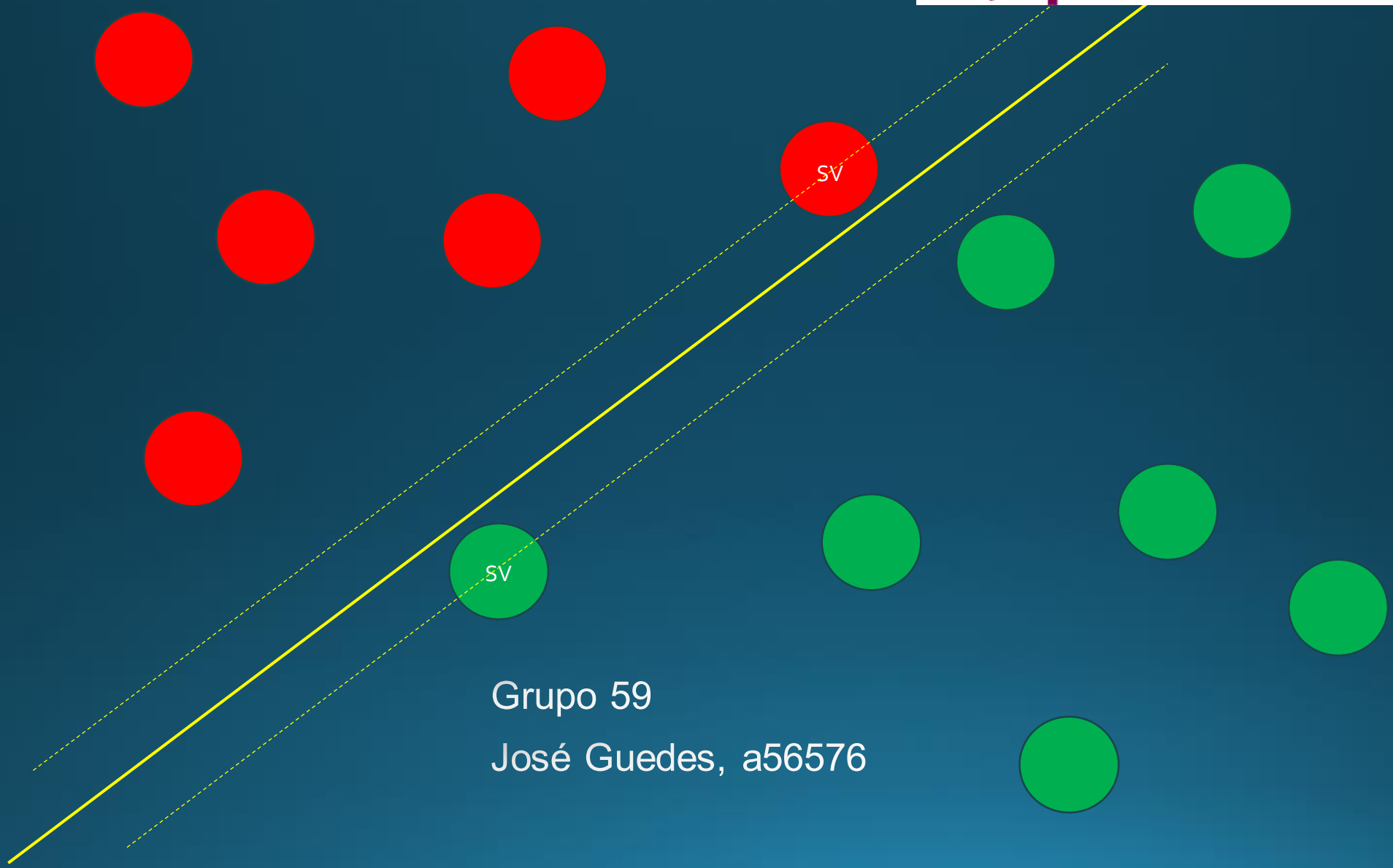
Trabalho prático de Inteligência artificial

INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão



Grupo 59

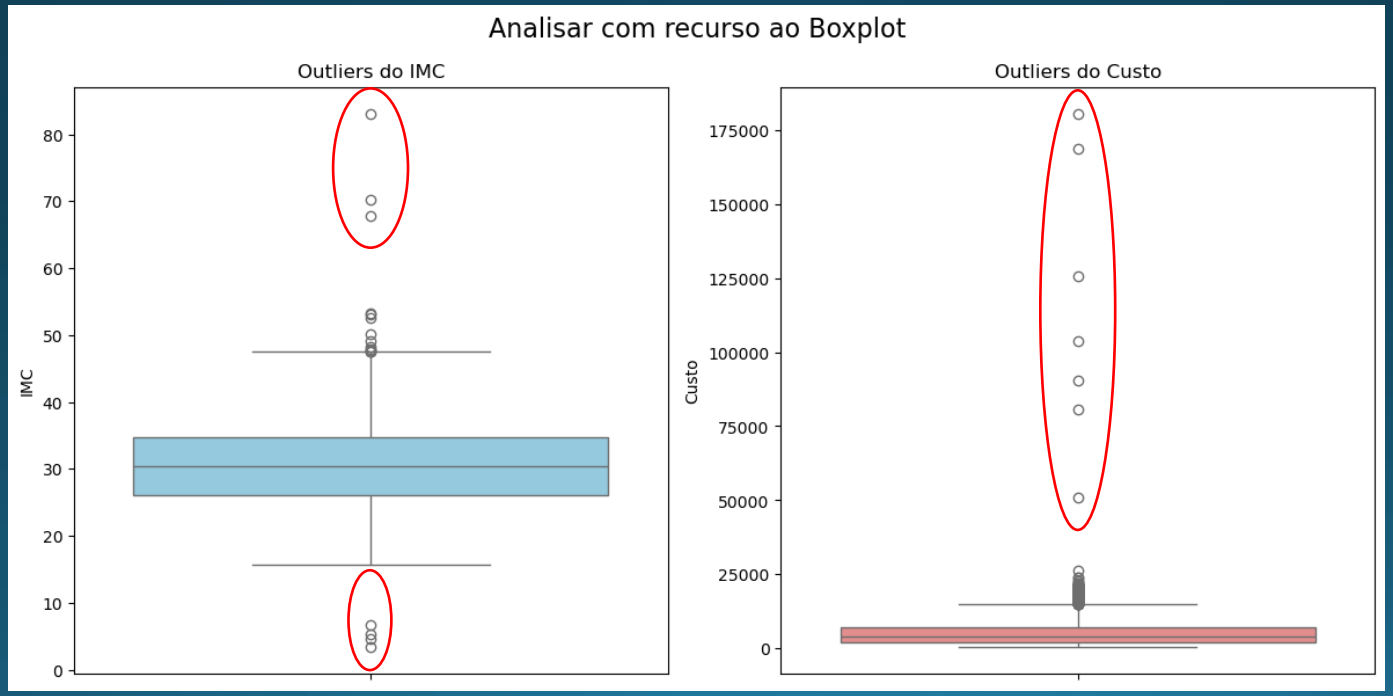
José Guedes, a56576

Metodologia aplicada no trabalho

1. Descarregar e analisar o Dataset
   - Breve análise do Dataset;
   - Visualização gráfica dos dados;
2. Pré-processamento dos dados
   - Verificar possíveis valores duplicados e nulos;
   - Eliminar valores extremos (Outliers);
   - Converter variáveis categóricas em numéricas;
3. Divisão do Dataset
   - Divisão entre conjunto de treino e dados;
4. Resultados de R^2
   - Antes da normalização;
   - Pós normalização;
   - Ajuste dos hiperparâmetros;
5. Analisar a importância das features
6. Resultados previstos
   - Criação do ficheiro com os custos previstos

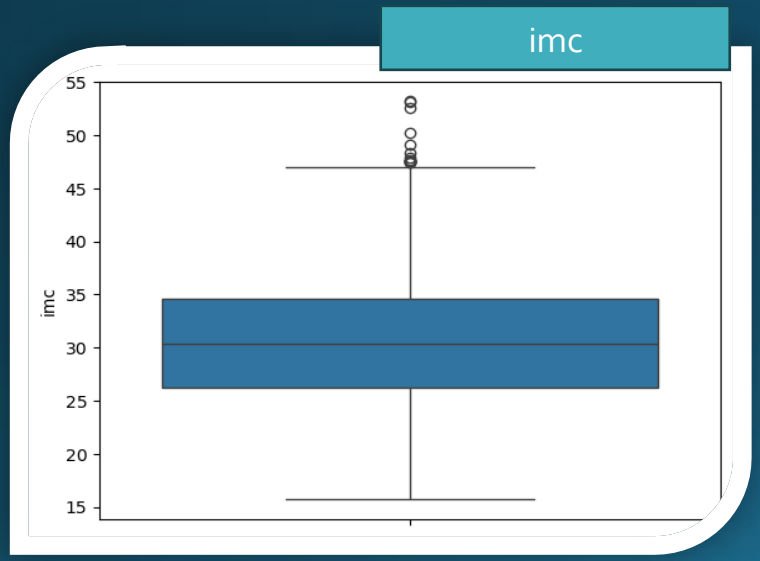
1. Descarregar e analisar o Dataset

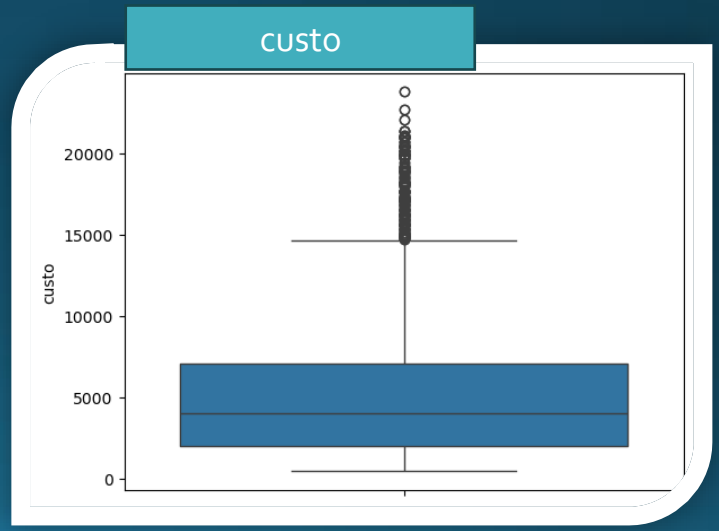
- Para analisar o dataset, comecei por importar o pandas. Depois de ler o dataset para a minha variável **df** utilizei os comandos *df.sample()*, *df.shape* e df.*describe()*, para um breve analise ao dataset.
- Nesta fase também verifiquei a existência de valores nulos e/ou repetidos.
- Através do comando *df.describe()*, foi possível perceber que as colunas **imc** e **custo** tinham indícios de valores atípicos (outliers).
- Para uma melhor compreensão dos outliers recorreu-se a bibliotecas gráficas(seaborn e matplotlib.pyplot).



2. Pré-processamento dos dados

- Esta parte do trabalho foi, sem dúvida, a mais desafiadora e demorada.
- Como anteriormente constatei que o dataset tinha uma row repetida, decidi proceder à eliminação, mantendo a primeira row.
- Decidi utilizar o *método Tukey Fences*, pois apresentou melhores resultados em comparação ao *método Z-score*. Procedi à eliminação dos outliers detetados nos gráficos de *boxplot,* começando pelo imc e depois os do custo. Após diversos e longos testes, optei pelo fator 2.2 para o **imc** e 3.3 para o **custo**. Através destes fatores, menos rigorosos do que o fator padrão 1.5, consegui eliminar 7 outliers no **imc** e 8 outliers no **custo**. Ficando com estes resultados:
- Com a eliminação de 16 rows (15 outliers e 1 valor repetido) o dataset viu o seu tamanho alterado para 2199 rows, o que representa uma perda de 0.72% dos dados, um valor que me parece aceitável.





2. Pré-processamento dos dados (continuação)

- Após a eliminação dos outliers, seguiu-se a conversão das variáveis categóricas para numéricas. As variáveis **genero** e **fumador** foram mapeadas para 0 e 1, visto que eram do tipo binário. Para as demais variáveis categóricas, utilizou-se a técnica *one-hot encoding* com o parâmetro *drop='first'* para evitar redundância.
- Com as variáveis todas no formato numérico, procedeu-se à separação das features do target(custo). Para tal usou-se os seguintes comandos:

```
X = df.drop('custo',axis=1) # Apenas Features
X = np.c_[X, ohe] # Junta as variaveis do OHE

y = df['custo'] # Apenas o target
```

[233] ✓ 0.0s

3. Divisão do Dataset

- Nesta etapa procedi à divisão dos datasets (X e Y) para criar os conjuntos de treino e teste, utilizando a função train_test_split. Isso permitiu separa os dados em duas partes: uma para treinar o modelo e outra para avaliar o seu desempenho, garantindo a GENERALIZAÇÃO dos dados.

```
from sklearn.model_selection import train_test_split
Xtreino,Xteste,ytreino,yteste = train_test_split(X,y,test_size=0.2,random_state=5)
```

[235] ✓ 0.0s

- O test_size indica que 20% dos dados serão usados na fase de teste e 80% serão usados para treinar o modelo.

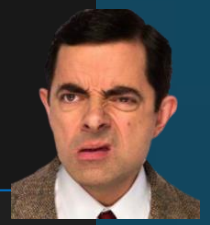
4. Resultados de R^2

- Chegou a altura de treinar o modelo e ver os primeiros resultados.
  - Antes de Normalizar

```
from sklearn.svm import SVR
modelo = SVR()
modelo.fit(Xtreino,ytreino)
modelo.score(Xteste,yteste) #Metrica R2
```

[237] ✓ 0.3s

```
-0.0704299120052283
```



- Humm… este resultado acontece pois SVMs são sensíveis à escala das variáveis. Obrigatoriamente temos de normalizar os nossos dados, decidi usar o StandardScaler, visto que apresentou melhores resultados do que o MinMaxScaler.

```
from sklearn.preprocessing import StandardScaler

scaler_X = StandardScaler()  # Escalador para X (variáveis independentes, Features)
scaler_y = StandardScaler()  # Escalador para y (variável dependente, Target)

#Para os dados de Treino
Xtreino_normalizado = scaler_X.fit_transform(Xtreino)
ytreino_normalizado = scaler_y.fit_transform(ytreino.values.reshape(-1, 1))

#Para os dados de Teste
Xteste_normalizado = scaler_X.transform(Xteste)
yteste_normalizado = scaler_y.transform(yteste.values.reshape(-1, 1))
```

[238] ✓ 0.0s

- Agora com os dados normalizados.

```
modelo = SVR()
modelo.fit(Xtreino_normalizado, ytreino_normalizado.ravel())
modelo.score(Xteste_normalizado, yteste_normalizado) #Agora c
```

[240] ✓ 0.1s

```
0.8596039027172833
```

- Muito melhor, ainda assim com os hiperparâmetros podemos tentar melhorar.

4. Resultados de R^2 (continuação)

- Para ajustar os hiperparâmetros, foi utilizado o *GridSearchCV*, que apresentou melhores resultados. Embora também tenha sido testado o *RandomizedSearchCV,* mas com resultados inferiores.

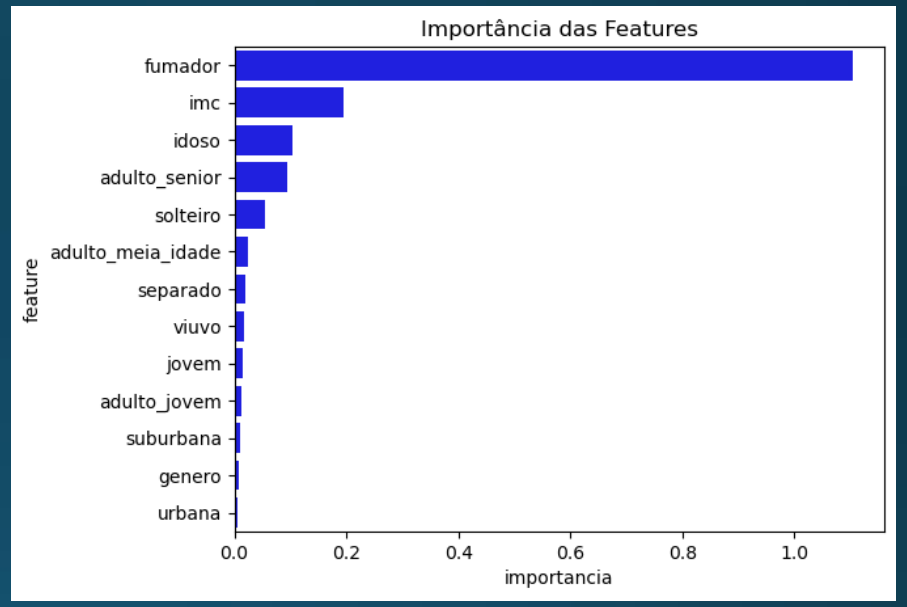
```
[241] from sklearn.model_selection import GridSearchCV

[243] grelha={'C':[4,3,2,1],'gamma':['auto','scale'],'kernel':['rbf','linear']}

[244] procura_modelo = GridSearchCV(modelo, param_grid=grelha,cv=10) #Validaçao cruzada(CV-CrossValidation)

[245] procura_modelo.fit(Xtreino_normalizado,ytreino_normalizado.ravel())

[246] procura_modelo.best_params_
... {'C': 2, 'gamma': 'scale', 'kernel': 'rbf'}

[247] modelo_otimo=procura_modelo.best_estimator_

[248] modelo_otimo.score(Xteste_normalizado,yteste_normalizado)
... 0.8652088733698249
```

Visto que estamos num problema de regressão, um R^2 de 0.86 indica que o modelo explica 86% da variabilidade da variável custo(target) com base nas variáveis independentes(features), evidenciando uma alta correlação entre as features e o target.

- Criou-se um modelo ótimo como um R^2 de 0.86, e será este o modelo que iremos usar para prever os custos.

5. Analisar a importância das features

- Para perceber as caraterísticas que mais influência têm nas despesão de um cliente usou-se a função permutation_importance do sklearn.
- As features que mais contribuem para o custo são:
  - **Fumador;**
  - **Imc;**
  - **Idoso;**
  - **Adulto sénior;**
  - **Solteiro;**



6. Resultados previstos

- Por fim importou-se o dataset just_features.csv e utilizou-se o modelo ótima para fazer a previsão dos valores. Criando no fim o ficheiro grupo59_custos_estimados.csv com os custos previstos.

FIM