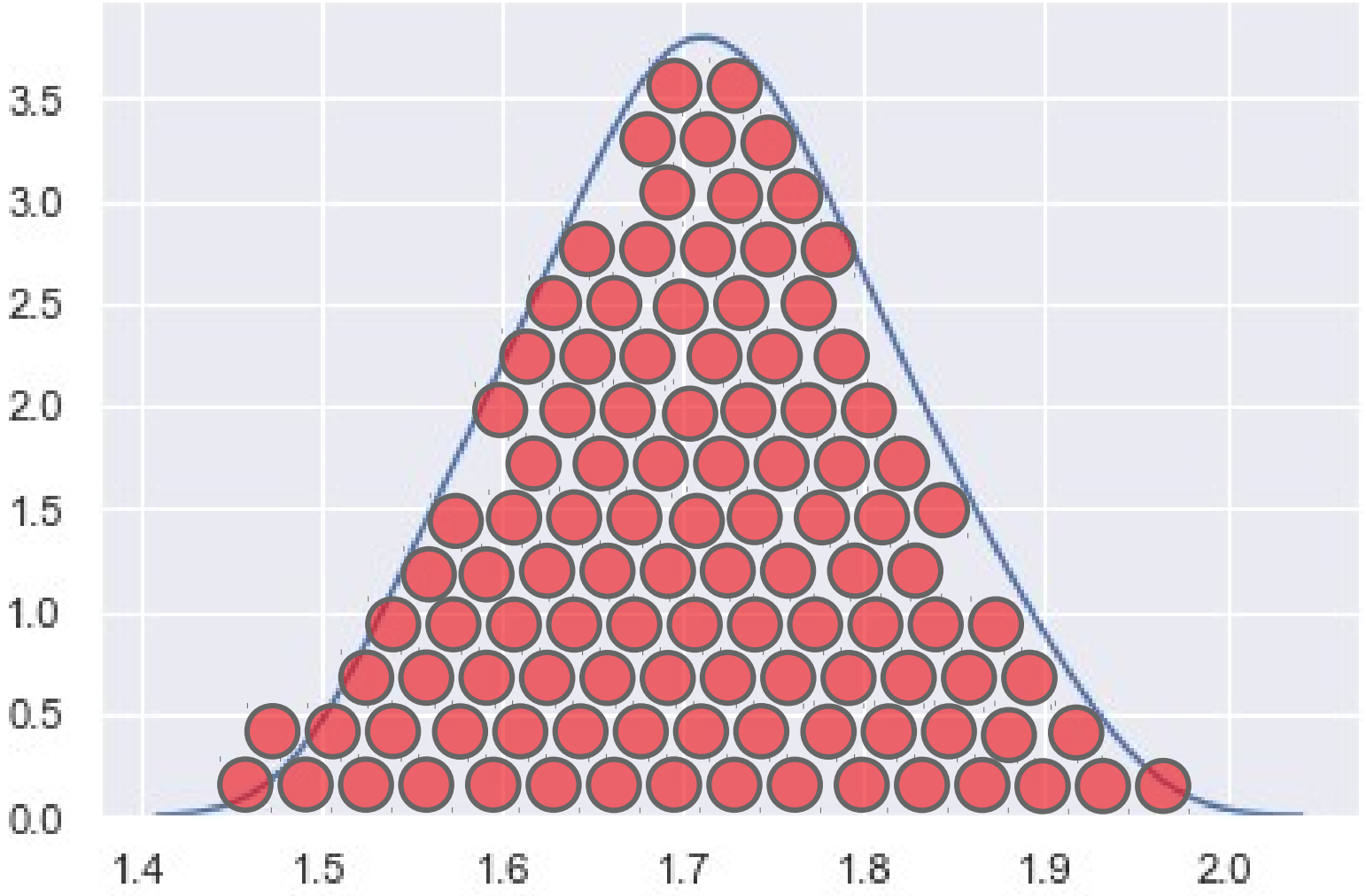
Distribuição Real

Exemplo: distribuição de alturas na sala



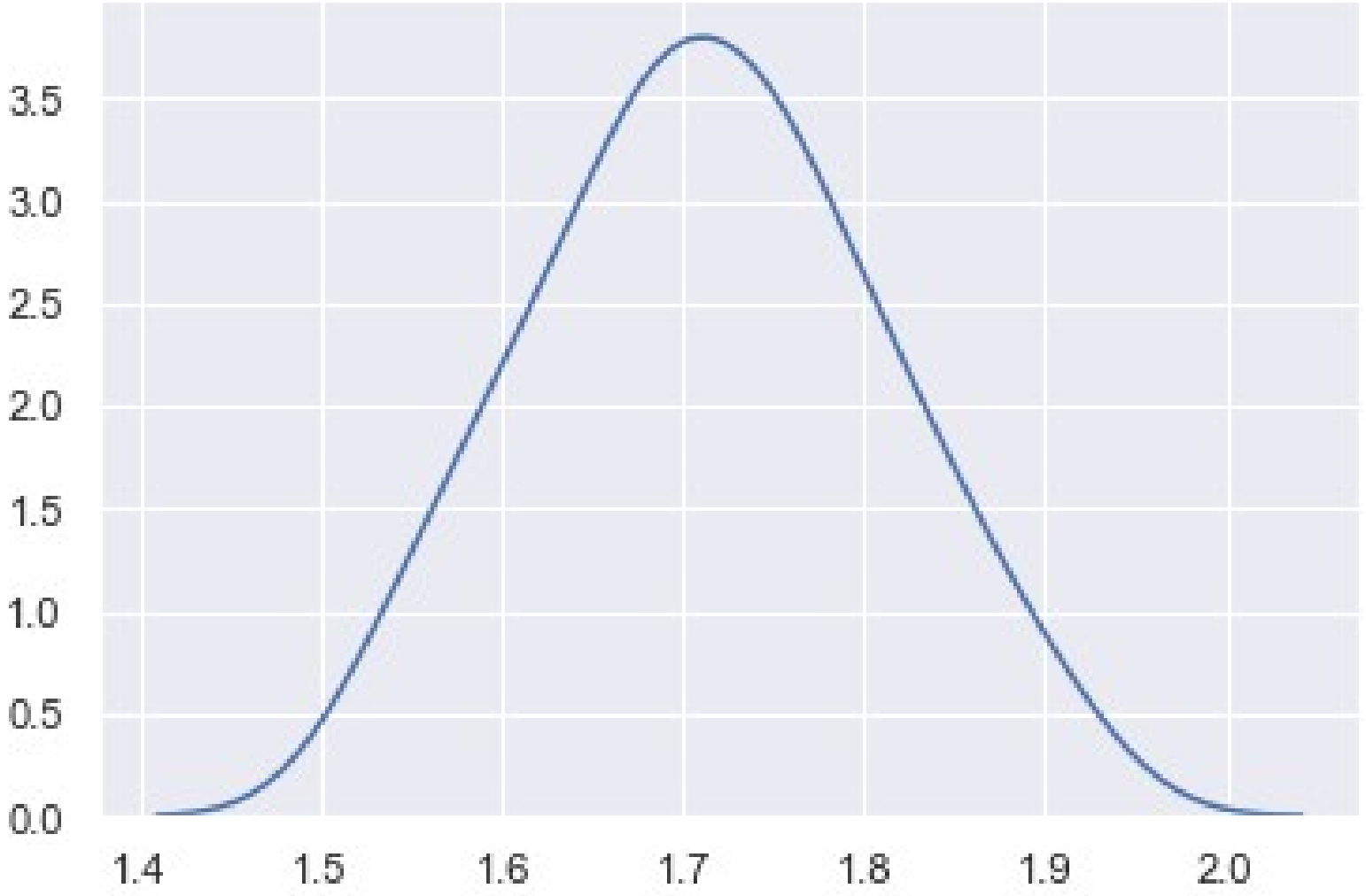
Distribuição Real

Distribuição de alturas na sala é uma distribuição normal.



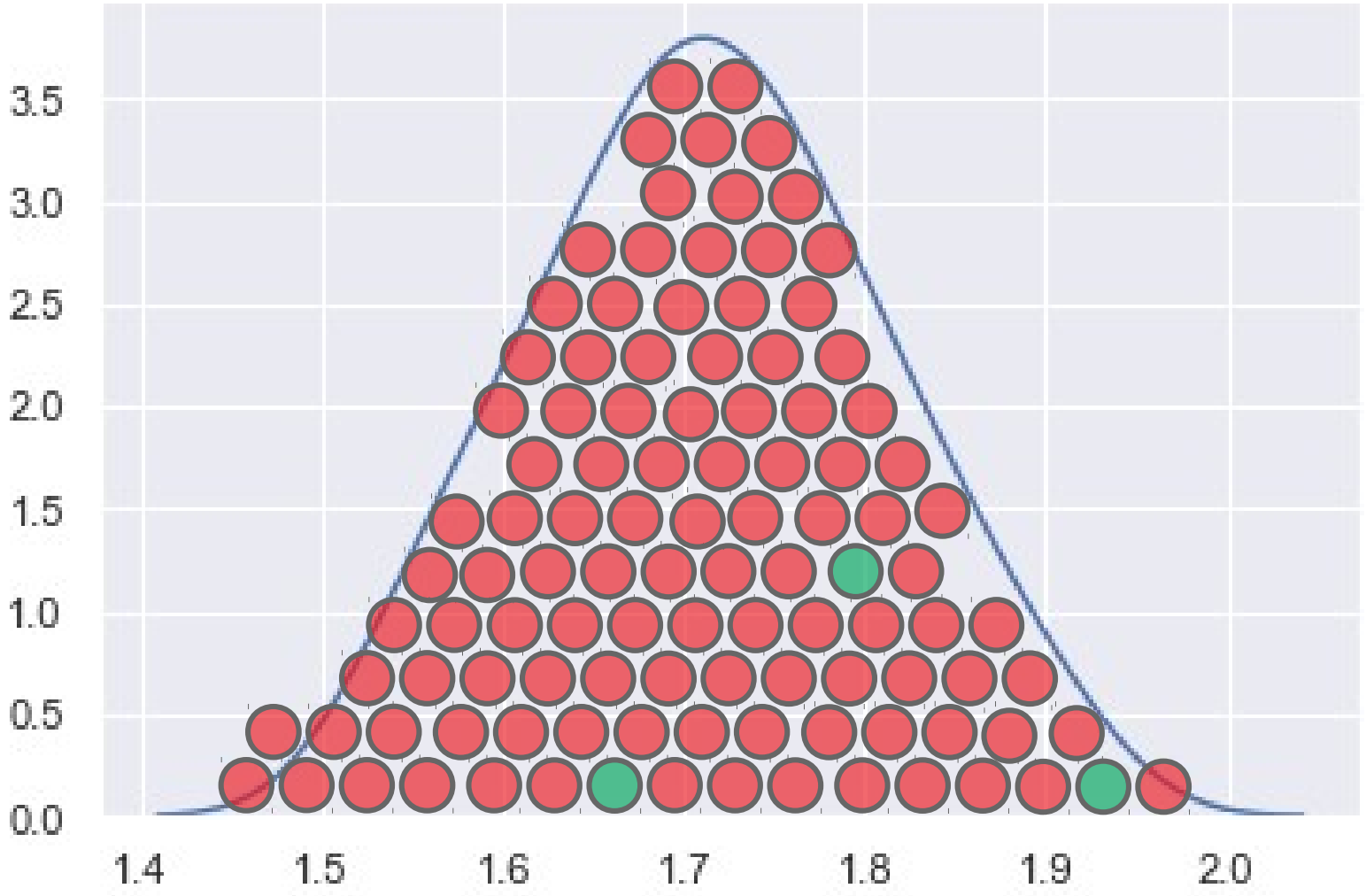
Distribuição Real

Distribuição de alturas na sala é uma distribuição normal.



Estimativa de Valor

Para estimar a média de alturas, podemos fazer uma amostragem. A amostragem abaixo é boa?

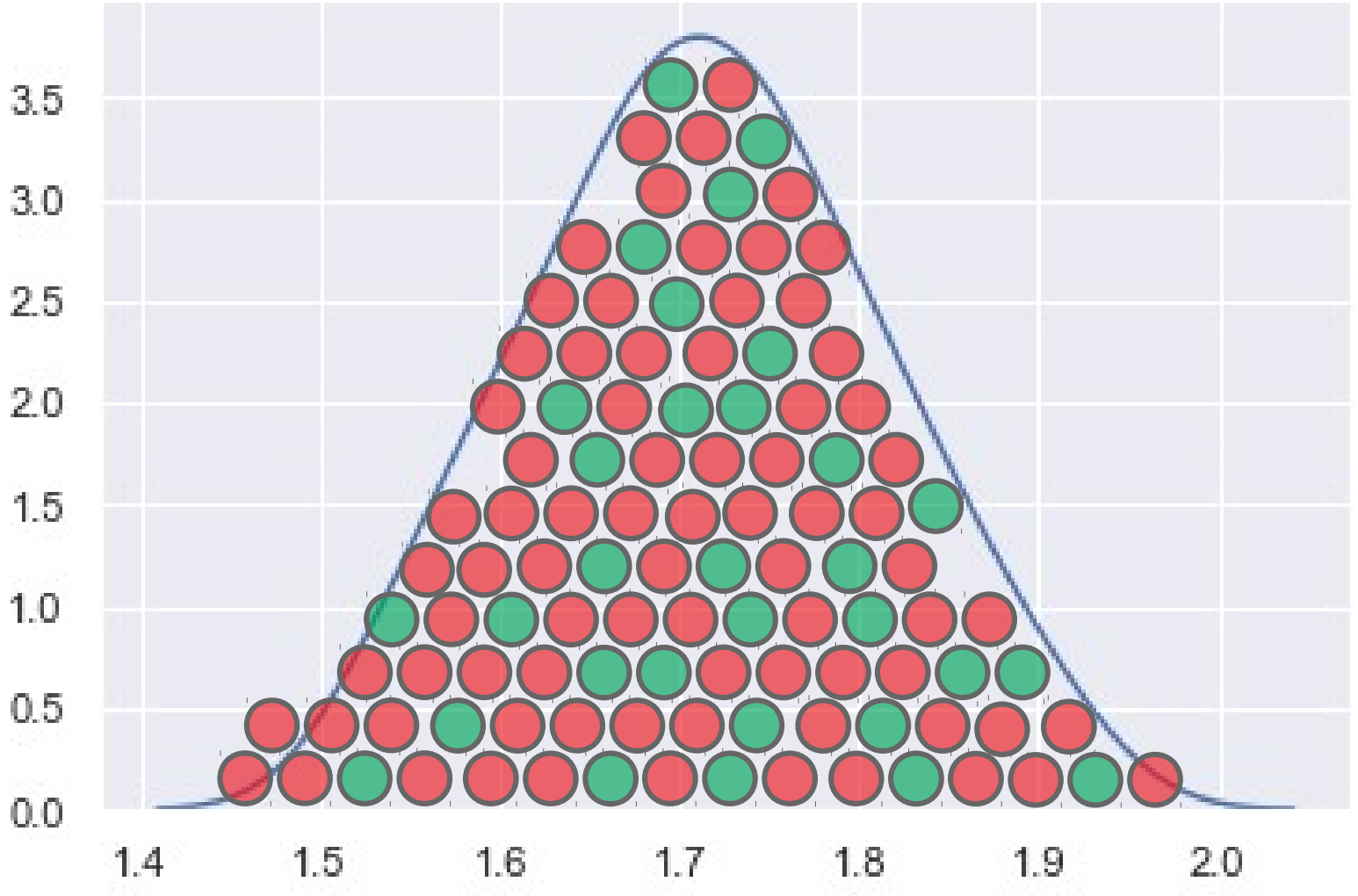


Alturas:
1.67
1.80
1.93

Média:
1.80

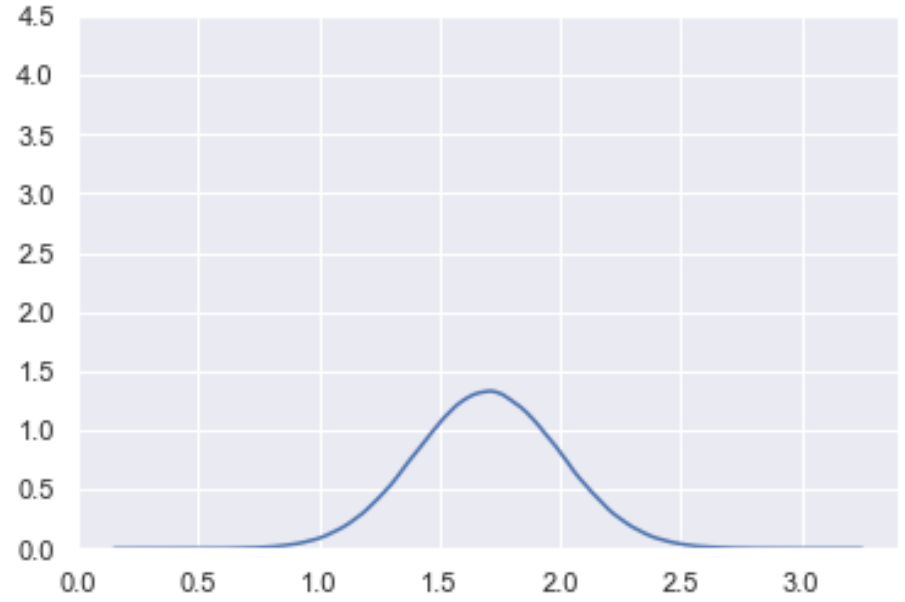
Estimativa de Valor

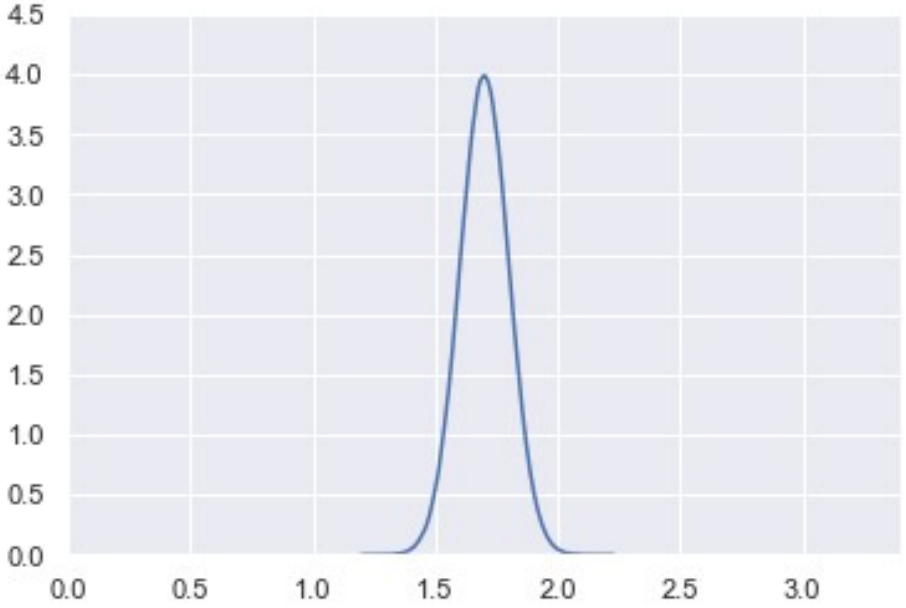
Amostragens com mais elementos são mais confiáveis (diminuem a probabilidade de amostragens tendenciosas)



Distribuição Real

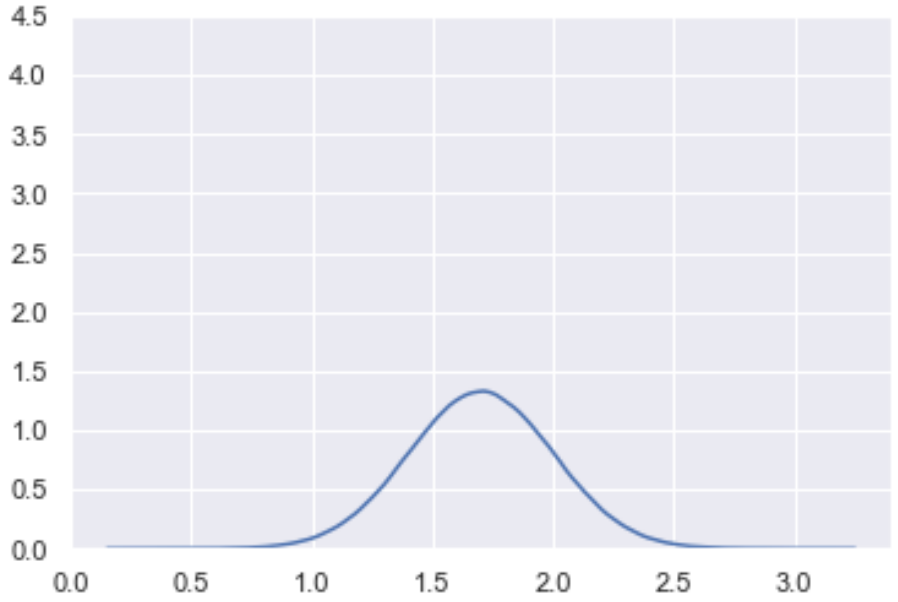
Para qual das distribuições abaixo seria mais fácil estimar precisamente a média a partir de uma amostragem?

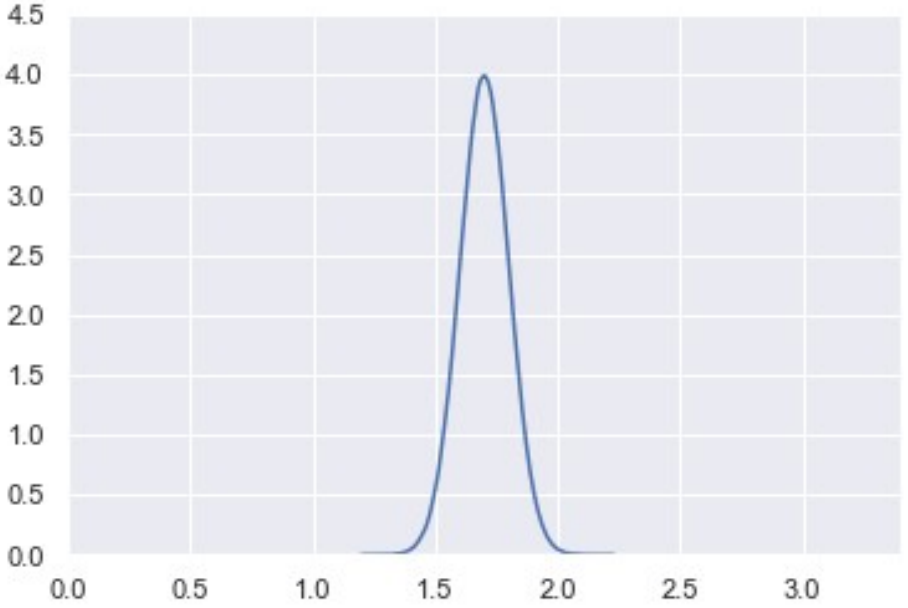




Distribuição Real

Distribuições com menor variabilidade diminuem a probabilidade de amostragens tendenciosas





Intervalo de Confiança

- Conceito importante na inferência estatística
- Considera o número de amostras e a variabilidade da distribuição para oferecer um intervalo de certeza do valor calculado a partir da amostra

Exemplo: Altura média estimada = 1.70m ± 5cm

Biblioteca StatsModels

Cálculo do intervalo de confiança para a média de aluguéis

```
df.head(3)
```

	codigo	endereco	quartos	suite	area	vaga	aluguel	condominio	data
0	34	Rua Desembargador Westphalen	2	0	90	0	900	371	11/10/17
1	167	Rua Jose Loureiro	2	0	64	0	650	428	15/07/17
2	6784	Rua Jose Loureiro	2	0	81	0	1100	400	23/08/17

```
import statsmodels.stats.api as sms

media = df['aluguel'].mean()

intervalo = sms.DescrStatsW(df['aluguel']).tconfint_mean()

print("Média: ", media, "Intervalo: ", intervalo)
```

```
Média:  898.0 Intervalo:  (745.3791042764885, 1050.6208957235115)
```

Biblioteca StatsModels

Podemos também querer saber se existe diferença significativa entre as médias dos valores de aluguéis de 1 e 2 quartos. O método **CompareMeans** fornece estatísticas sobre a diferença dos valores das médias. O resultado indica que, com 95% de certeza, podemos afirmar que a diferença entre as médias é entre -641 e -146.

```
df_1q = df[df['quartos']==1]
df_2q = df[df['quartos']==2]

cm = sms.CompareMeans(sms.DescrStatsW(df_1q['aluguel']),
                      sms.DescrStatsW(df_2q['aluguel']))
cm.summary()
```

Test for equality of means

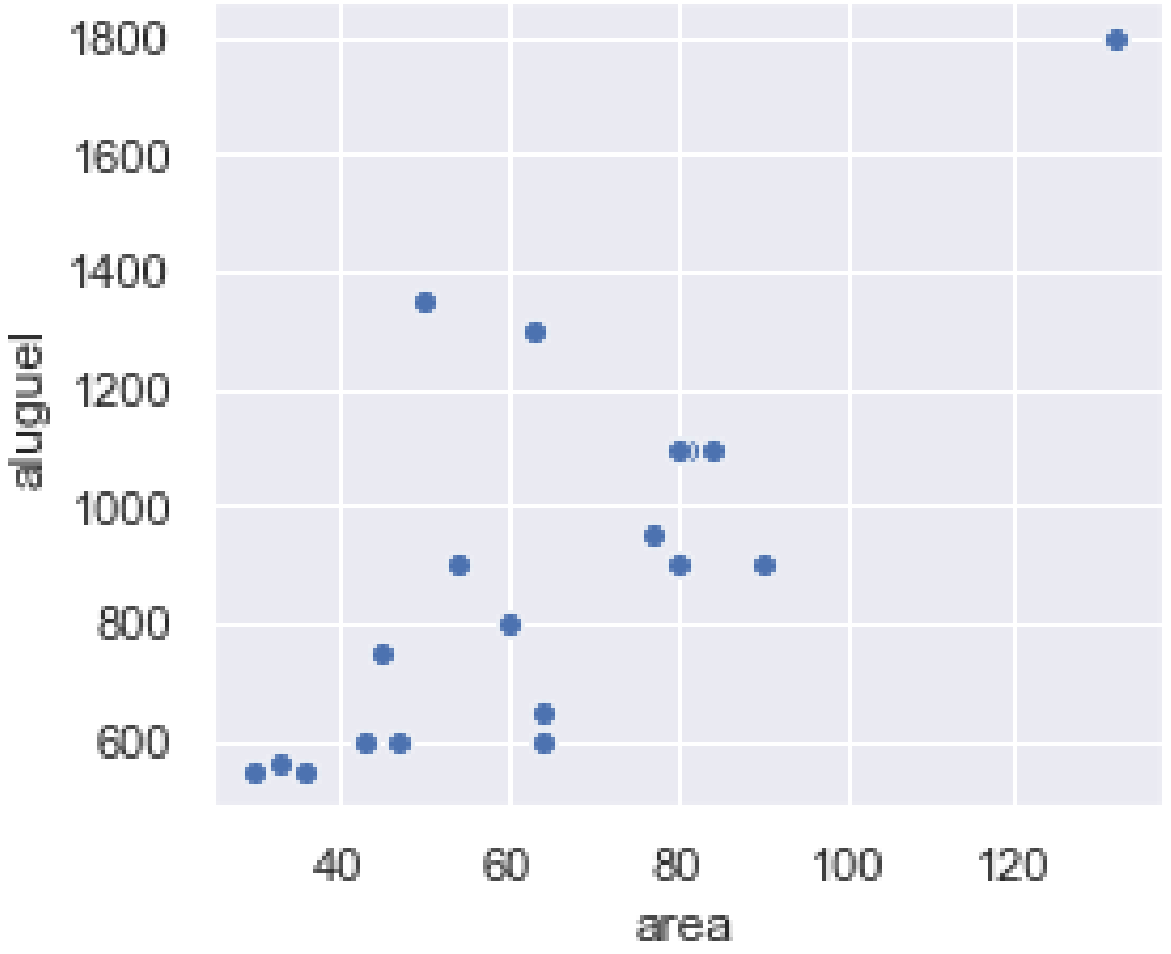
	coef	std err	t	P>|t|	[0.025	0.975]
subset #1	-394.0000	117.584	-3.351	0.004	-641.035	-146.965

Regressão Linear

- Modelos de regressão são muito importantes em Ciência de Dados
- Eles são usados para se entender a influência das variáveis sobre um determinado aspecto ou para prever o comportamento deste aspecto.

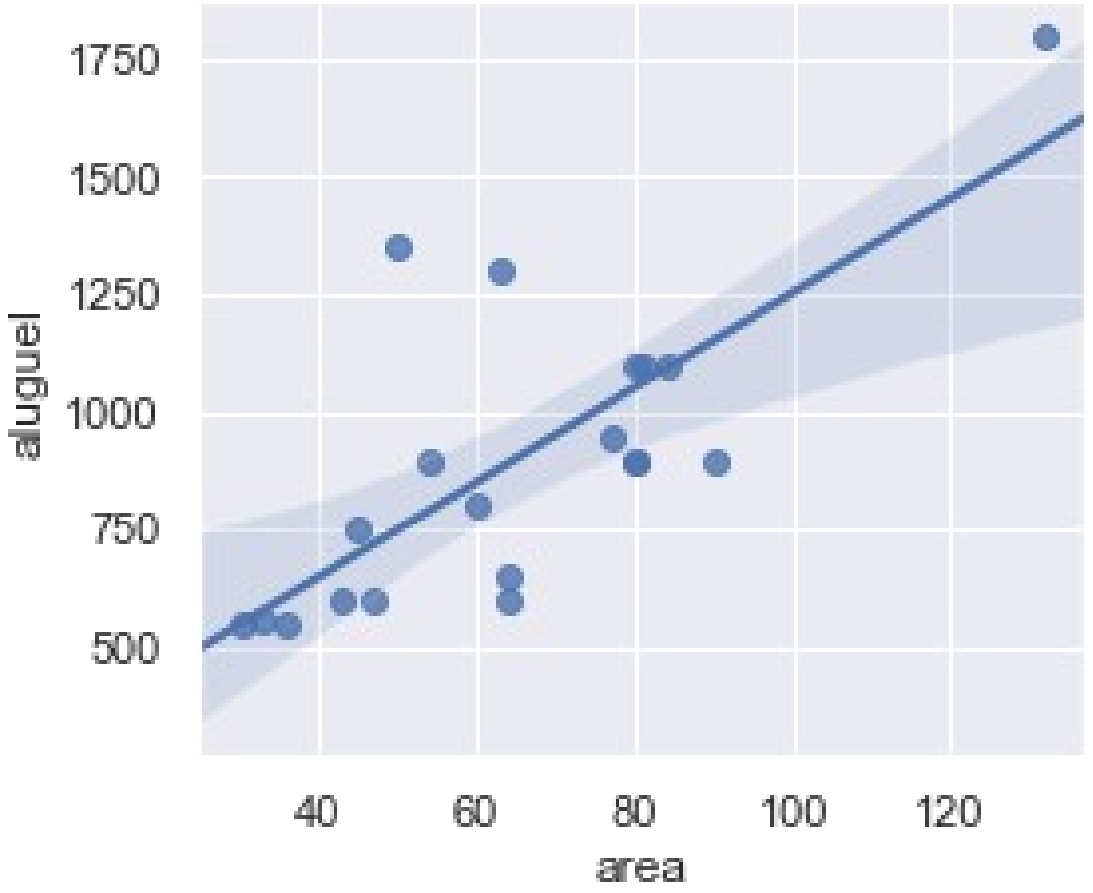
Relação entre variáveis

Por exemplo, podemos querer entender como a variável área influencia no valor do aluguel.



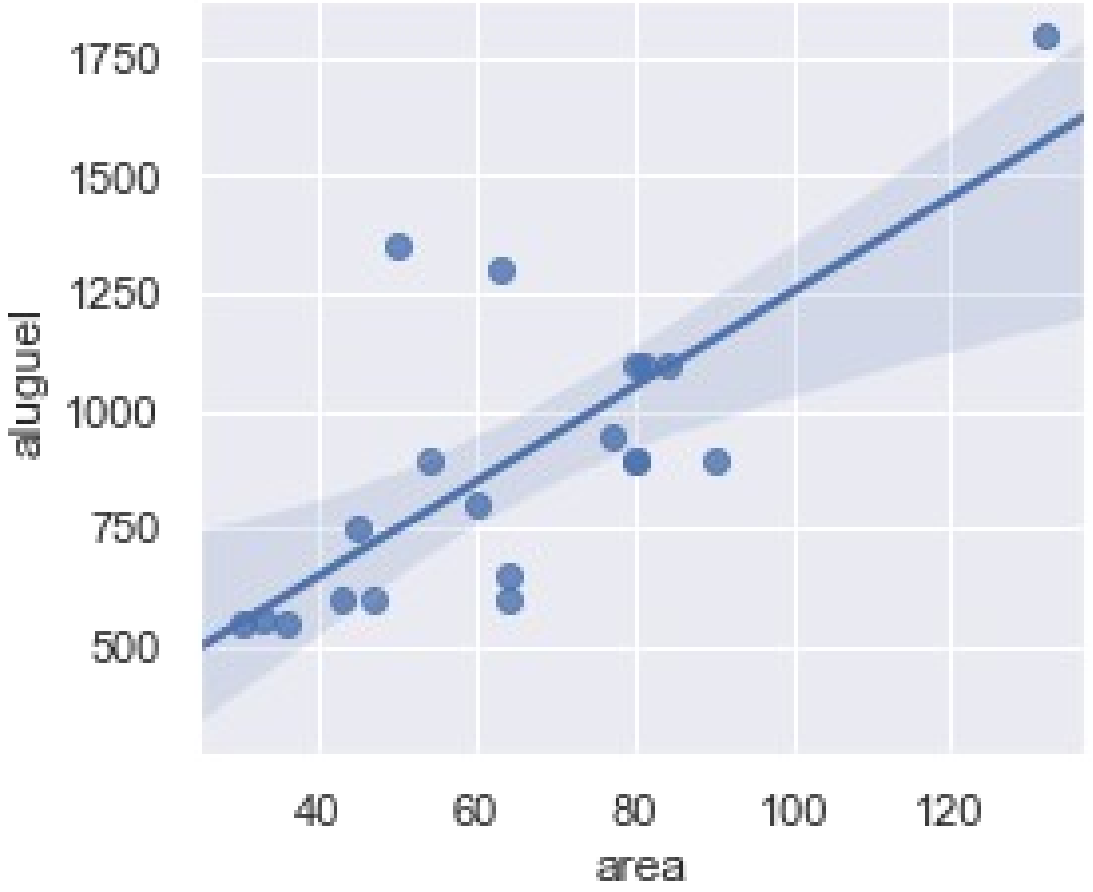
Relação entre variáveis

Modelos de regressão linear consideram que as variáveis possuem uma relação linear e estimam uma reta (função) que representa este relacionamento.



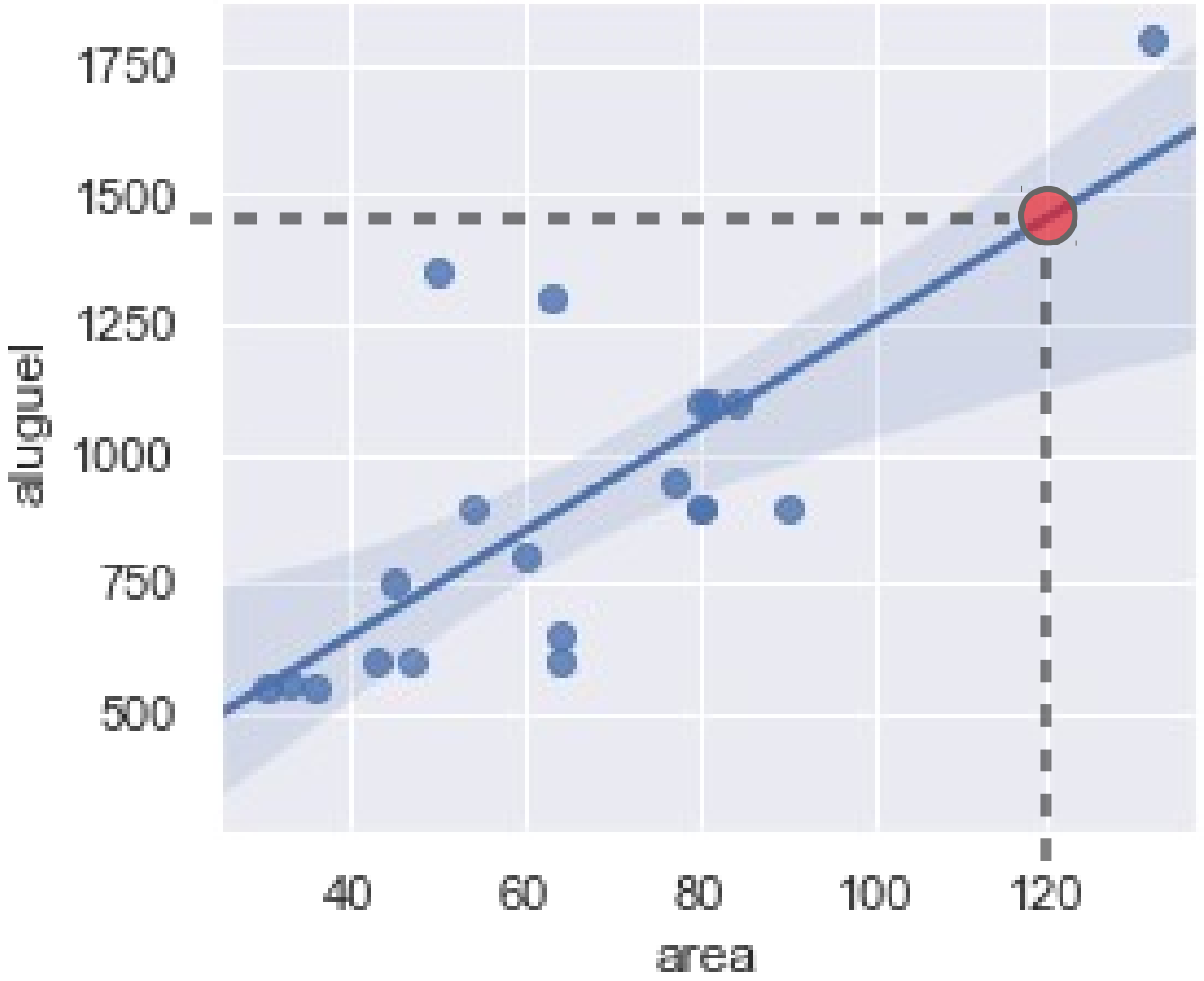
Relação entre variáveis

Modelos de regressão linear podem ser usados para estimar um valor para novas observações. Por exemplo, qual seria o aluguel estimado para um aparamento de área 120m^2?.



Relação entre variáveis

Qual seria o aluguel estimado para um aparamento de área 120m^2?.



Regressão - StatsModels

Abaixo construímos um modelo de regressão linear usando as variáveis quartos, area e vaga para estimar o valor do aluguel.

```
from statsmodels.formula.api import ols

model = ols("aluguel ~ quartos + area + vaga", data=df)
response = model.fit()
response.summary()
```

OLS Regression Results

Dep. Variable:	aluguel	**R-squared:**	0.626
Model:	OLS	**Adj. R-squared:**	0.556
Method:	Least Squares	**F-statistic:**	8.933
Date:	Tue, 17 Sep 2019	**Prob (F-statistic):**	0.00104
Time:	15:05:46	**Log-Likelihood:**	-133.77
No. Observations:	20	**AIC:**	275.5
Df Residuals:	16	**BIC:**	279.5
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	104.7992	166.560	0.629	0.538	-248.293	457.891
quartos	173.0788	138.969	1.245	0.231	-121.522	467.680
area	8.4528	3.448	2.452	0.026	1.144	15.762
vaga	-36.8360	144.643	-0.255	0.802	-343.466	269.794

Omnibus:	2.313	**Durbin-Watson:**	1.468
Prob(Omnibus):	0.315	**Jarque-Bera (JB):**	1.085
Skew:	0.552	**Prob(JB):**	0.581
Kurtosis:	3.288	**Cond. No.**	279.

Regressão - StatsModels

Abaixo usamos o modelo construído para estimar o valor de aluguel de um novo apartamento.

```
novo_apartamento = pd.DataFrame([{'quartos': 2,
                                  'area': 60,
                                  'vaga': 1}])
response.predict(novo_apartamento)
```

```
0    921.290363
dtype: float64
```

Exercícios!

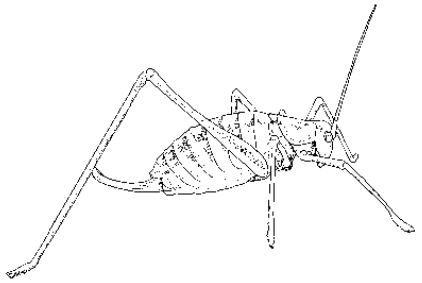
- Revise o conteúdo e faça os exercícios do notebook:
  05a-Modelagem - Estatística.ipynb

Aprendizado de Máquina

- Ampla gama de algoritmos capazes de dar resultados melhores a medida que recebem mais dados
- Base estatística, mas sem ênfase em inferência clássica
- Principais tarefas: Classificação, Clusterização, Regressão, Detecção de Outliers

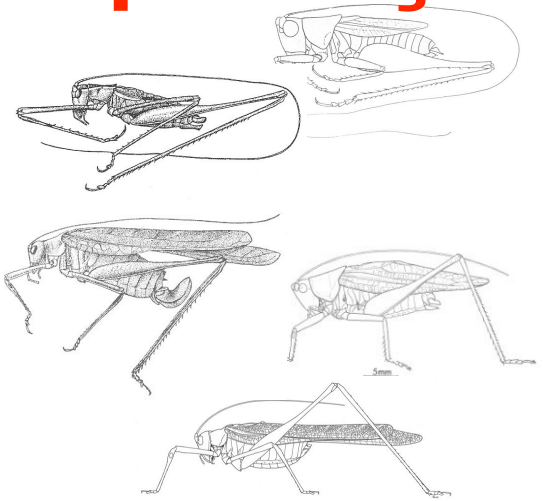
Classificação
(definição informal)

Dada uma coleção com 5 **Esperanças** e 5 **Gafanhotos**, aprender a classificar novas instâncias.

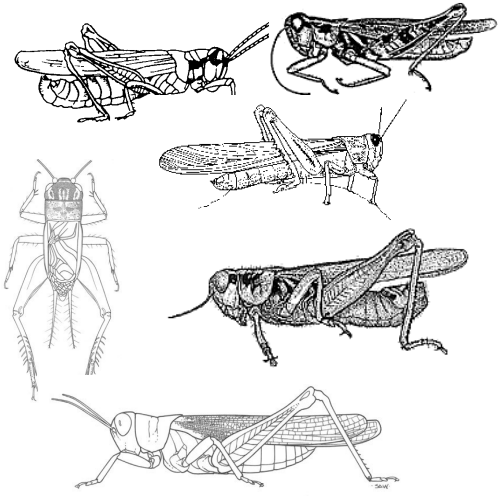


Esperança ou **Gafanhoto**?

Esperanças



Gafanhotos

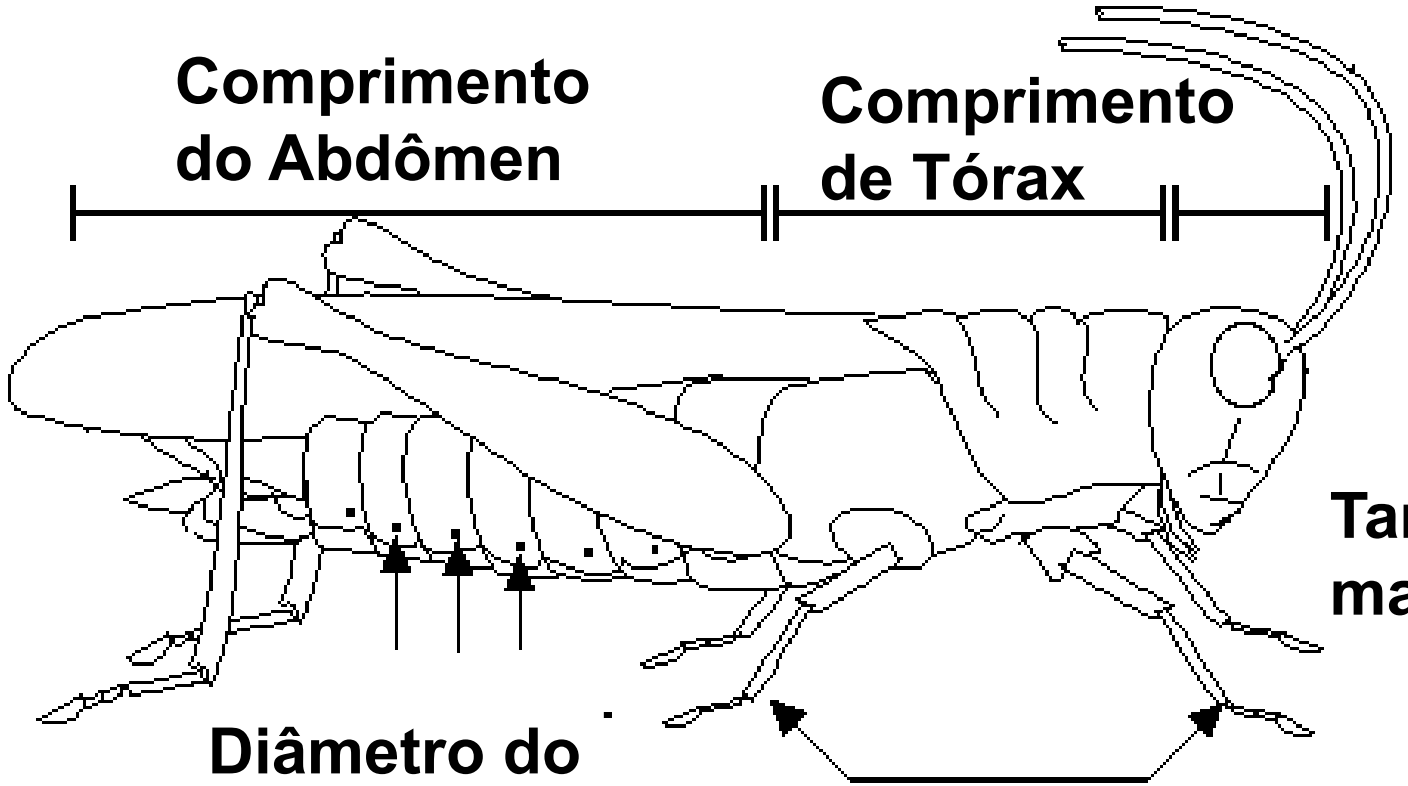


Medindo Características

Também chamadas de *features*, atributos

Cor {Verde, Marrom, Cinza, Outra}

Tem asas?



As características compõem nosso Dataset.

Problema de classificação:

• Dado o Dataset de treinamento, predizer a **classe** de uma nova instância

Dataset

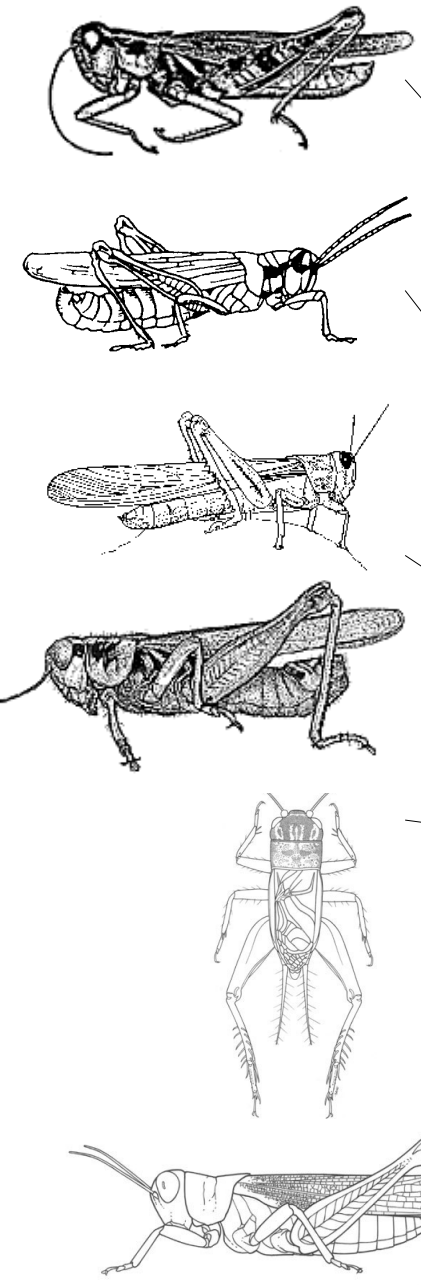
ID	Abdômen	Antena	Classe
1	2.7	5.5	Gafanhoto
2	8.0	9.1	Esperança
3	0.9	4.7	Gafanhoto
4	1.1	3.1	Gafanhoto
5	5.4	8.5	Esperança
6	2.9	1.9	Gafanhoto
7	6.1	6.6	Esperança
8	0.5	1.0	Gafanhoto
9	8.3	6.6	Esperança
10	8.1	4.7	Esperança

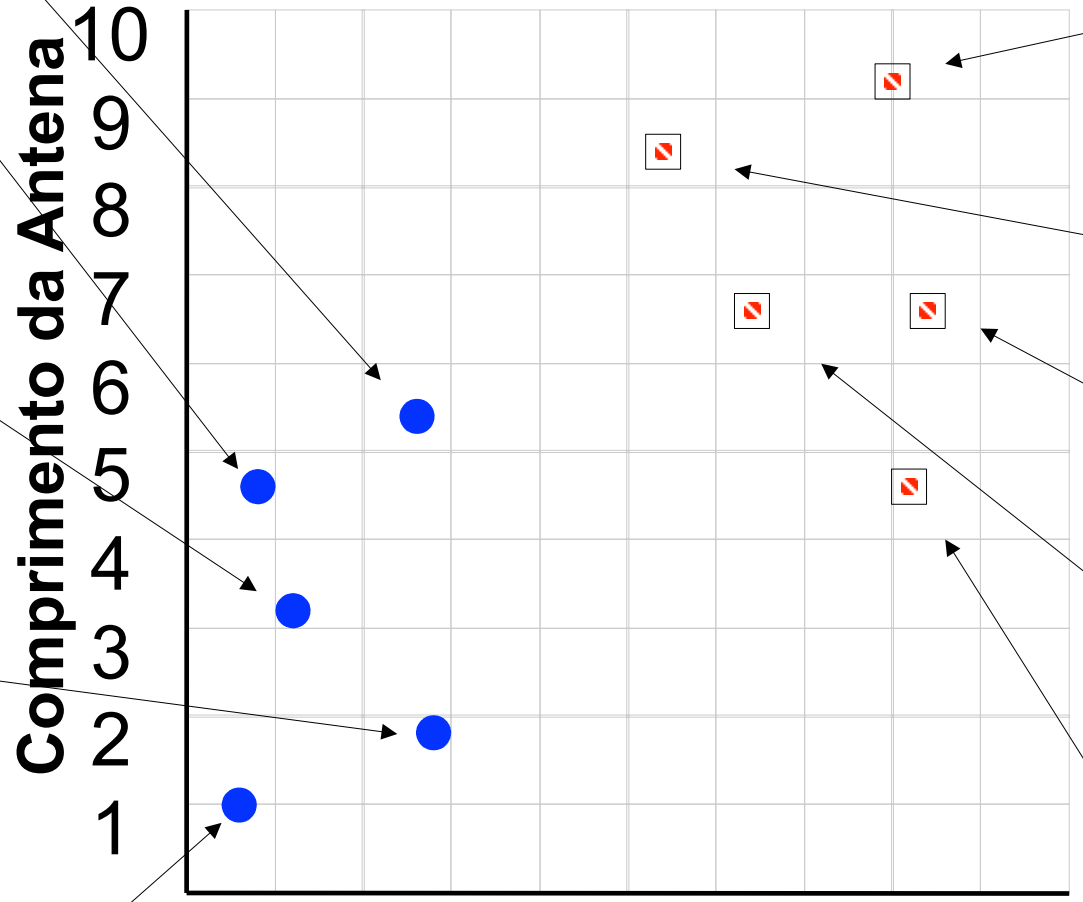
nova instância =

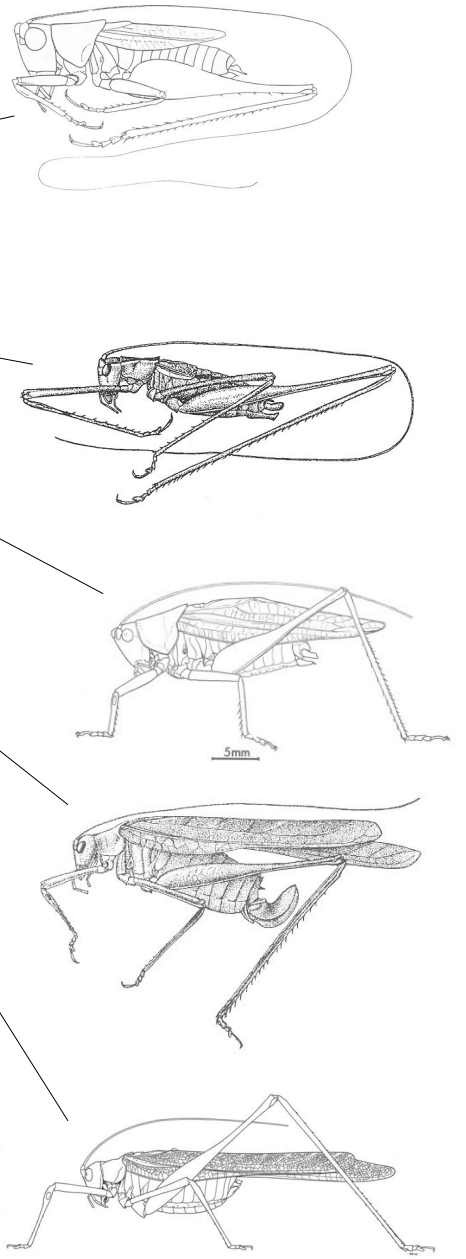
11	5.1	7.0	?????

Gafanhotos

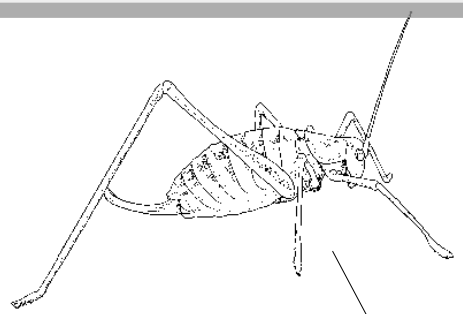
Esperanças

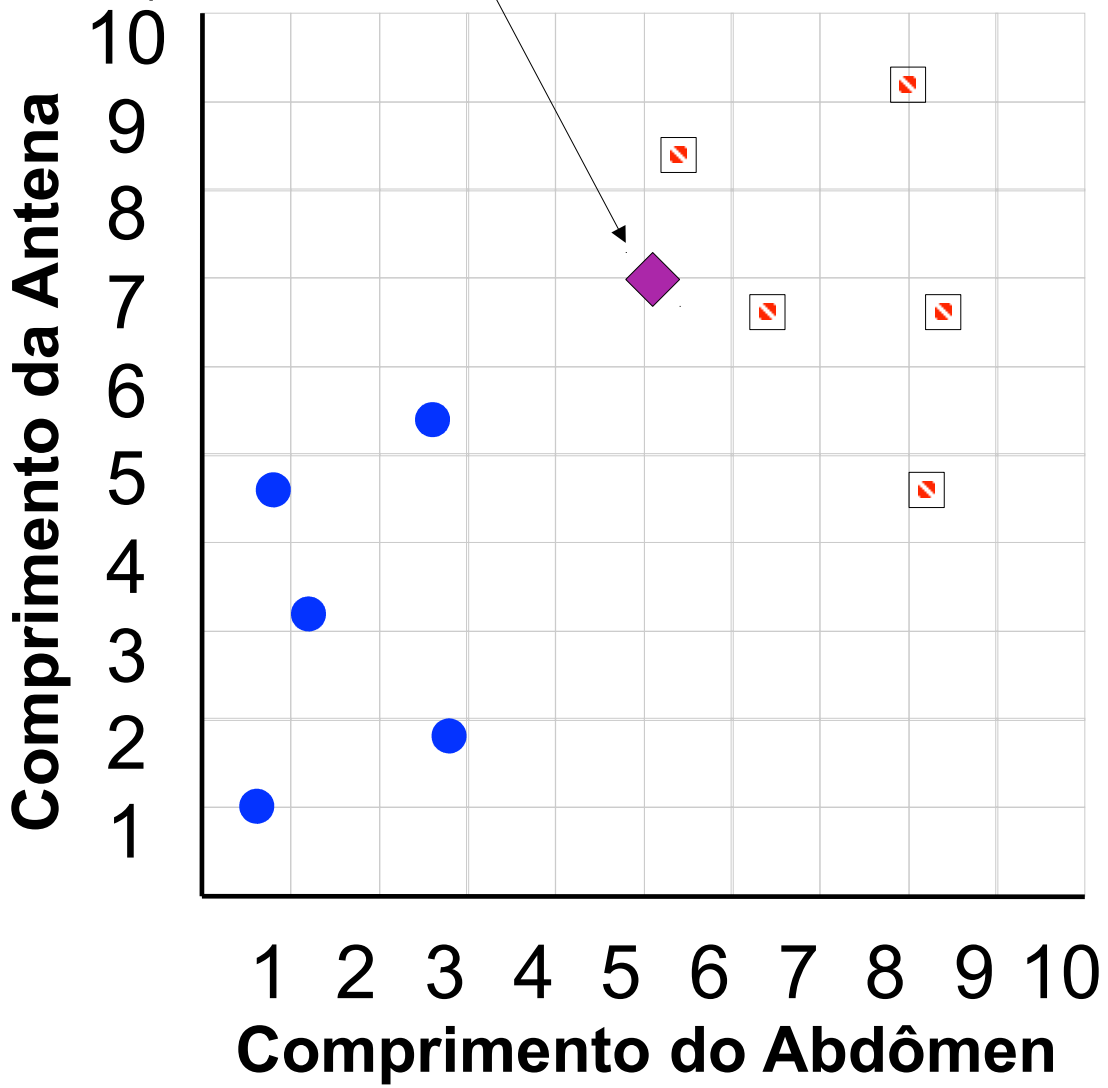






nova instância =				
	11	**5.1**	**7.0**	?????

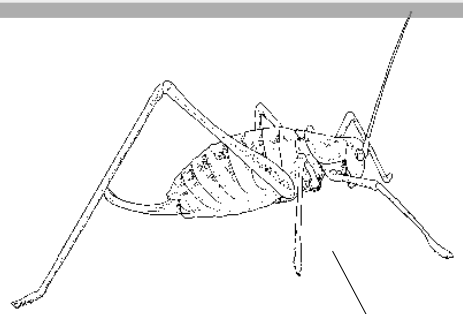


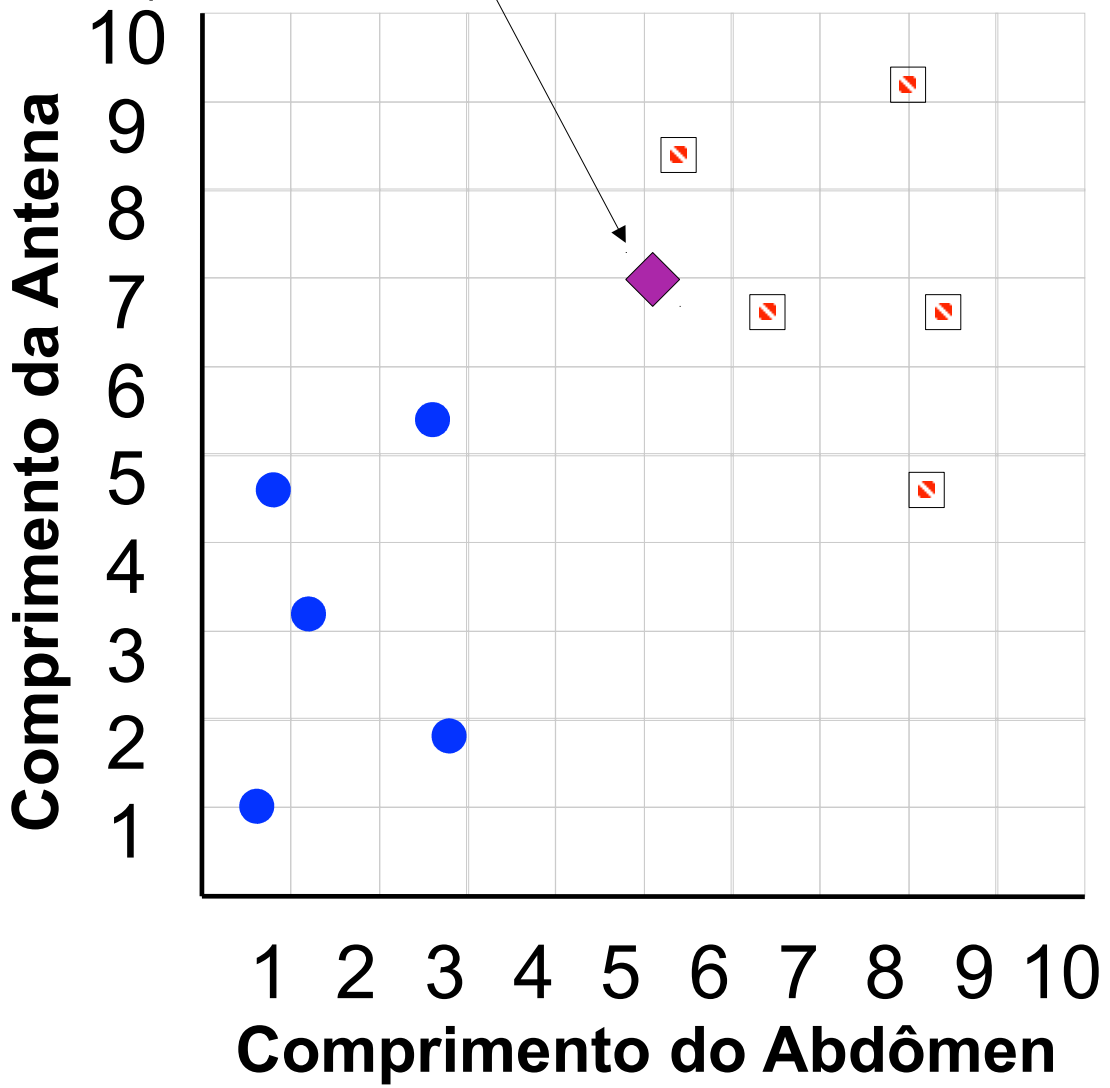


Qual critério usar para fazer a classificação?

- Esperanças
- Gafanhotos

nova instância =				
	11	**5.1**	**7.0**	?????





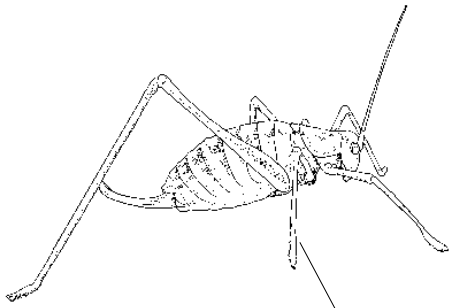
- Podemos "projetar" a nova instância no mesmo espaço do dataset.

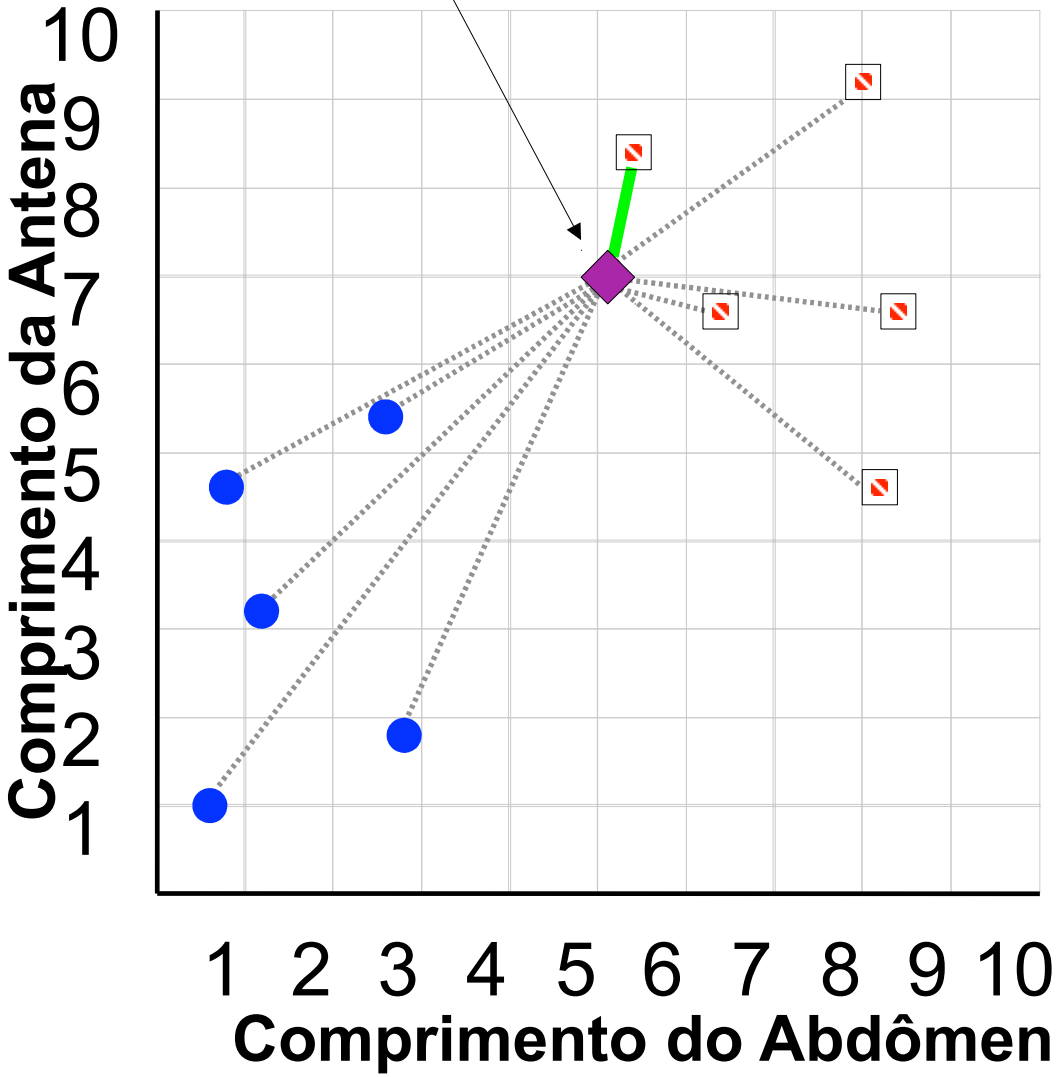
- Agora o problema passa a tratar apenas de pontos no espaço.

Esperanças

- **Gafanhotos**

Classificador Vizinho mais Próximo





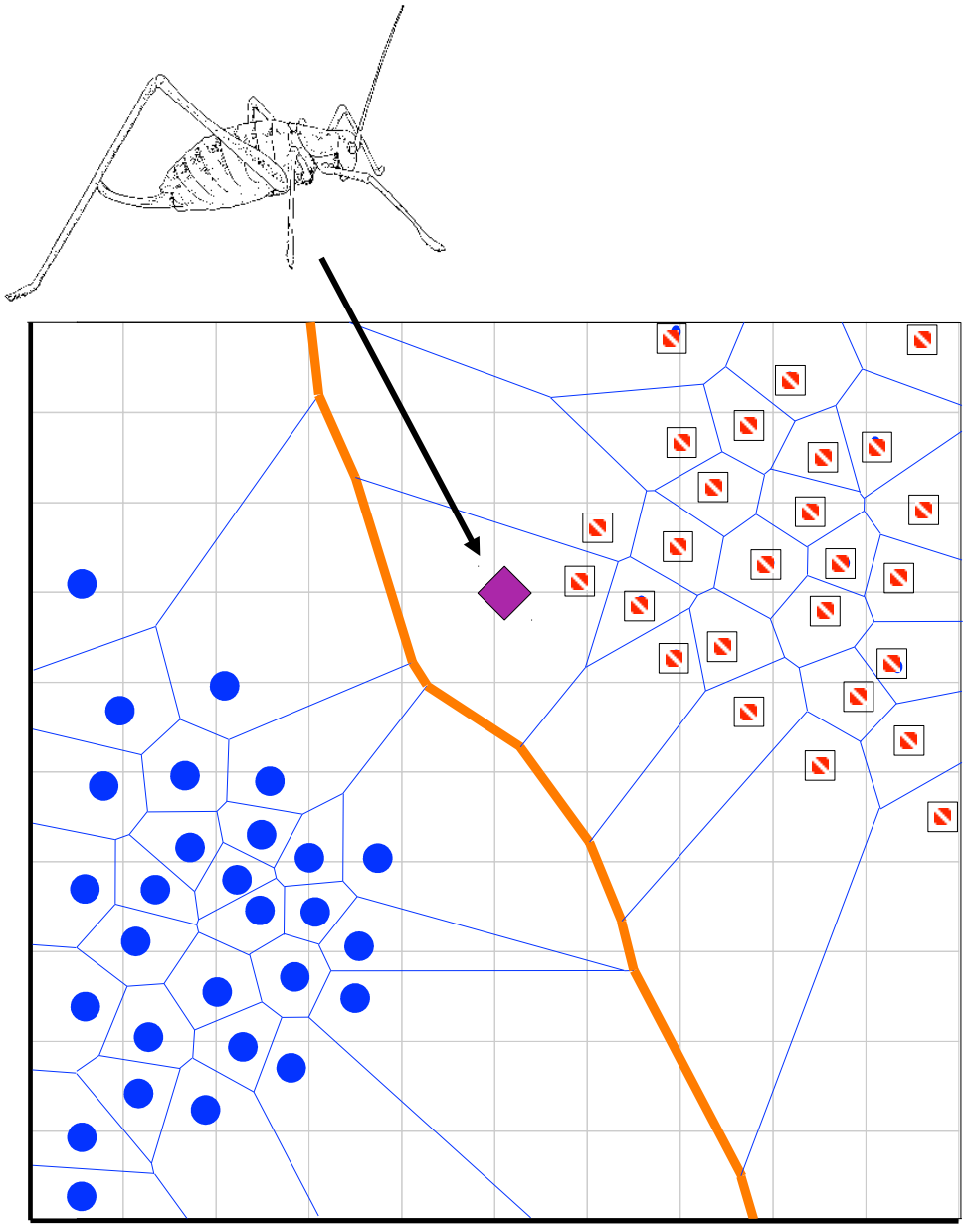
Se a instância **mais próxima** da nova instância **é uma Esperança**
classe é **Esperança**
se não
classe é **Gafanhoto**

Esperanças

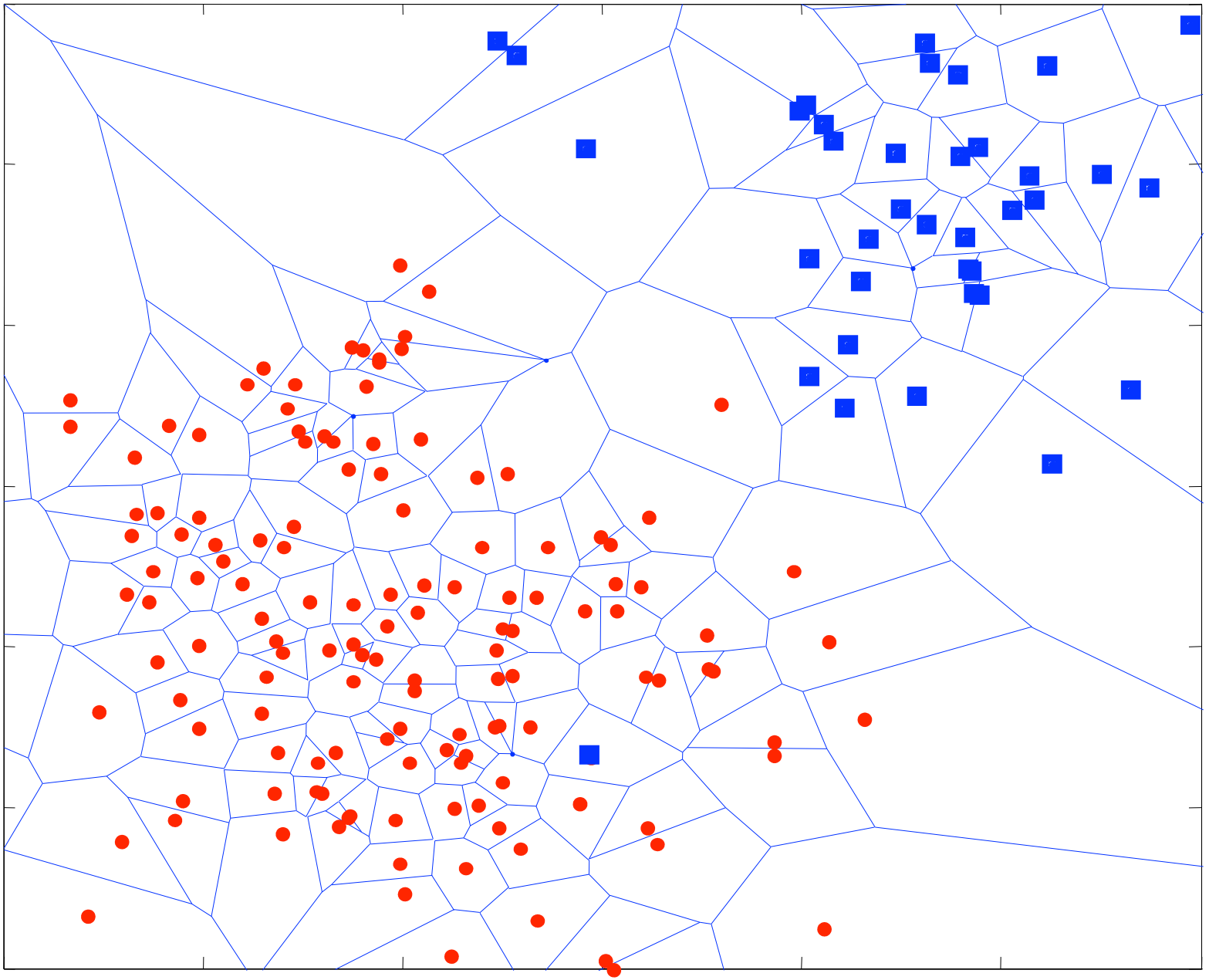
Gafanhotos

Superfície de Decisão

O algoritmo de classificação de Vizinho mais Próximo cria, **implicitamente**, regiões de classificação.



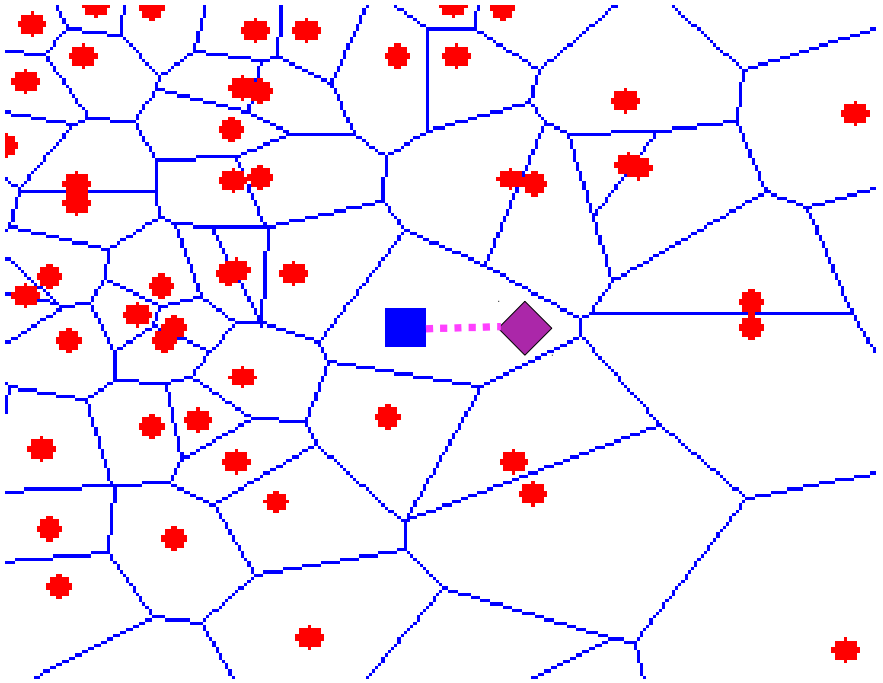
Sensibilidade a outliers:



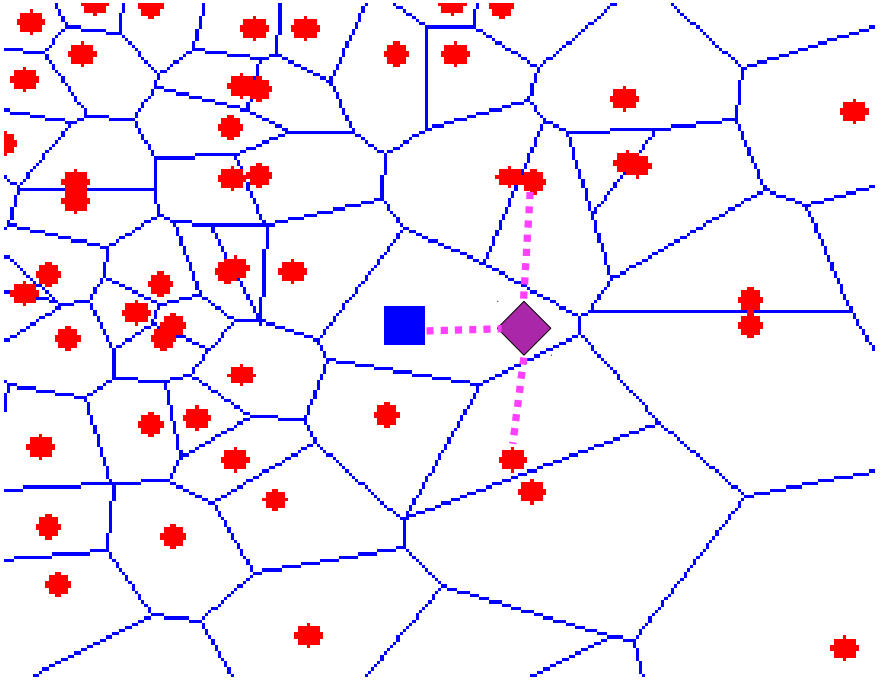
Solução???

Algoritmo K-Vizinhos mais Próximos
(k-nearest neighbors - KNN)

Encontramos os k vizinhos mais próximos e adotamos a classe da maioria.



K = 1



K = 3

Scikit-learn - KNN

O primeiro passo para se usar um algoritmo de classificação é preparar o dataset de **treinamento**. Abaixo selecionamos 10 apartamentos para indicar nosso interesse (que será a classe a ser prevista).

```
df_treino = df.head(10).copy()
df_treino['interesse'] = 'Não'
df_treino.loc[[167, 24, 469, 82],'interesse'] = 'Sim'

df_treino
```

codigo	quartos	area	vaga	aluguel	condominio	interesse
34	2	90	0	900	371	Não
167	2	64	0	650	428	Sim
6784	2	81	0	1100	400	Não
82	2	50	0	1350	300	Sim
2970	2	63	0	1300	300	Não
34197	2	80	1	900	410	Não
5072	2	84	0	1100	382	Não
469	1	30	0	550	210	Sim
24	1	60	1	800	120	Sim
74	2	132	1	1800	520	Não

Scikit-learn - KNN

Usamos os dados classificados anteriormente para treinar o modelo (**fit**). Com o modelo treinado podemos prever o interesse para novos apartamentos (**predict**).

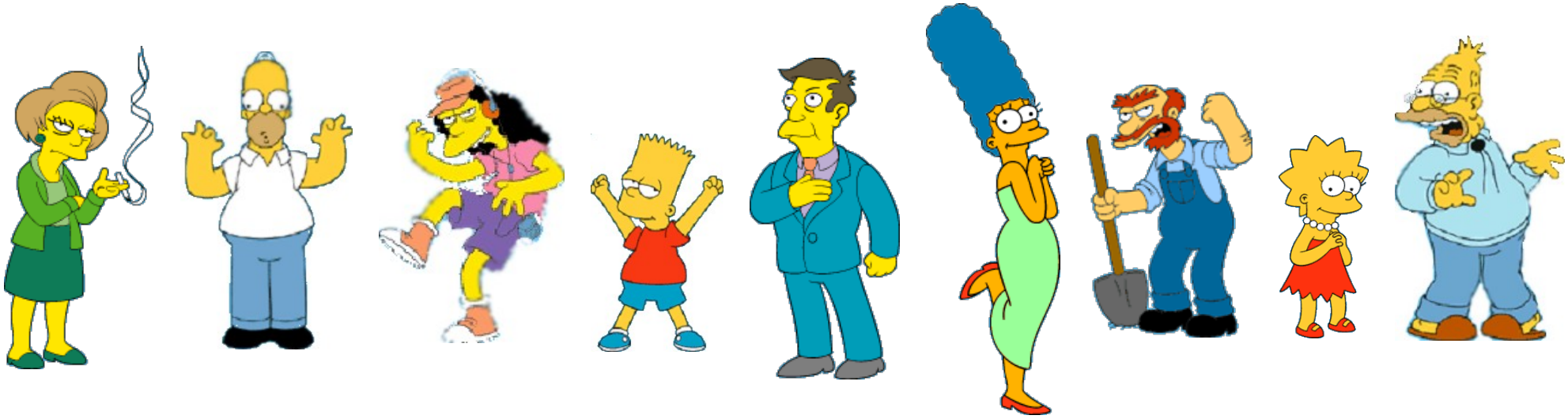
```
from sklearn.neighbors import KNeighborsClassifier
neigh = KNeighborsClassifier(n_neighbors=3)
neigh.fit(
    df_treino[['quartos', 'area', 'vaga',
               'aluguel', 'condominio']], df_treino['interesse'])

df_teste['predições'] = neigh.predict(df_teste)
df_teste
```

codigo	quartos	area	vaga	aluguel	condominio	predições
9850	1	64	1	600	326	Sim
82343	1	45	0	750	420	Não
20802	1	47	0	600	405	Sim
568	1	43	0	600	330	Sim
294579	1	36	0	550	350	Sim

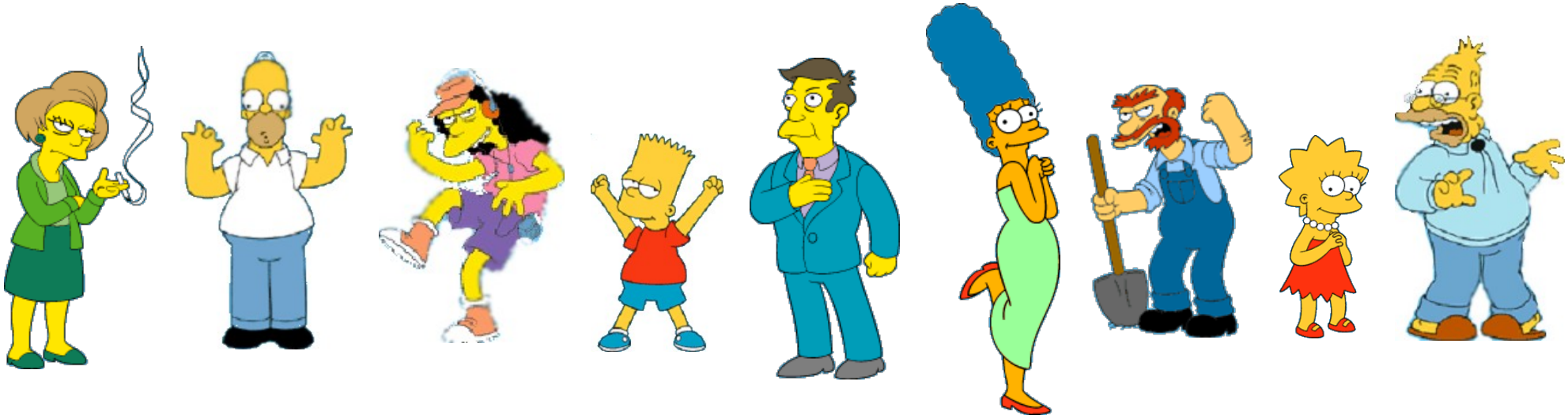
Clusterização

- Organiza os dados em classes que possuem
  - alta similaridade intra-classe
  - baixa similaridade inter-classe
- Determina as classes sem ajuda de exemplos (aprendizado não supervisionado).
- Informalmente, busca encontrar agrupamentos naturais entre os objetos (observações)

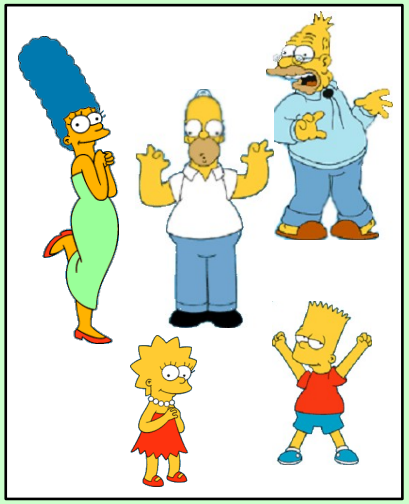
Como agrupar estes objetos?



Como agrupar estes objetos?

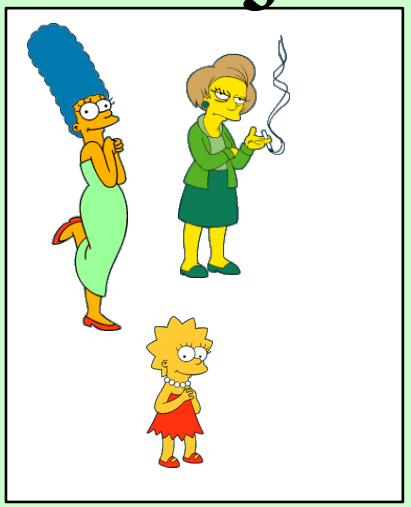


Clusterização é subjetiva!



Família

Escola



Feminino

Masculino

Clusterização

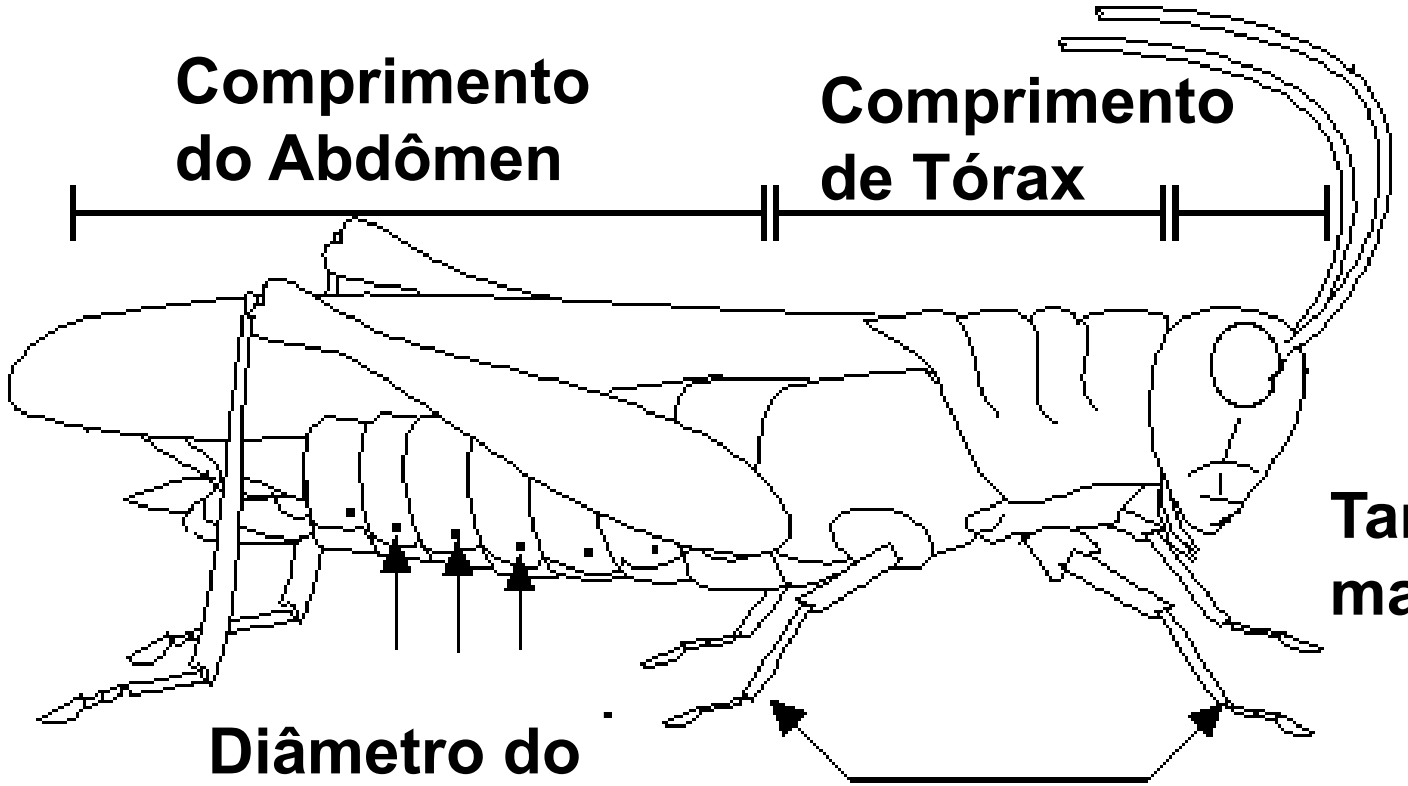
- Como no caso da classificação, vamos simplificar o problema para considerar apenas as características das instâncias no espaço
- Porém, neste caso não sabemos as classes de antemão. Por isto chamamos de aprendizado não supervisionado.

Medindo Características

Também chamadas de *features*, atributos

Cor {Verde, Marrom, Cinza, Outra}

Tem asas?



As características compõem nosso Dataset.

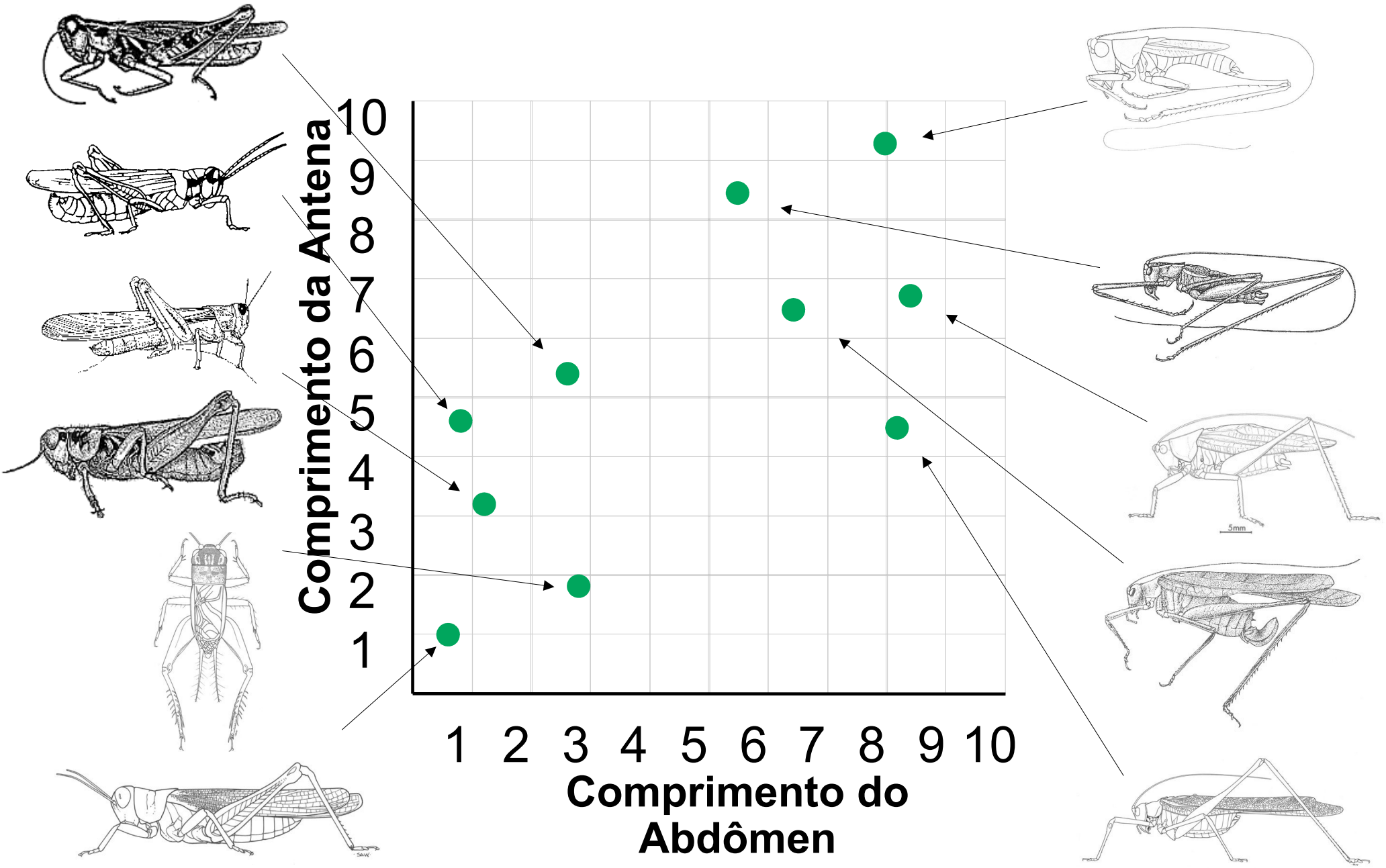
Problema de clusterização:

- Como organizar o dataset em **classes** distintas?

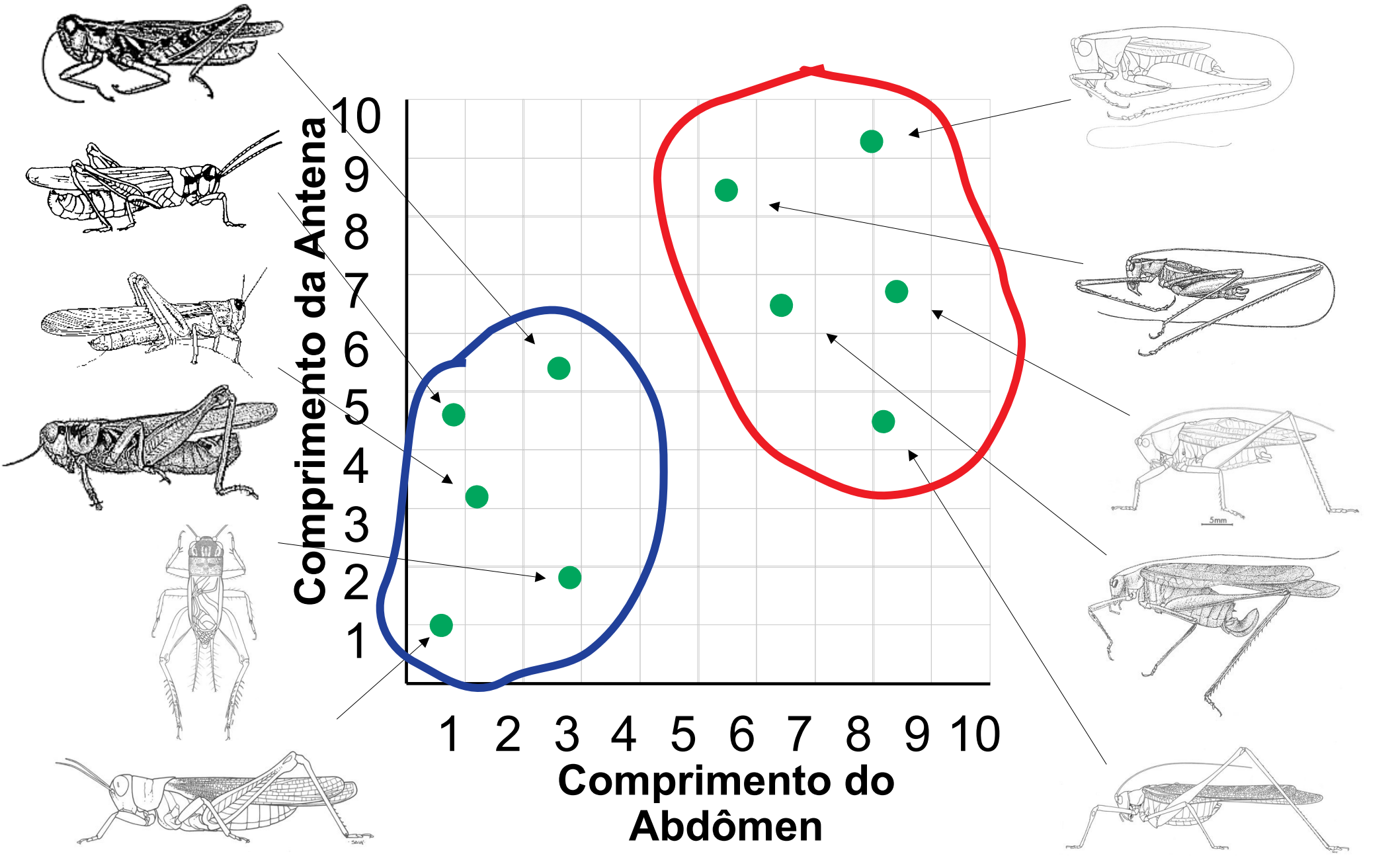
Dataset

ID	Abdômen	Antena	Classe
1	2.7	5.5	???
2	8.0	9.1	???
3	0.9	4.7	???
4	1.1	3.1	???
5	5.4	8.5	???
6	2.9	1.9	???
7	6.1	6.6	???
8	0.5	1.0	???
9	8.3	6.6	???
10	8.1	4.7	???

Como organizar o dataset em **classes** distintas?



Como organizar o dataset em **classes** distintas?



Algoritmo *k-means*

1. Escolher um valor de k (número de classes).

2. Inicializar os centros dos k clusters aleatoriamente.

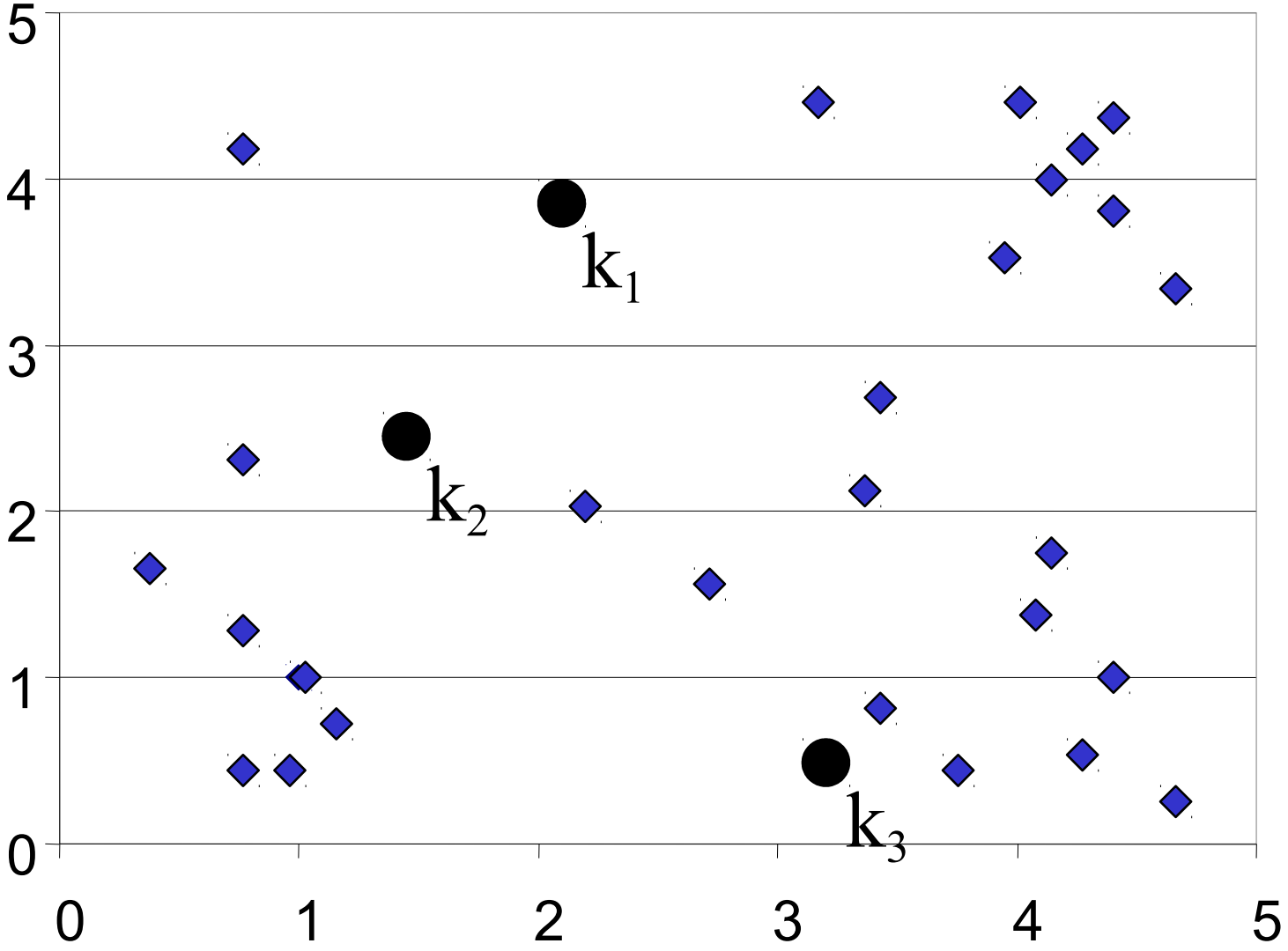
3. Classificar os N objetos de acordo com o centro do cluster mais próximo.

4. Re-calcular os contros dos clusters com base nas novas classes dos objetos.

5. Se nenhum dos N objetos mudar de classe, FIM! Caso contrário, ir para passo 3.

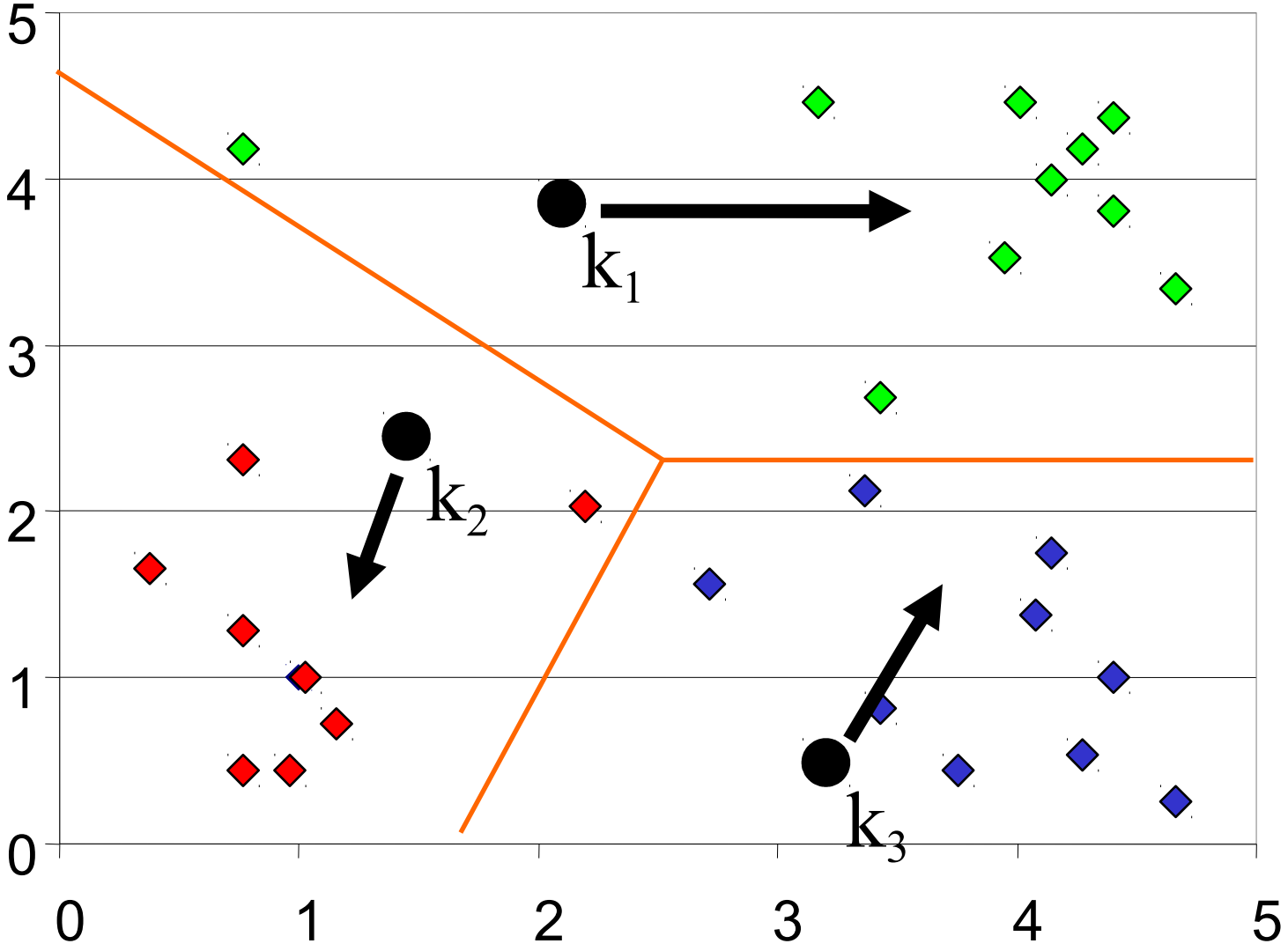
K-means Clustering: Step 1

Algorithm: k-means, Distance Metric: Euclidean Distance



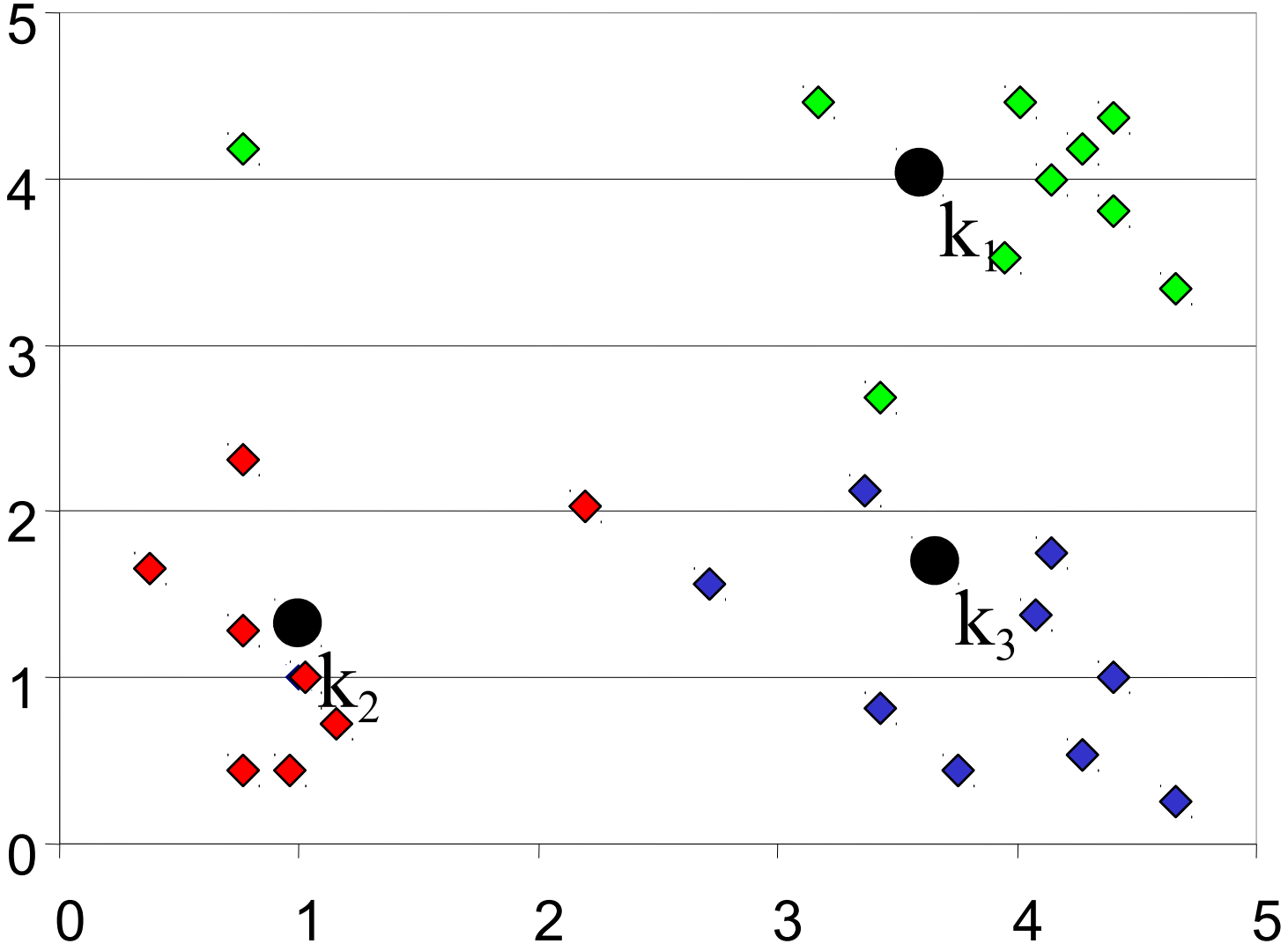
K-means Clustering: Step 2

Algorithm: k-means, Distance Metric: Euclidean Distance



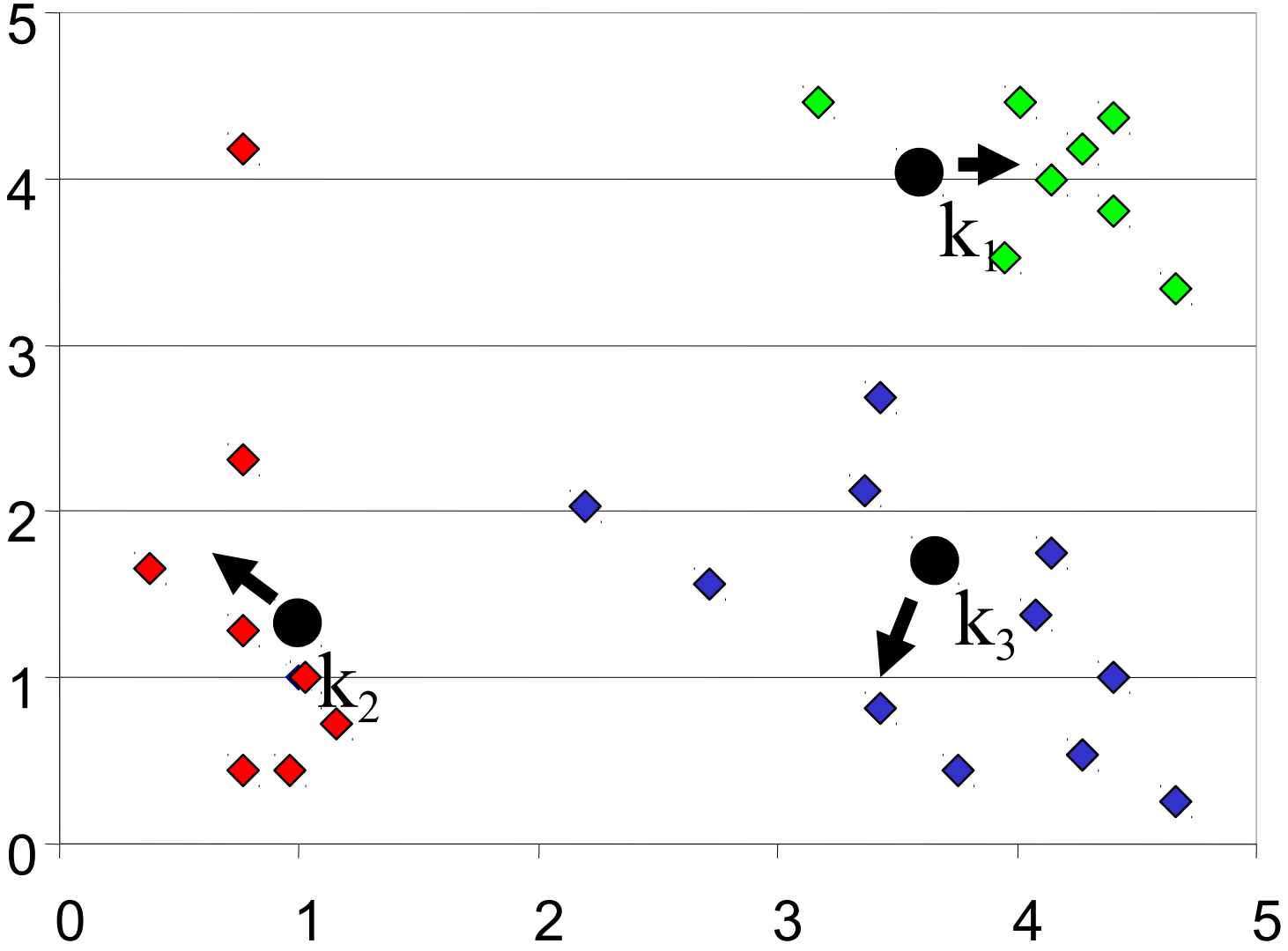
K-means Clustering: Step 3

Algorithm: k-means, Distance Metric: Euclidean Distance



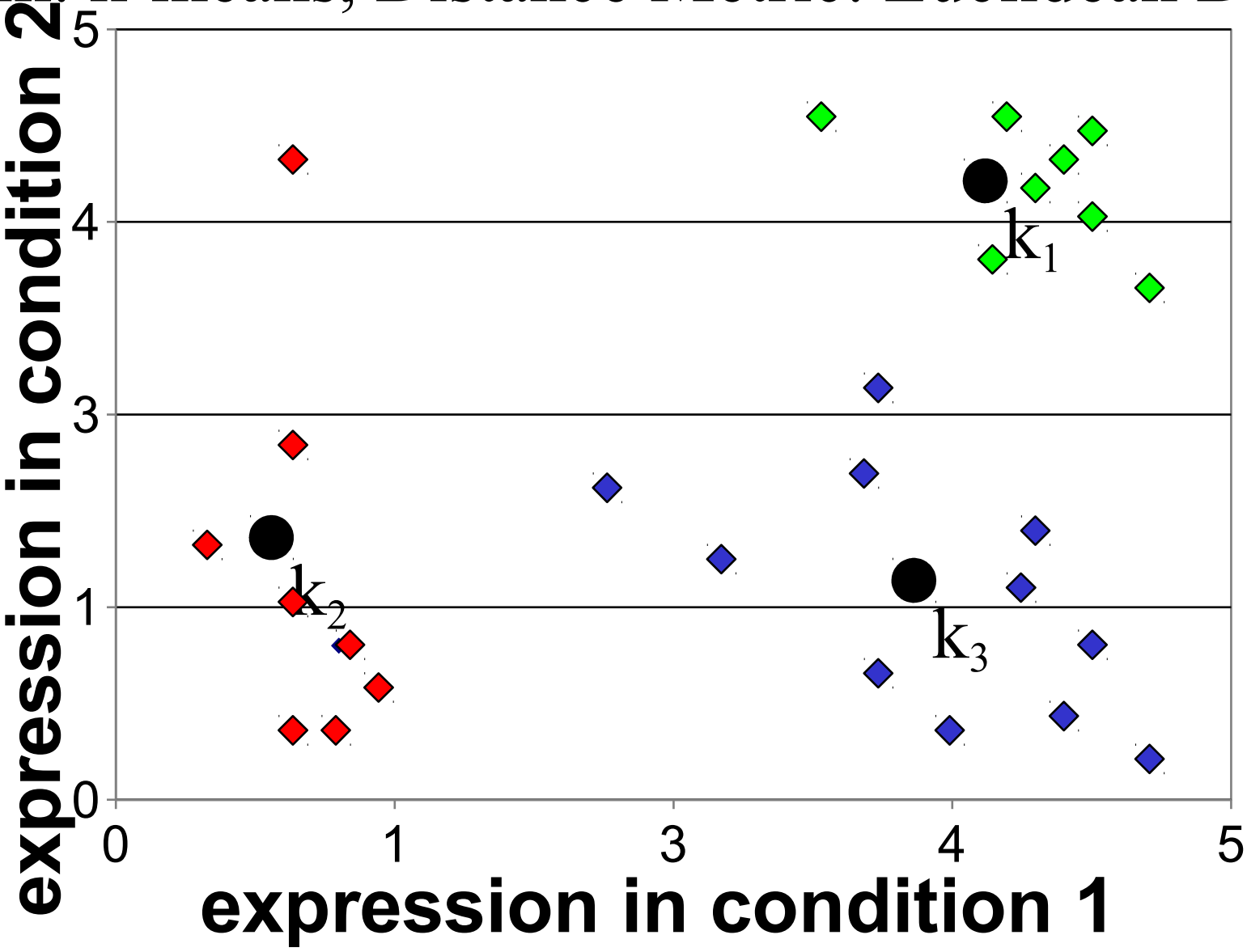
K-means Clustering: Step 4

Algorithm: k-means, Distance Metric: Euclidean Distance



K-means Clustering: Step 5

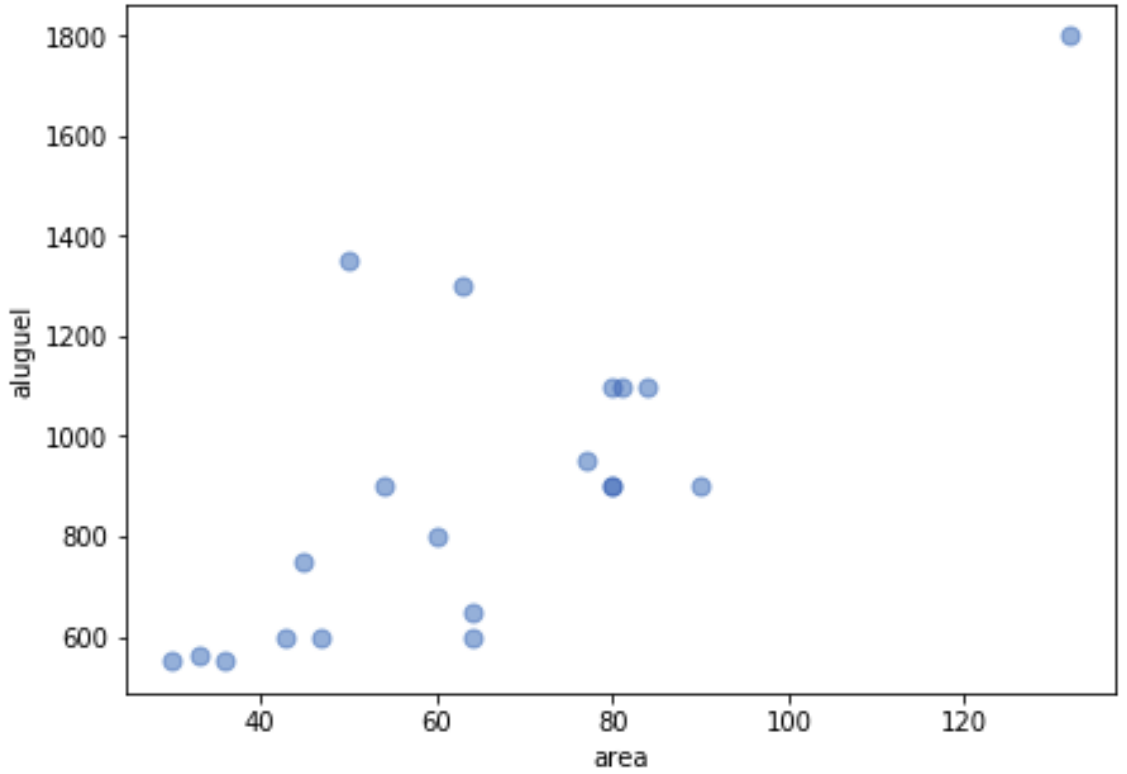
Algorithm: k-means, Distance Metric: Euclidean Distance



Scikit-learn - K-means

Vamos usar o algoritmo k-means para agrupar ofertas de aluguel considerando as variáveis área e valor do aluguel.

```
dfX = df[['area', 'aluguel']]

ax = dfX.plot.scatter(x='area', y='aluguel',
                      s = 50, alpha = 0.5, figsize=(7,5))
```



Scikit-learn - K-means

O k-means e muitos outros algoritmos de clusterização requerem que os dados sejam normalizados. O código abaixo faz esta normalização. Veja que agora os valores de área e aluguel estão sempre entre 0.0 e 1.0.

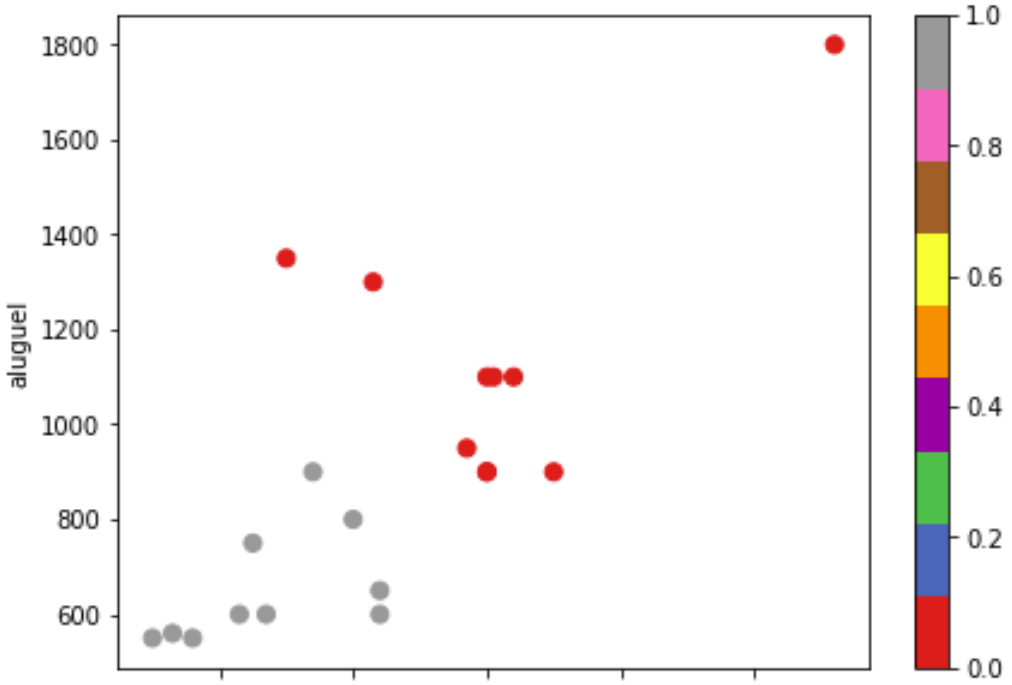
```
dfX_norm = pd.DataFrame(
                    MinMaxScaler().fit_transform(dfX),
                    index = dfX.index,
                    columns=dfX.columns)

dfX_norm
```

	area	aluguel
0	0.588235	0.280
1	0.333333	0.080
2	0.500000	0.440
3	0.196078	0.640
4	0.323529	0.600
5	0.490196	0.280
6	0.529412	0.440
7	0.000000	0.000

Scikit-learn - K-means

Abaixo aplicamos o algoritmo sobre nosso DataFrame. O algoritmo parece ter divido os apartamentos entre pequenos/baratos e grandes/caros.

```
kmeans1 = KMeans(n_clusters = 2, random_state = 42)
kmeans1.fit(dfX_norm)
df['Agrupamento 1'] = kmeans1.labels_

ax = dfX.plot.scatter(x='area', y='aluguel',
                      s = 50, c = kmeans1.labels_, colormap='Set1',
                      figsize=(7,5))
```



Scikit-learn - K-means

Esta divisão em grupos em geral não é determinística e é responsabilidade do cientista de dados interpretar os resultados.

	codigo	quartos	suite	area	vaga	aluguel	condominio	data	Agrupamento 1
0	34	2	0	90	0	900	371	11/10/17	0
17	66490	1	0	80	1	1100	350	29/08/17	0
16	80	1	0	80	1	900	350	12/08/17	0
6	5072	2	0	84	0	1100	382	02/09/17	0
5	34197	2	0	80	1	900	410	23/10/17	0
9	74	2	1	132	1	1800	520	12/10/17	0
3	82	2	0	50	0	1350	300	19/09/17	0
2	6784	2	0	81	0	1100	400	23/08/17	0
4	2970	2	0	63	0	1300	300	05/08/17	0
19	44803	2	0	77	1	950	200	19/07/17	0
7	469	1	0	30	0	550	210	03/07/17	1
8	24	1	0	60	1	800	120	30/09/17	1
18	2381	2	0	54	0	900	240	19/09/17	1
10	9850	1	0	64	1	600	326	15/07/17	1
11	82343	1	0	45	0	750	420	11/10/17	1
12	20802	1	0	47	0	600	405	27/07/17	1
13	568	1	0	43	0	600	330	12/08/17	1

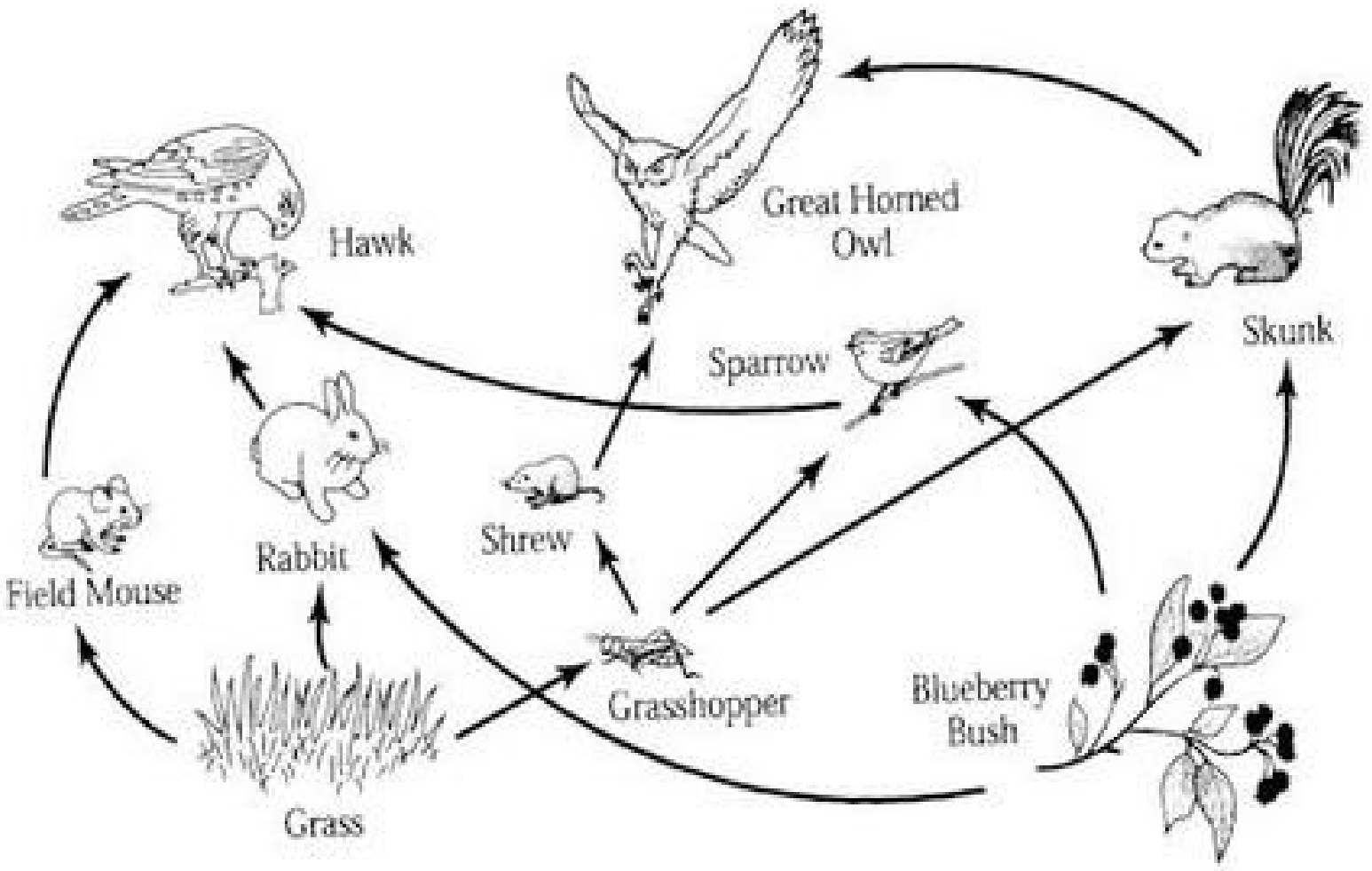
Exercícios!

- Revise o conteúdo e faça os exercícios do notebook:
  05b-Modelagem - Clusterização e Classificação.ipynb

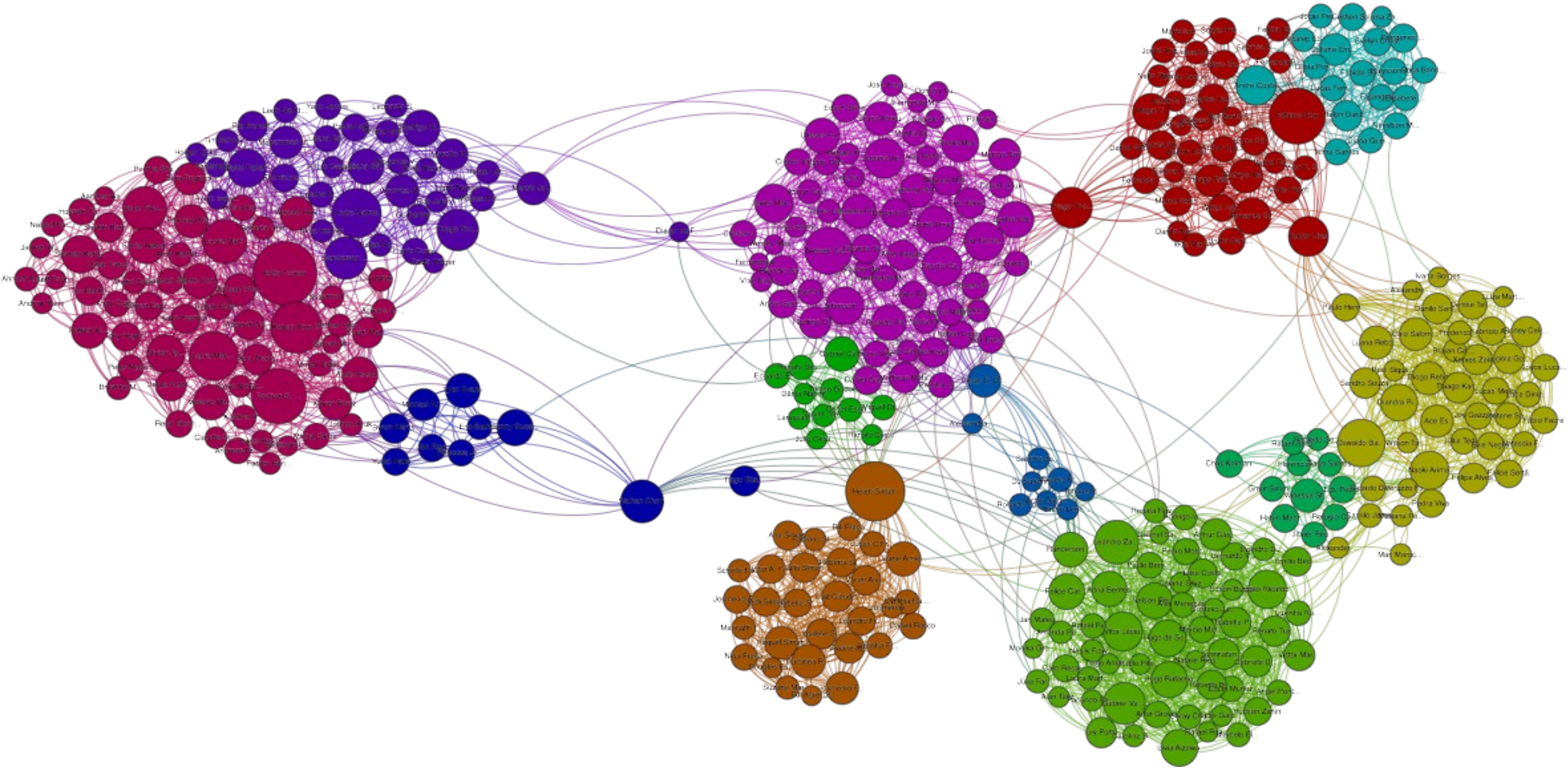
Grafos

- Usados para modelar dados interconectados
- Importantes para se entender diversos fenômenos complexos (epidemias, interações sociais, redes tróficas, eficiência em transportes, etc.)

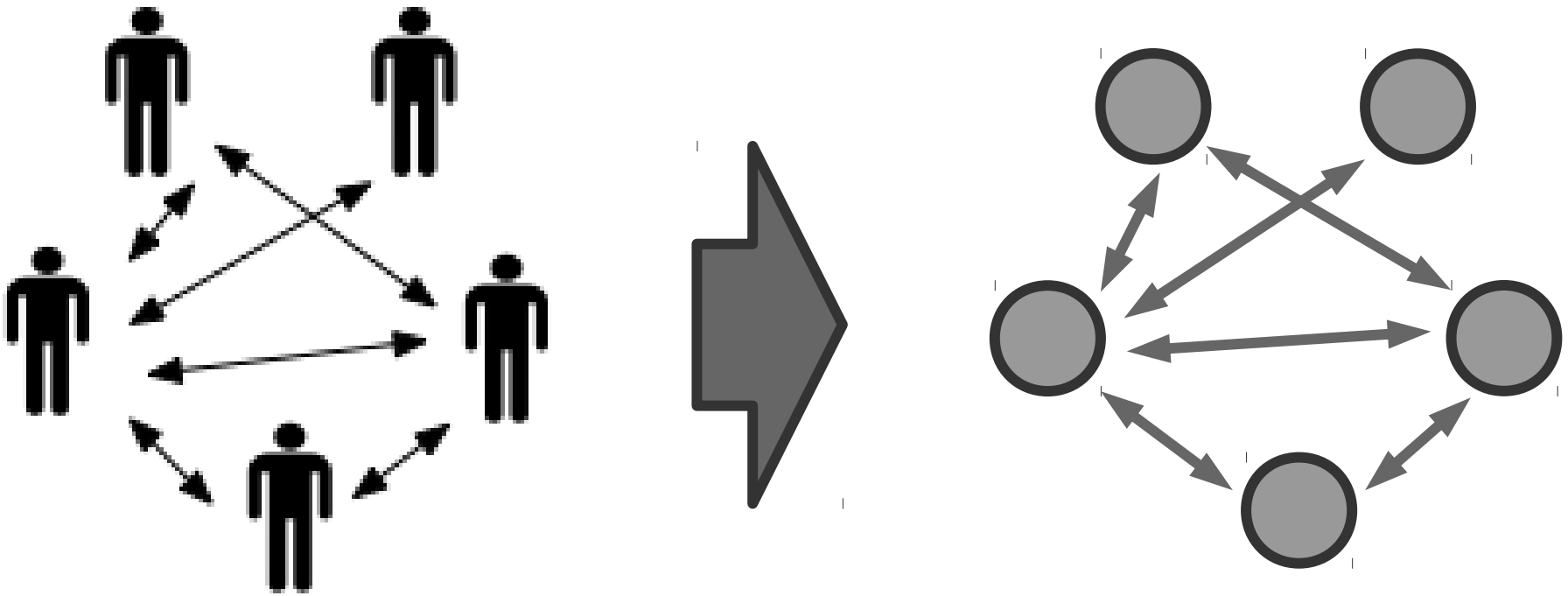
Grafos


Hawk
Great Horned Owl
Skunk
Sparrow
Field Mouse
Rabbit
Shrew
Grasshopper
Blueberry Bush
Grass

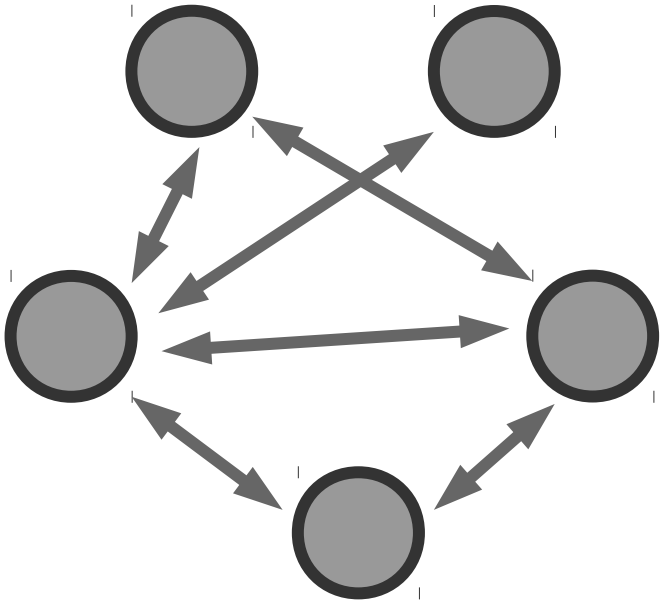
Social Networks

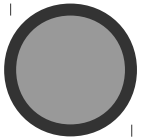


Modelo de Grafos

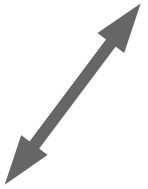


Modelo de Grafos





vértice ou nó



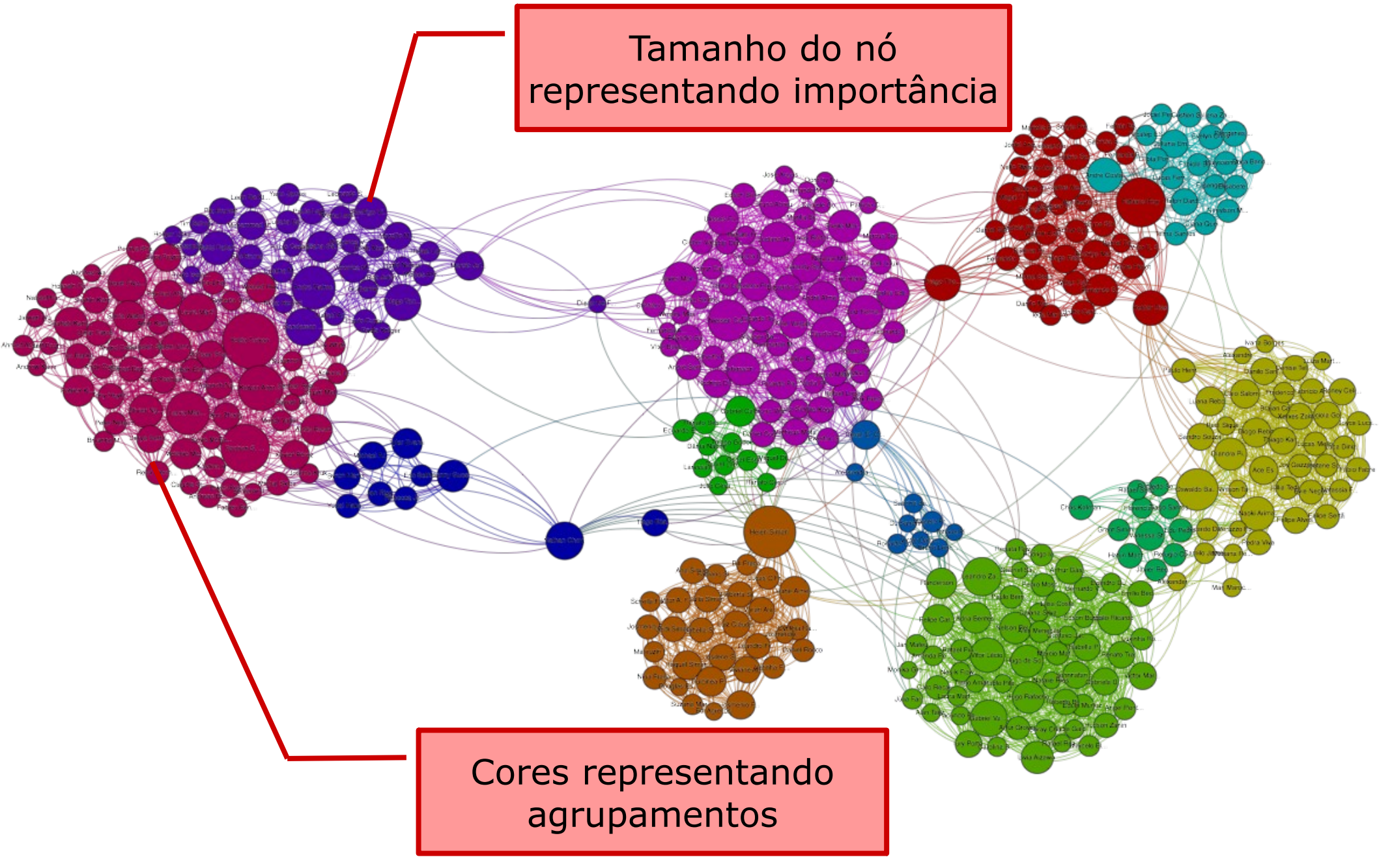
aresta ou ligação

Análise de Redes Complexas

Usa diversos algoritmos para entender padrões em grafos.

- Clusterização – Como os nós se agrupam?
- Centralidade – Quais são os nós mais importantes?
- Diversidade – Nós vizinhos tendem a ser parecidos?
- . . .

Análise de Redes Complexas



NetworkX

Usaremos a biblioteca NetworkX para construir e analisar os grafos. O NetworkX oferece vários algoritmos de análise, plotagem e exportação de grafos.

```
#importação de bibliotecas
import networkx as nx
import pandas as pd
import matplotlib.pyplot as plt

%matplotlib inline
```

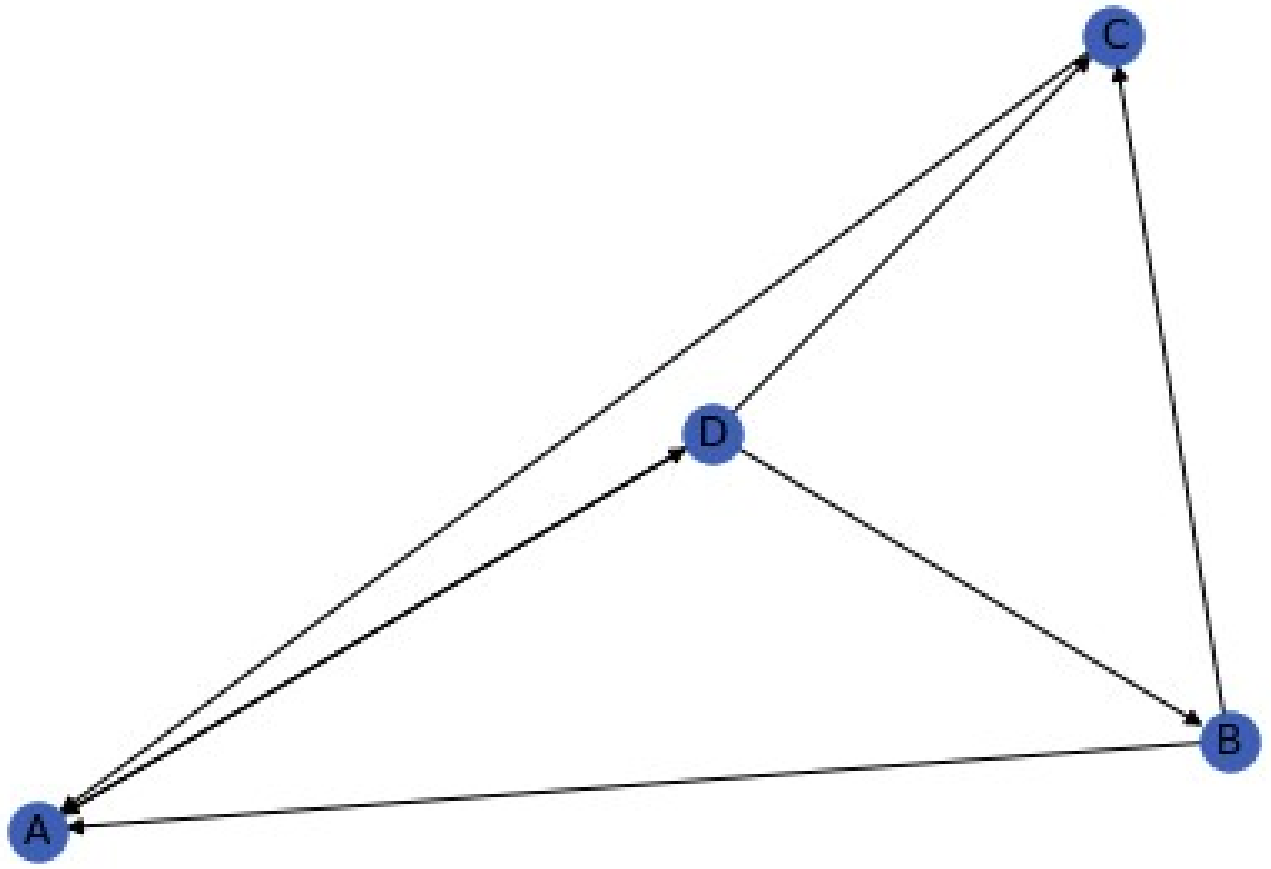
NetworkX - Construindo um grafo

O código abaixo inicializa um grafo direcionado usando nx.DiGraph(). Para adicionar nós e ligações, usamos os métodos add_nodes e add_edges.

```
# Inicializa um grafo direcionado
G = nx.DiGraph()

# Define o label de cada nó
labels = {0: 'A', 1: 'B', 2: 'C', 3: 'D'}

G.add_nodes_from(labels)

# Adiciona links a partir de uma lista
G.add_edges_from([(1,2),(1,0),(2,0),(3,0),(3,1),(3,2),(0,3)])

# Desenha o grafo
nx.draw(G, labels = labels)
```

NetworkX - Construindo um grafo

```
# Desenha o grafo
nx.draw(G, labels = labels)
```



Construindo um grafo a partir de um DataFrame

É possível construir um grafo a partir de colunas de um DataFrame. As colunas devem indicar os nós de origem e destino das ligações. Usando o DataSet de reclamações, podemos construir um grafo ligando os bairros de origem dos cidadãos (BAIRRO_CIDADAO) e os respectivos bairros alvo das reclamações.

```
G = nx.convert_matrix.from_pandas_edgelist(df_g,
                                           'BAIRRO_ASS',
                                          'BAIRRO_CIDADAO')

type(G)
```

```
networkx.classes.graph.Graph
```

Exibindo o grafo gerado

```
plt.figure(figsize=(15, 15))

# Define as posições para exibição dos nós
pos = nx.spring_layout(G)

# Desenha os rótulos (bairros)
nx.draw_networkx_labels(G, pos, font_size = 9)

# Desenha o restante do grafo
nx.draw(G, pos, alpha = 0.5, edge_color='grey')
```

Exibindo o grafo gerado

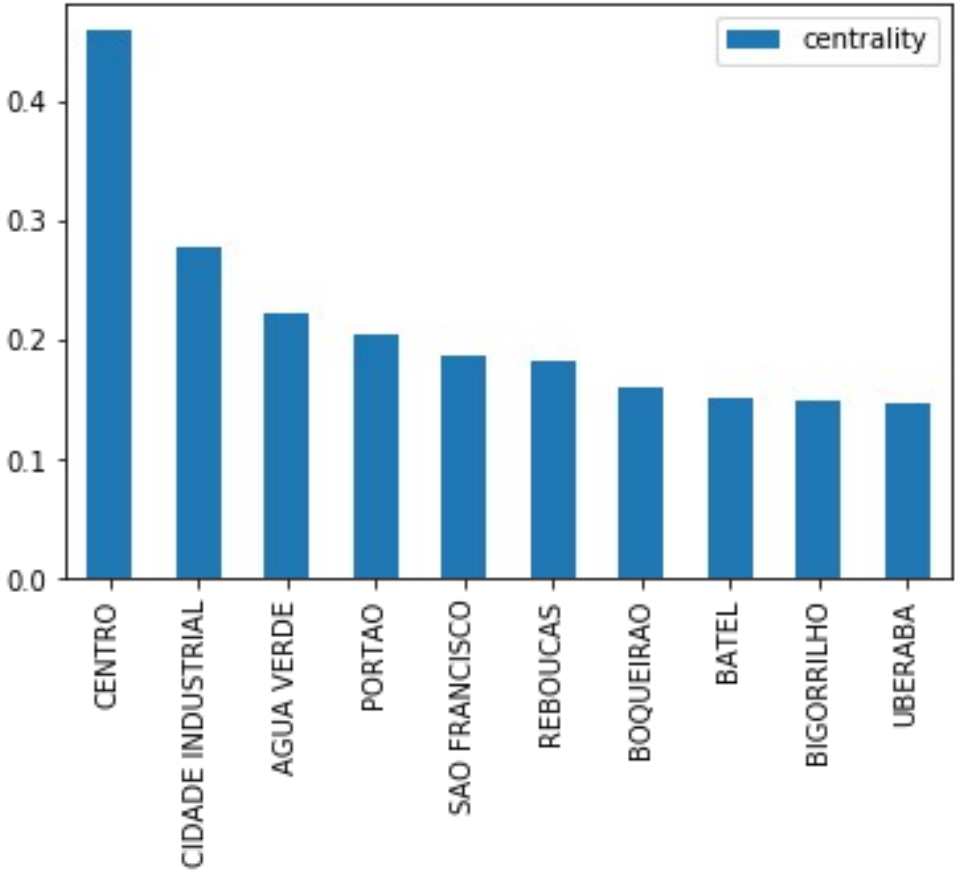
Análise do grafo

O NetworkX permite o cálculo de diversas métricas sobre os grafos. No código abaixo calculamos a centralidade e exibimos um gráfico com os bairros mais importantes de acordo com a métrica.

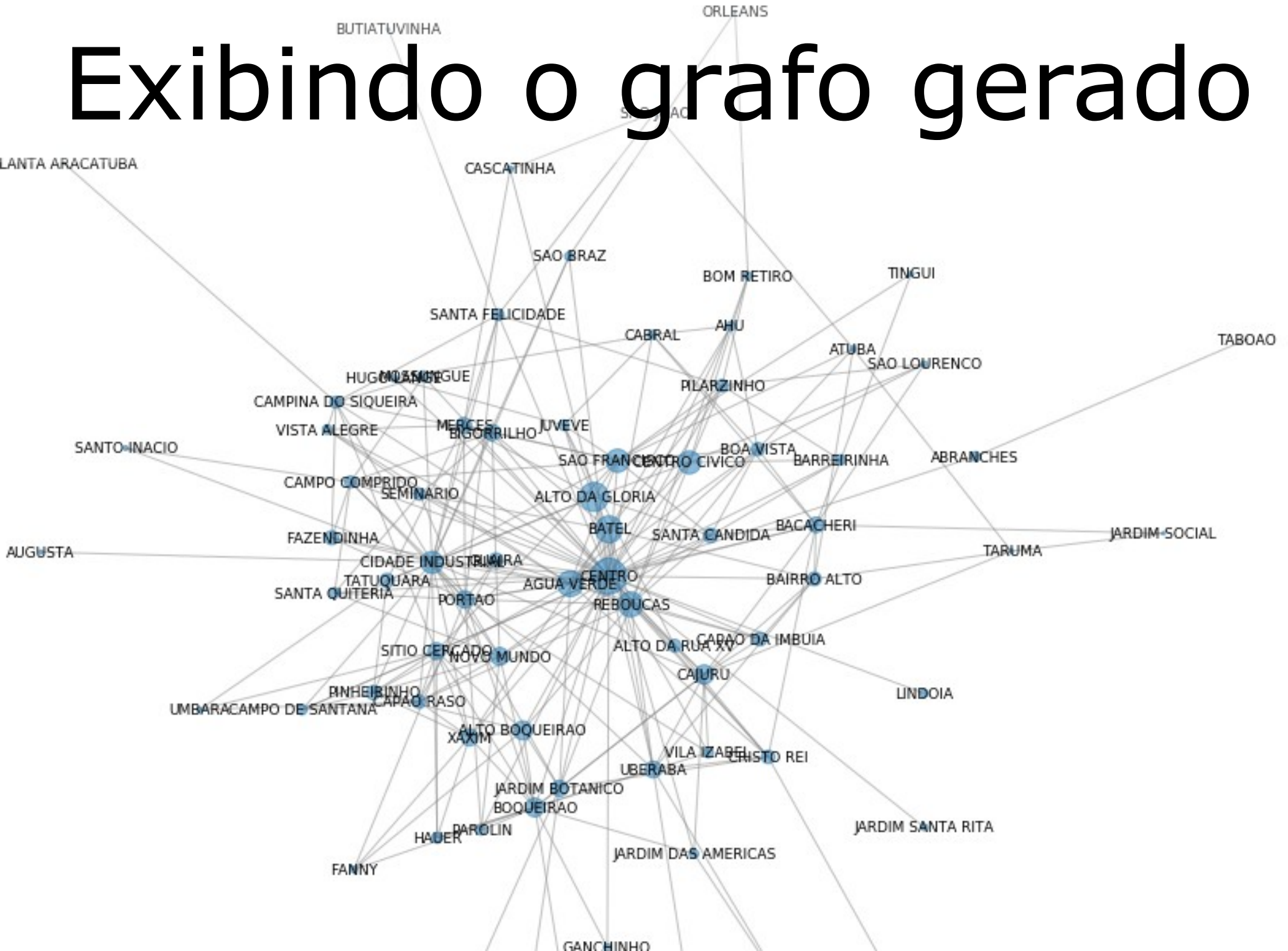
```
# Calculando os valores de centralidade
centrality=nx.eigenvector_centrality_numpy(G, weight='weight')

# Atribuindo os valores aos nós do grafo
nx.set_node_attributes(G, centrality, 'centrality')

# Exibindo os valores em um gráfico de barras
df_i = pd.DataFrame([centrality], index=['centrality']).T
df_i.sort_values('centrality', ascending=False).head(10).plot.bar()
```

Análise do grafo

O NetworkX permite o cálculo de diversas métricas sobre os grafos. No código abaixo calculamos a centralidade e exibimos um gráfico com os bairros mais importantes de acordo com a métrica.



Exibindo o grafo gerado



Exercícios!

- Revise o conteúdo e faça os exercícios do notebook:
  05c-Modelagem - Grafos.ipynb

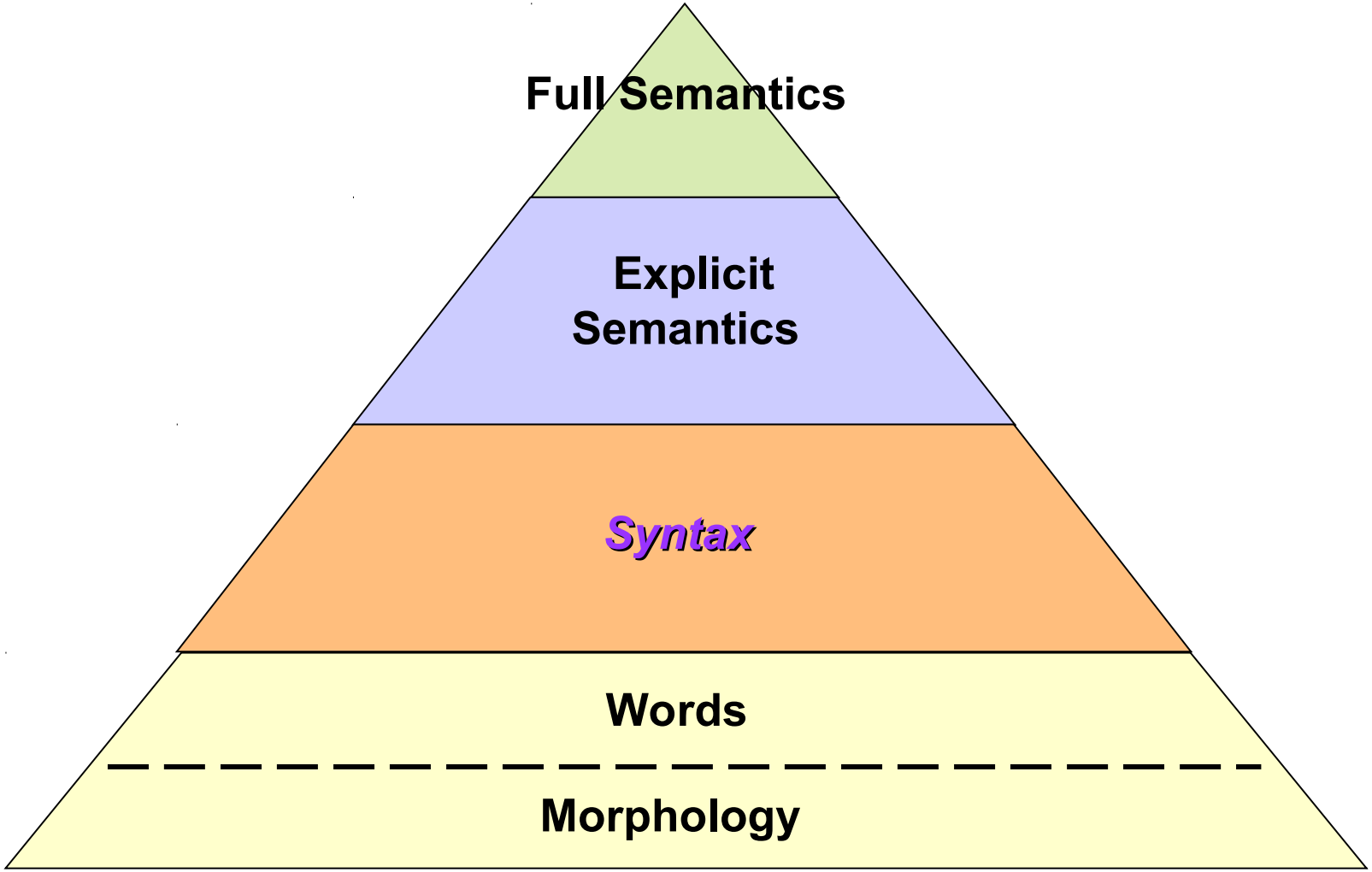
Texto

Major threats to the species include **cattle grazing**, **agriculture** activities and **mining** activities throughout its range. A museum specimen collected from Reserva Forestal de Yotoco in 1996 tested positive for *Batrachochytrium dendrobatidis* (Velasquez et al. 2007). The presence of chytrid in this species in 1996 is consistent with the timing of the declines observed in the Yotoco subpopulations at the end of the 1990s, as well as the timing of other Bd declines in montane Andean species, suggesting it as a plausible, but unconfirmed cause. However, the species can still be found within Reserva Forestal de Yotoco (Velasquez et al. 2007).

Processamento de Linguagem Natural

Processamento de Linguagem Natural (Natural language processing - NLP) é um campo da **ciência da computação**, **inteligência artificial** e **linguística computacional**. A área foca nas interações entre **computadores e linguagens humanas**, particularmente no treinamento de computadores para processar **grandes corpus de linguagem natural**.

Processamento de Linguagem Natural



Also, higher representations *require* lower

NLTK

A biblioteca NLTK - Natural Language Toolkit possui diversas ferramentas para processamento e análise de texto. Abaixo importamos a biblioteca e obtemos as stopwords em português:

```
# Importando as bibliotecas
import nltk
from nltk.tokenize import word_tokenize
import unicodedata

stopwords = nltk.corpus.stopwords.words('portuguese')
print(stopwords)
```

```
['de', 'a', 'o', 'que', 'e', 'do', 'da', 'em', 'um', 'para', 'co
m', 'não', 'uma', 'os', 'no', 'se', 'na', 'por', 'mais', 'as', 'do
s', 'como', 'mas', 'ao', 'ele', 'das', 'à', 'seu', 'sua', 'ou', 'q
uando', 'muito', 'nos', 'já', 'eu', 'também', 'só', 'pelo', 'pel
a', 'até', 'isso', 'ela', 'entre', 'depois', 'sem', 'mesmo', 'ao
s', 'seus', 'quem', 'nas', 'me', 'esse', 'eles', 'você', 'essa',
'num', 'nem', 'suas', 'meu', 'às', 'minha', 'numa', 'pelos', 'ela
s', 'qual', 'nós', 'lhe', 'deles', 'essas', 'esses', 'pelas', 'est
e', 'dele', 'tu', 'te', 'vocês', 'vos', 'lhes', 'meus', 'minhas',
'teu', 'tua', 'teus', 'tuas', 'nosso', 'nossa', 'nossos', 'nossa
```

Normalização de Texto

Frequentemente precisamos normalizar o texto antes de processá-lo. A normalização desejada depende do problema em questão. Procedimentos comuns são:

- Remoção de acentos
- Remoção de stopwords
- Remoção de plural e outros sufixos (stemming)
- Correção ortográfica
- Remoção de números e outros elementos não linguísticos (URLs, emoticons…)
- Identificação de termos individuais (tokenização)

Normalização de Texto

A função **normaliza_texto** definida abaixo tokeniza o texto de entrada, passa as palavras para caixa-baixo, retira *stopwords* e números.

```
def normaliza_texto(txt):
    return ' '.join([word for word in word_tokenize(str.lower(remov

df['RESP_NOR'] = df.apply(
       lambda linha: normaliza_texto(str(linha['RESPOSTA_FINAL']))
       , axis = 1)

df[['RESP_NOR']]
```

	RESP_NOR
0	abordagem realizada pessoa orientada quanto pr...
1	abordagem realizada encontrada pessoa solicita...
2	abordagem realizada encontrada pessoa solicita...
3	pessoa solicitacao abordada nesta data atendim...

Agrupamento com Concatenação de Texto

Nesta análise, estamos interessados em saber, para cada bairro, o que é dito no campo RESPOSTA_FINAL. Para isto vamos agrupar nosso DataFrame por bairro e fazer a união de todos os textos de um mesmo bairro.

```
# Agrupamentos e união dos textos por bairro
df_bairros = df.groupby('BAIRRO_ASS')['RESP_NOR']
df_bairros = df_bairros.apply(lambda x: ' '.join(x))

# Mostrando o conteúdo do bairro Centro
df_bairros['CENTRO']
```

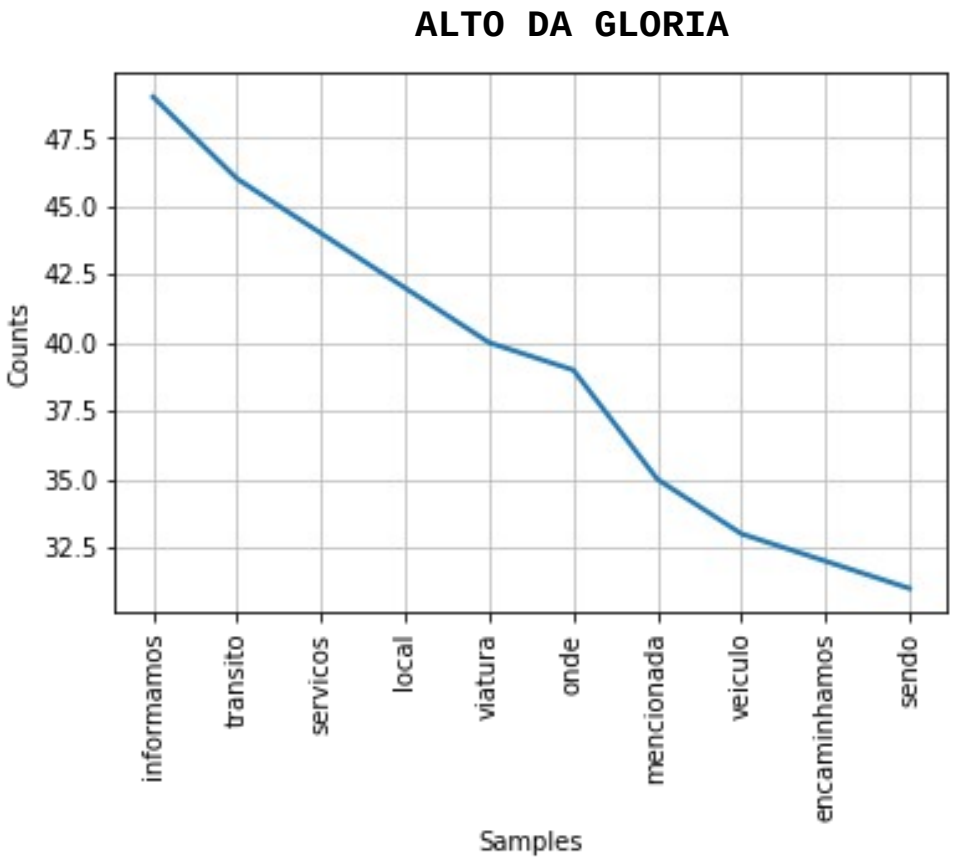
```
'abordagem realizada pessoa orientada quanto procedimentos unida
des fas disponiveis abordagem realizada encontrada pessoa solici
tacao local indicado imediacoes abordagem realizada encontrada p
essoa solicitacao local indicado imediacoes pessoa solicitacao a
bordada nesta data atendimento car localizada rodoferroviaria pe
ssoa solicitacao faz parte rede atendimento socio assistencial m
unicipio curitiba sendo orientado sobre procedimentos fas podend
o procurar servicos procura espontanea orientado durante periodo
diurno deve frequentar centro pop noite liberado acesso acolhime
nto noturno pessoa solicitacao faz parte rede atendimento socio
assistencial municipio curitiba sendo orientado sobre procedimen
tos fas podendo procurar servicos procura espontanea orientado d
```

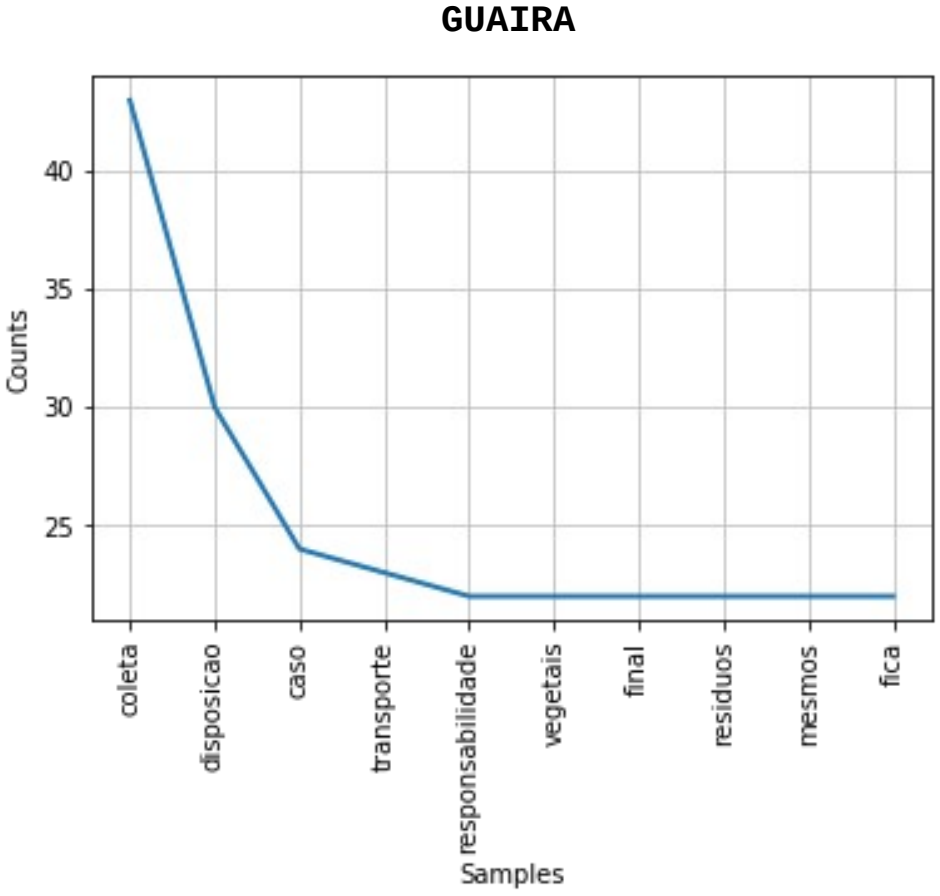
Análise da distribuição de palavras

Vamos agora analisar a frequência das palavras em cada texto, para termos uma ideia do tipo de reclamação em cada bairro. Para isto iteramos pelos dados e aplicamos a função FreqDist do NLTK.

```
from nltk.probability import FreqDist

for bairro, texto in df_bairros.items():
    print("################ {} ################".format(bairro))
    freqDist = FreqDist(texto.split(" "))
    freqDist.plot(10)
```

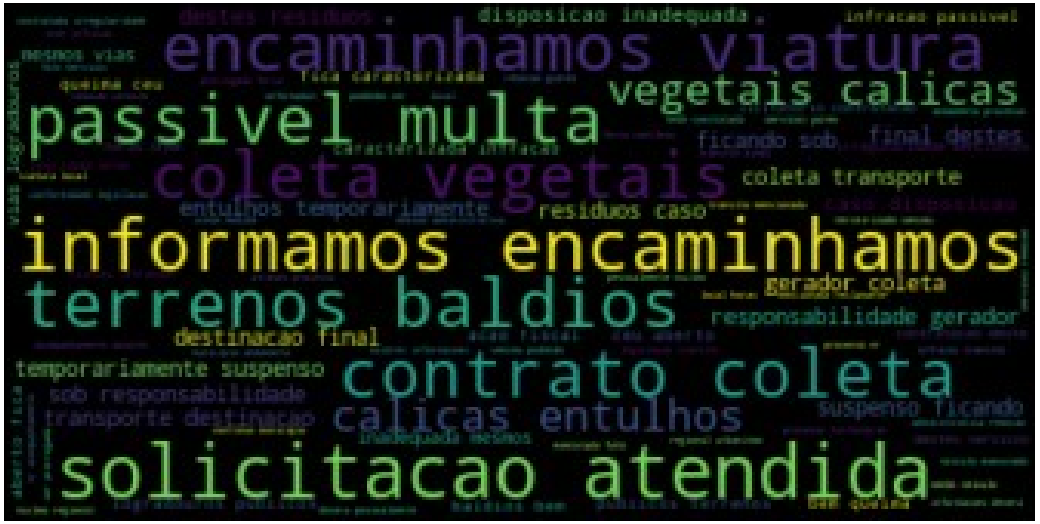
Análise da distribuição de palavras



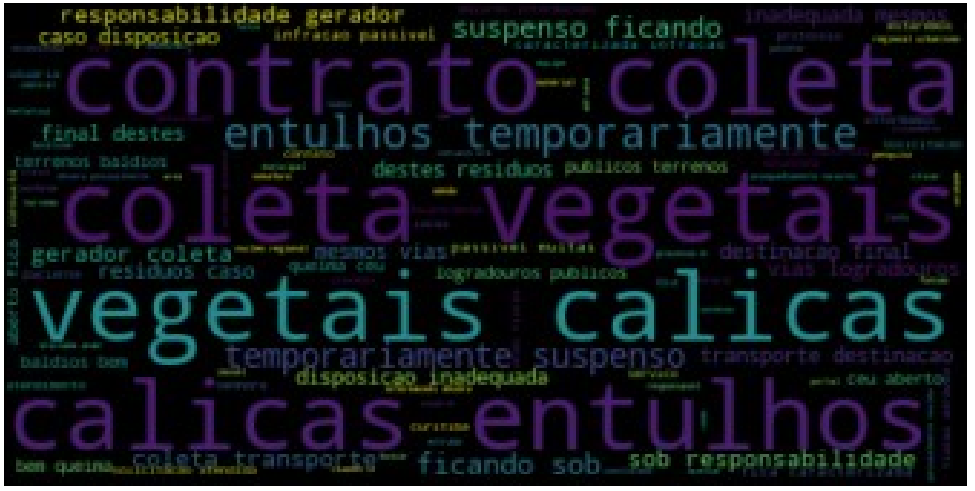


Word Clouds

ALTO DA GLORIA



GUAIRA



Exercícios!

- Revise o conteúdo e faça os exercícios do notebook:
  05d-Modelagem – Texto.ipynb
- Exercícios Extra (bônus):
  - **Modelagem:**
    05x1-Exercício-Modelagem.ipynb
  - **Manipulação:**
    03c1-Exercício-Pandas_Manipulação e Agregação de Dados.ipynb

Referências

- Slides baseados na apresentação “A Gentle Introduction to Machine Learning and Data Mining for the Database Community” por Dr Eamonn Keogh(University of California – Riverside) http://www.cs.ucr.edu/~eamonn/tutorials.html
-