Tarefa de Regressão - parte 1

Nesta tarefa, você deve carregar um dataset sobre tratores , construir modelos de Regressão com os algoritmos vistos em aula e predizer o preço de venda (SalesPrice).

Dica: Para toda a tarefa, além da biblioteca pandas e numpy, você pode querer explorar funções da biblioteca sklearn.ensemble (em particular o pacote RandomForestRegressor), sklearn.neighbors (KNeighborsRegressor) e sklearn.tree (DecisionTreeRegressor). Além disso, você vai precisar usar funções de pré-processamento e de pós-procesamento (das bibliotecas sklearn.preprocessing, sklearn.model_selection e sklearn.metrics)

IMPORTANTE: Ao realizar etapas de pré-processamento, verifique se o procedimento funcionou.

Importe os pacotes e carregue o arquivo com os dados

O dataset a ser utilizado encontra-se no arquivo **Tratores.csv**, disponível no EAD.

Este dataset contém dados sobre as vendas de tratores, descritas pelos seguintes atributos/variáveis:

- SalesID: unique identifier of a particular sale of a machine at auction
- MachineID: dentifier for a particular machine; machines may have multiple sales
- ModelID: identifier for a unique machine model
- YearMade:year of manufacturer of the Machine
- MachineHoursCurrentMeter: current usage of the machine in hours at time of sale (saledate); null or 0 means no hours have been reported for that sale
- Saledate: time of sale
- Product Group: Identifier for top-level hierarchical grouping of fiModelDesc
- **Saleprice (target)**:cost of sale in USD

```
import pandas as pd
import sklearn
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.preprocessing import MinMaxScaler, OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from datetime import datetime
from sklearn.model_selection import train_test_split
from sklearn.tree import plot_tree
```

```
df = pd.read_csv('Tratores.csv',sep=',')
df.head()
```

	SalesID	SalePrice	MachineID	ModelID	YearMade	MachineHoursCurrentMeter	salec
0	1139246	66000	999089	3157	2004	68	11/16/2
1	1139248	57000	117657	77	1996	4640	3/26/2
2	1139249	10000	434808	7009	2001	2838	2/26/2

Next steps: Generate code with df View recommended plots

Pré-processe a base de dados

Dica: avalie a necessidade de converter os tipos das variáveis, normalizar os dados, ...

Transforme a variável saledate em três outras variáveis: ano, mês e dia. Adicione-as como colunas no dataset

Dica: Uma passo inicial pode ser transformar a variável em *datetime*.

```
df['saledate'] = pd.to_datetime(df['saledate'])
df['Ano'] = df['saledate'].dt.year
df['Mês'] = df['saledate'].dt.month
df['Dia'] = df['saledate'].dt.day

df.head()
```

	SalesID	SalePrice	MachineID	ModelID	YearMade	MachineHoursCurrentMeter	saleda
0	1139246	66000	999089	3157	2004	68	2006-
1	1139248	57000	117657	77	1996	4640	2004-
2	1139249	10000	434808	7009	2001	2838	2004-
3	1139251	38500	1026470	332	2001	3486	2011-
4	1139253	11000	1057373	17311	2007	722	2009-

Next steps: Generate code with df View recommended plots

Implemente outras etapas de pré-processamento que julgue necessárias.

```
missing_values = df.isnull().sum()

print("Valores ausentes em cada coluna:")
print(missing_values)
```

```
Valores ausentes em cada coluna:
SalesID                     0
SalePrice                   0
MachineID                   0
ModelID                     0
YearMade                    0
MachineHoursCurrentMeter    0
saledate                    0
ProductGroup                0
Ano                         0
Mês                         0
Dia                         0
dtype: int64
```

```
#normalizando colunas
scaler = MinMaxScaler()
df[['SalePrice', 'YearMade','MachineHoursCurrentMeter']] = scaler.fit_transform(df[['SalePrice', 'YearMade', 'MachineH
```

```
df = pd.get_dummies(df, columns=['ProductGroup'])
```

```
df = df.replace({True: 1, False: 0})
print(df)
```

```
     SalesID  SalePrice  MachineID  ModelID  YearMade  \
0    1139246   0.449541     999089     3157  0.904762
1    1139248   0.383486     117657       77  0.714286
2    1139249   0.038532     434808     7009  0.833333
3    1139251   0.247706    1026470      332  0.833333
4    1139253   0.045872    1057373    17311  0.976190
..       ...        ...        ...      ...       ...
995  1142566   0.207339    1069733     5428  0.928571
996  1142567   0.420183     531918    23162  0.928571
997  1142568   0.023853    1064508    17472  0.761905
998  1142577   0.082569    1046210    13391  0.904762
999  1142582   0.071560    1031625     9578  0.952381

     MachineHoursCurrentMeter   saledate   Ano  Mês  Dia  ProductGroup_BL  \
0                    0.001799 2006-11-16  2006   11   16                0
1                    0.122764 2004-03-26  2004    3   26                0
2                    0.075087 2004-02-26  2004    2   26                0
3                    0.092232 2011-05-19  2011    5   19                0
4                    0.019103 2009-07-23  2009    7   23                0
..                        ...        ...   ...  ...  ...              ...
995                  0.099455 2009-07-16  2009    7   16                0
996                  0.034131 2007-06-14  2007    6   14                0
997                  0.049344 2005-09-22  2005    9   22                0
998                  0.022516 2005-07-28  2005    7   28                0
999                  0.072759 2011-06-16  2011    6   16                0

     ProductGroup_MG  ProductGroup_SSL  ProductGroup_TEX  ProductGroup_TTT  \
0                  0                 0                 0                 0
1                  0                 0                 0                 0
2                  0                 1                 0                 0
3                  0                 0                 1                 0
4                  0                 1                 0                 0
..               ...               ...               ...               ...
995                0                 0                 1                 0
996                0                 0                 1                 0
997                0                 1                 0                 0
998                0                 0                 1                 0
999                0                 1                 0                 0

     ProductGroup_WL
0                  1
1                  1
2                  0
3                  0
4                  0
..               ...
995                0
996                0
997                0
998                0
999                0

[1000 rows x 16 columns]
```

Crie os conjuntos de treinamento e de teste

Atenção: Selecione aleatoriamente e sem reposição (para que não se repitam) 75% das observações para o conjunto de treinamento. As 25% observações restantes serão usadas para o conjunto de teste. Fixe a semente de geração de dados aleatórios.

```
train_df, test_df = train_test_split(df, test_size=0.25, random_state=50)

print("Tamanho do conjunto (treinamento):", len(train_df))
print("Tamanho do conjunto (teste):", len(test_df))
```

```
Tamanho do conjunto (treinamento): 750
Tamanho do conjunto (teste): 250
```

Construa modelos de KNN, Árvore para Regressão e Random Forest.

Utilizando cada um deles, faça a predição do atributo SalePrice no conjunto teste.

```
#knn
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error

# separando features e o target
X_train_knn = train_df.drop(columns=['SalePrice', 'saledate'])
y_train_knn = train_df['SalePrice']
X_test_knn = test_df.drop(columns=['SalePrice', 'saledate'])

knn_model = KNeighborsRegressor()
knn_model.fit(X_train_knn, y_train_knn)

# predições
y_pred_knn = knn_model.predict(X_test_knn)
```

```
print(y_pred_knn)
```

```
[0.29541284 0.24183486 0.20440367 0.26018349 0.20366972 0.29908257
 0.36733945 0.0293578  0.06715596 0.15963303 0.09357798 0.11119266
 0.19119266 0.10605505 0.39082569 0.19486239 0.24183486 0.10091743
 0.27266055 0.28146789 0.18165138 0.22862385 0.25284404 0.10605505
 0.20256881 0.20733945 0.2359633  0.09798165 0.0187156  0.27706422
 0.20366972 0.09394495 0.20513761 0.39229358 0.42752294 0.21981651
 0.29174312 0.23743119 0.18311927 0.07302752 0.25027523 0.20587156
 0.14422018 0.23522936 0.42752294 0.2866055  0.07522936 0.06201835
 0.20733945 0.29541284 0.15522936 0.23522936 0.37027523 0.10348624
 0.22055046 0.09724771 0.26311927 0.2        0.24550459 0.14422018
 0.04807339 0.05908257 0.16183486 0.4906422  0.29321101 0.13174312
 0.17798165 0.26385321 0.08954128 0.12807339 0.26311927 0.2146789
 0.10605505 0.16550459 0.39229358 0.17504587 0.18825688 0.20366972
 0.14311927 0.24917431 0.09651376 0.18899083 0.26018349 0.2293578
 0.25431193 0.20733945 0.15963303 0.14862385 0.14422018 0.24146789
 0.31963303 0.30055046 0.19633028 0.15009174 0.24550459 0.18972477
 0.09981651 0.20366972 0.19559633 0.12880734 0.14568807 0.27266055
 0.16550459 0.24844037 0.17798165 0.19412844 0.07669725 0.13027523
 0.1266055  0.14201835 0.20366972 0.19633028 0.1933945  0.13321101
 0.24330275 0.15889908 0.1493578  0.04917431 0.27486239 0.26311927
 0.11816514 0.36220183 0.11853211 0.1412844  0.08697248 0.11669725
 0.0466055  0.12513761 0.14605505 0.09284404 0.20807339 0.30422018
 0.28880734 0.17504587 0.18899083 0.13100917 0.14055046 0.2
 0.24917431 0.12073394 0.08330275 0.09577982 0.25211009 0.16110092
 0.08088073 0.35266055 0.2733945  0.23229358 0.17944954 0.11706422
 0.1133945  0.14422018 0.13100917 0.36440367 0.21174312 0.23082569
 0.1412844  0.36366972 0.07889908 0.36807339 0.17357798 0.20770642
 0.16697248 0.31963303 0.31449541 0.24550459 0.11045872 0.32183486
 0.08088073 0.14715596 0.1346789  0.20513761 0.21908257 0.21981651
 0.16550459 0.15743119 0.17211009 0.10862385 0.27853211 0.0187156
 0.35706422 0.15669725 0.16550459 0.0546789  0.1346789  0.06055046
 0.29247706 0.07376147 0.26018349 0.26311927 0.23669725 0.31669725
 0.19412844 0.33211009 0.27266055 0.22275229 0.24073394 0.13321101
 0.05321101 0.14788991 0.14715596 0.19926606 0.16183486 0.09944954
 0.30495413 0.19559633 0.12256881 0.04146789 0.16036697 0.26385321
 0.2066055  0.10348624 0.14788991 0.36807339 0.27486239 0.16917431
 0.20807339 0.11706422 0.29174312 0.10055046 0.18972477 0.09284404
 0.22642202 0.25724771 0.3240367  0.23229358 0.08770642 0.20733945
 0.27779817 0.16550459 0.13908257 0.23376147 0.15522936 0.19192661
 0.11486239 0.25247706 0.29174312 0.05834862 0.21981651 0.2
 0.14018349 0.23816514 0.20440367 0.17137615 0.23155963 0.18972477
 0.15963303 0.14422018 0.20587156 0.18605505]
```

```
#arvore de regressao
from sklearn.tree import DecisionTreeRegressor

dt_model = DecisionTreeRegressor(random_state=42)
dt_model.fit(X_train_knn, y_train_knn)

# predições
y_pred_dt = dt_model.predict(X_test_knn)
```

```
print(y_pred_dt)
```

```
[0.15963303 0.11559633 0.00917431 0.14862385 0.17798165 0.43486239
 0.42752294 0.05688073 0.66972477 0.09724771 0.08256881 0.10458716
 0.05688073 0.15229358 0.48623853 0.2733945  0.37614679 0.07889908
 0.17431193 0.28807339 0.36146789 0.2440367  0.24770642 0.56697248
 0.25137615 0.59633028 0.29541284 0.56697248 0.08623853 0.14862385
 0.42752294 0.07889908 0.04587156 0.44954128 0.10091743 0.25137615
 0.26605505 0.05321101 0.83119266 0.43486239 0.06055046 0.05321101
 0.02752294 0.07155963 0.17431193 0.06422018 0.44220183 0.04587156
 0.58899083 0.05321101 0.76513761 0.16330275 0.47889908 0.08990826
 0.03853211 0.45688073 0.42018349 0.1853211  0.20733945 0.23669725
 0.2        0.04587156 0.07522936 0.12293578 0.19633028 0.08256881
 0.17798165 0.10091743 0.05321101 0.16330275 0.14495413 0.1412844
 0.1853211  0.66238532 0.2733945  0.07889908 0.02385321 0.05688073
 0.20733945 0.1266055  0.79449541 0.1266055  0.13394495 0.25137615
 0.2293578  0.65504587 0.08256881 0.2        0.18899083 0.16330275
 0.08990826 0.04220183 0.14862385 0.24770642 0.08990826 0.2440367
 0.35412844 0.32477064 0.14862385 0.15963303 0.08990826 0.2293578
 0.05321101 0.36146789 0.26605505 0.2733945  0.2        0.03853211
 0.15963303 0.03853211 0.1266055  0.40550459 0.17431193 0.07889908
 0.33211009 0.30275229 0.04954128 0.06422018 0.2587156  0.23669725
 0.07522936 0.33211009 0.05137615 0.16330275 0.04587156 0.04587156
 0.15229358 0.23669725 0.2293578  0.2293578  0.42752294 0.2587156
 0.23669725 0.04220183 0.25137615 0.15229358 0.2293578  0.11559633
 0.08990826 0.19633028 0.11559633 0.05504587 0.33211009 0.88990826
 0.06422018 0.2587156  0.30275229 0.06788991 0.04587156 0.06055046
 0.15229358 0.46422018 0.10825688 0.38348624 0.07522936 0.1706422
 0.05688073 0.28073394 0.04220183 0.13394495 0.28073394 0.02018349
 0.2        0.34678899 0.07522936 0.06788991 0.09724771 0.2293578
 0.38348624 0.30275229 0.14495413 0.14862385 0.17431193 0.39816514
 0.04844037 0.05688073 0.2293578  0.02568807 0.39816514 0.02752294
 0.2293578  0.35412844 0.31743119 0.03486239 0.05504587 0.03486239
 0.2293578  0.03853211 0.2293578  0.29541284 0.10458716 0.30275229
 0.04587156 0.39816514 0.11926606 0.07889908 0.67706422 0.04220183
 0.05688073 0.19266055 0.17431193 0.03302752 0.38348624 0.23669725
 0.39816514 0.31743119 0.07889908 0.0440367  0.38348624 0.2587156
 0.32110092 0.1412844  0.06788991 0.08990826 0.35412844 0.16330275
 0.2        0.38348624 0.83119266 0.31743119 0.31009174 0.05321101
 0.31743119 0.02752294 0.18899083 0.04954128 0.02752294 0.29541284
 0.20733945 0.39816514 0.03853211 0.1853211  0.32110092 0.08990826
 0.26605505 0.2146789  0.10091743 0.44954128 0.07889908 0.35412844
 0.2293578  0.05321101 0.33211009 0.1706422  0.2587156  0.08256881
 0.17798165 0.08623853 0.49357798 0.44954128]
```

```
#random forest
from sklearn.ensemble import RandomForestRegressor

# Construindo e treinando o modelo Random Forest
rf_model = RandomForestRegressor(random_state=42)
rf_model.fit(X_train_knn, y_train_knn)

# Fazendo predições no conjunto de teste
y_pred_rf = rf_model.predict(X_test_knn)
```

```
print(y_pred_rf)
```

```
[0.22176147 0.11080734 0.02831193 0.23207339 0.29658716 0.34649541
 0.37009174 0.05979817 0.37622018 0.19056881 0.18113761 0.23027523
 0.06411009 0.20381651 0.38895413 0.36458716 0.29491743 0.21126606
 0.18744954 0.17633028 0.4172844  0.37416514 0.30245872 0.39691743
 0.34062385 0.42372477 0.29930275 0.37825688 0.07326606 0.38645872
 0.27283303 0.04337615 0.15658716 0.31155963 0.0693211  0.35933945
 0.32649541 0.06625688 0.6866789  0.44005505 0.22387156 0.09760367
 0.02517431 0.12645872 0.19834862 0.04699083 0.31823853 0.07557798
 0.31266055 0.06423853 0.41581651 0.19155963 0.34187156 0.24590826
 0.06517431 0.22897248 0.22286239 0.26453211 0.17691743 0.40422018
 0.25922936 0.30388991 0.07618349 0.25144954 0.19111927 0.06998165
 0.32363303 0.29544954 0.02715596 0.17023853 0.14565138 0.20344954
 0.20884404 0.38433028 0.36029358 0.11594495 0.0440367  0.04886239
 0.18012844 0.14322936 0.40607339 0.1586789  0.16440367 0.22722936
 0.21992661 0.3493945  0.09394495 0.23363303 0.2226422  0.25463486
 0.19812844 0.05150459 0.3027156  0.23726606 0.21904587 0.21750459
 0.24495413 0.32143119 0.27765138 0.17677064 0.05691743 0.17295413
 0.04222018 0.43950459 0.20979817 0.32495413 0.14682202 0.06205505
 0.21545688 0.10414679 0.10686239 0.24361468 0.13607339 0.04902752
 0.30776147 0.20458716 0.04486239 0.04704587 0.30106422 0.23889541
 0.18350459 0.22685505 0.10018349 0.33258716 0.09724771 0.09077064
```

```
 0.22715596 0.24763303 0.17111927 0.18422018 0.21882569 0.2546422
 0.25405505 0.03519266 0.35255046 0.15677064 0.18234862 0.13466789
 0.11761468 0.33974312 0.20622018 0.03699083 0.30216514 0.68201835
 0.22016514 0.20612844 0.37310092 0.06433028 0.03357798 0.05717431
 0.20165138 0.32565138 0.10552294 0.2519633  0.05693578 0.17711927
 0.1113211  0.29144954 0.04992661 0.16616514 0.19777982 0.02143119
 0.19954128 0.37574312 0.172      0.09627523 0.0779633  0.29669725
 0.28752294 0.26023853 0.22605505 0.26168807 0.20623853 0.18110092
 0.13781284 0.03774312 0.1733211  0.19809174 0.31225688 0.02387156
 0.40807339 0.2986789  0.34781651 0.09016514 0.04242202 0.05926606
 0.18500917 0.03455046 0.23688073 0.3107156  0.168      0.32763303
 0.13552294 0.25631193 0.30344954 0.13295413 0.41106422 0.03752294
 0.03889908 0.21611009 0.20088073 0.12992661 0.33981651 0.25064954
 0.1519633  0.30069725 0.09144954 0.09282569 0.24306422 0.2933578
 0.25188991 0.21574312 0.0607156  0.08247706 0.18554128 0.26577982
 0.19394495 0.27633028 0.68286239 0.24077064 0.21761468 0.05416514
 0.23900917 0.07849541 0.17009174 0.11633028 0.028      0.37836697
 0.31310092 0.16844037 0.05023853 0.35974312 0.25075229 0.08517431
 0.28062385 0.23295413 0.29926606 0.24840367 0.12851376 0.26308257
 0.25976147 0.06212844 0.38811009 0.12764037 0.21366972 0.11950459
 0.25768807 0.12809174 0.39651376 0.28216514]
```

Pós-processamento: Avalie cada modelo de regressão

Calcule as medidas de desempenho vistas em aula (raiz do erro quadrático médio, R2)

```
# knn --> erro quadrático médio (MSE) no conjunto de teste
mse_knn = mean_squared_error(test_df['SalePrice'], y_pred_knn)
print("Erro quadrático médio (MSE) -> KNN:", mse_knn)

# arvore de decisao --> erro quadrático médio (MSE) no conjunto de teste
mse_dt = mean_squared_error(test_df['SalePrice'], y_pred_dt)
print("MSE-> Árvore de Decisão:", mse_dt)

# random forest --> erro quadrático médio (MSE) no conjunto de teste
mse_rf = mean_squared_error(test_df['SalePrice'], y_pred_rf)
print("MSE -> Random Forest:", mse_rf)
```

```
Erro quadrático médio (MSE) -> KNN: 0.03860644367039811
MSE-> Árvore de Decisão: 0.019336321993098225
MSE -> Random Forest: 0.015984602812019185
```

```
from sklearn.metrics import r2_score

# raiz quadrada do erro quadrático médio (RMSE)
rmse_knn = np.sqrt(mse_knn)
rmse_dt = np.sqrt(mse_dt)
rmse_rf = np.sqrt(mse_rf)

print("Raiz do Erro Quadrático Médio (RMSE) -->  KNN:", rmse_knn)
print("Raiz do Erro Quadrático Médio (RMSE) -->  Árvore de Decisão:", rmse_dt)
print("Raiz do Erro Quadrático Médio (RMSE) --> Random Forest:", rmse_rf)
```

```
Raiz do Erro Quadrático Médio (RMSE) -->  KNN: 0.19648522506895552
Raiz do Erro Quadrático Médio (RMSE) -->  Árvore de Decisão: 0.13905510416053854
Raiz do Erro Quadrático Médio (RMSE) --> Random Forest: 0.12643022902778903
```

```
#  coeficiente de determinação (R²)
r2_knn = r2_score(test_df['SalePrice'], y_pred_knn)
r2_dt = r2_score(test_df['SalePrice'], y_pred_dt)
r2_rf = r2_score(test_df['SalePrice'], y_pred_rf)

print("Coeficiente de Determinação (R²) --> KNN:", r2_knn)
print("Coeficiente de Determinação (R²) --> Árvore de Decisão:", r2_dt)
print("Coeficiente de Determinação (R²) --> Random Forest:", r2_rf)
```

```
Coeficiente de Determinação (R²) --> KNN: -0.07620418814807217
Coeficiente de Determinação (R²) --> Árvore de Decisão: 0.46097519652378294
Coeficiente de Determinação (R²) --> Random Forest: 0.5544086723178586
```

Qual modelo apresentou melhor desempenho segundo as métricas calculadas?

O Erro Quadrático Médio (MSE) é uma métrica que mede a média dos quadrados dos erros. Logo, quanto menor o RMSE, melhor o desempenho do modelo. Além disso, quanto maior o R^2 , melhor a capacidade de previsão do modelo em relação aos outros. Sendo assim, como o Random Forest tem menor RMSE e maior R^2, ele possui melhor desempenho.

Avalie a importância dos atributos (feature importances) na construção dos modelos de Árvore de Decisão e Random Forest Regressor.

Diga os três atributos que apresentaram maior relevância na predição de cada modelo.

Feature Importances: AR

```
importances_dt = dt_model.feature_importances_

importances_dt_df = pd.DataFrame({'atributo': X_train_knn.columns, 'importância': importances_dt})
importances_dt_df = importances_dt_df.sort_values(by='importância', ascending=False)

print(" três atributos mais importantes --> Árvore de Decisão :")
importances_dt_df.head(3)
```

```
 três atributos mais importantes --> Árvore de Decisão :
```

	atributo	importância
4	MachineHoursCurrentMeter	0.195782
10	ProductGroup_SSL	0.166642
3	YearMade	0.124379

Next steps: Generate code with `importances_dt_df` View recommended plots

Feature Importances Random forest

```
importances_rf = rf_model.feature_importances_

importances_rf_df = pd.DataFrame({'atributo': X_train_knn.columns, 'importância': importances_rf})
importances_rf_df = importances_rf_df.sort_values(by='importância', ascending=False)

print(" três atributos mais importantes --> Random Forest:")
importances_rf_df.head(3)
```

```
 três atributos mais importantes --> Random Forest:
```

	atributo	importância
10	ProductGroup_SSL	0.165323
4	MachineHoursCurrentMeter	0.146476
2	ModelID	0.136366

Next steps: Generate code with `importances_rf_df` View recommended plots

Elabore um gráfico para visualizar a árvore de regressão, utilizando a biblioteca dtreeviz (precisa ser instalada). (EXTRA)

Para isso, treinem novamente o modelo de árvore de regressão, estipulando como parâmetro de máxima profundidade da árvore (max_depth) um número até 4.

Essa visualização é muito interessante e nos mostra a distribuição do atributo de decisão em cada nó e a distribuição e a média da resposta da folha.

```
!pip install -U dtreeviz
import dtreeviz
```

```
Requirement already satisfied: dtreeviz in /usr/local/lib/python3.10/dist-packages (2.2.2)
Requirement already satisfied: graphviz>=0.9 in /usr/local/lib/python3.10/dist-packages (from dtreeviz) (0.20.3)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from dtreeviz) (2.0.3)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from dtreeviz) (1.25.2)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (from dtreeviz) (1.2.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from dtreeviz) (3.7.1)
Requirement already satisfied: colour in /usr/local/lib/python3.10/dist-packages (from dtreeviz) (0.1.5)
Requirement already satisfied: pytest in /usr/local/lib/python3.10/dist-packages (from dtreeviz) (7.4.4)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->dtree
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->dtreeviz)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->dtre
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->dtre
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->dtreev
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->dtreeviz
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->dtree
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib->d
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas->dtreeviz) (20
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas->dtreeviz) (
Requirement already satisfied: iniconfig in /usr/local/lib/python3.10/dist-packages (from pytest->dtreeviz) (2.0.0
Requirement already satisfied: pluggy<2.0,>=0.12 in /usr/local/lib/python3.10/dist-packages (from pytest->dtreeviz
Requirement already satisfied: exceptiongroup>=1.0.0rc8 in /usr/local/lib/python3.10/dist-packages (from pytest->d
Requirement already satisfied: tomli>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from pytest->dtreeviz) (2.
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from scikit-learn->dtreevi
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn->dtreev
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn-
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->mat
```

```
from sklearn.datasets import make_regression
from sklearn.tree import DecisionTreeRegressor
from dtreeviz.trees import *

# não produzir warning "Arial font not found warnings"
import logging
logging.getLogger('matplotlib.font_manager').setLevel(level=logging.CRITICAL)


# separando o dataset em treino e teste
train_df, test_df = train_test_split(df, test_size=0.25, random_state=50)

X_train_dt = train_df.drop(columns=['SalePrice', 'saledate'])
y_train_dt = train_df['SalePrice']
X_test_dt = test_df.drop(columns=['SalePrice', 'saledate'])

#parâmetro de máxima profundidade da árvore (max_depth) um número até 4
max_depth = 4
dt_model = DecisionTreeRegressor(max_depth=max_depth, random_state=42)
dt_model.fit(X_train_dt, y_train_dt)

# Visualização do gráfico
viz = dtreeviz.model(
    dt_model,
    X_train_dt,
    y_train_dt,
    target_name='SalePrice',
    feature_names=X_train_dt.columns

)
```

```
)

viz.view()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but
```