Exercício 1 - Preços de computadores - Regressão

Referência

Um grupo muito grande de investidores deseja entrar no mercado de produção de computadores pessoais. Como o projeto começará do zero, foi reunida uma base com tipos de equipamentos já existentes e seus respectivos preços. Os investidores desejam obter um modelo para que, através dele, possam ter estimativas de preço de um possível novo produto, dada as suas características.

Inicialmente, o grupo de investidores deseja explorar combinações de tamanho e resolução da tela, versão do CPU e o peso do equipamento. Através do dataset 'laptop_price.csv', faça uma análise de quais são as variáveis mais importantes para a definição do preço e desenvolva um modelo que receba características do aparelho como entrada e tenha o preço final como saída. Você tem liberdade para propor um outro conjunto de variáveis.

Ao final do notebook, reporte um pequeno resumo dos resultados que obteve, e indique se a estratégia desejada pelos investidores é viável ou não.

Dicas:

- Tome cuidado com a variável "Company", estamos criando uma nova empresa;
- Lembre-se de separar a base de teste antes de começar os estudos;
- Você pode utilizar, para medir a performance do modelo de regressão, as métricas MSE/RMSE;

In [94]:
```
import pandas as pd

df = pd.read_csv('laptop_price.csv')

print(df.shape)

df.head()
```

```
(1303, 13)
```

Out[94]:

	laptop_ID	Company	Product	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price_euros
0	1	Apple	MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8GB	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37kg	1339.69
1	2	Apple	Macbook Air	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8GB	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34kg	898.94
2	3	HP	250 G6	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8GB	256GB SSD	Intel HD Graphics 620	No OS	1.86kg	575.00
3	4	Apple	MacBook Pro	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16GB	512GB SSD	AMD Radeon Pro 455	macOS	1.83kg	2537.45
4	5	Apple	MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8GB	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37kg	1803.60

Avaliação inicial do Dataset

laptop_ID: Chave única para cada modelo. Não faz sentido adicionar no modelo.
Company: Sabidamente Apple pratica preços mais altos. Minha decisão é substituir por uma variável categórica com 1 para Apple e 0 para outras marcas.
Product: O nome do produto não influencia o preço. Será desconsiderada para o modelo.
TypeName: Característica importante. Deve ser convertida para numérica.
Inches: Tamanho da tela. Influencia o preço. Será mantida.
ScreenResolution: Importante, e temos que preprocessar para obter um fator numérico. Para tanto, criarei uma função para extrair os dois números, que serão então multiplicados com o tamanho de tela para obter a qualidade de tela final.
Cpu: É relevante, com chips mais baratos e caros. Alguns ajustes "De x Para" precisam ser realizados:
"Intel Celeron Dual Core N3060 1.60GHz" para "Intel Celeron Dual Core N3060 1.6GHz"
"Intel Core M m7-6Y75 1.2GHz" para "Intel Core M M7-6Y75 1.2GHz"
"Intel Core i3 6006U 2GHz" para "Intel Core i3 6006U 2.0GHz"
"Intel Core i5 7200U 2.50GHz" para "Intel Core i5 7200U 2.5GHz" "Intel Core i5 7200U 2.70GHz" para "Intel Core i5 7200U 2.7GHz"
"Intel Core i7 6500U 2.50GHz" para "Intel Core i7 6500U 2.5GHz"
São diferentes designações para uma mesma CPU.
Ram: É relevante. Temos que converter em numérica, removendo o sufixo 'GB'. O modelo tem que refletir que mais memória indica preço mais alto, em geral.
Memory: Indica o total de armazenamento em disco/SSD. Um pré-processamento precisa ser construído para as tarefas:

```
* Obter valores para Flash Storage, HDD (disco rotativo), SSD e Hybrid, unificando o tamanho para cada categoria em GB.
* Substituir a coluna textual "Memory" pelas quatro numéricas novas.
```

Gpu: Temos que manter e tornar numérica, é muito relevante para fins de preço. Alguns "De" x "Para" são necessários:
"AMD Radeon R2 Graphics" para "AMD Radeon R2"
"AMD Radeon R4 Graphics" para "AMD Radeon R4"

"AMD Radeon R7 Graphics" para "AMD Radeon R7"
"Intel Graphics 620" para "Intel HD Graphics 620"
"Intel HD Graphics 620 " para "Intel HD Graphics 620"
"Nvidia GeForce 920MX " para "Nvidia GeForce 920MX"
"Nvidia GeForce 930MX " para "Nvidia GeForce 930MX"
"Nvidia GeForce GTX 1050 Ti" para "Nvidia GeForce GTX 1050Ti"
"Nvidia GeForce GTX 960<U+039C>" para "Nvidia GeForce GTX 960"
"Nvidia GeForce GTX1050 Ti" para "Nvidia GeForce GTX 1050Ti"
"Nvidia GeForce GTX1060" para "Nvidia GeForce GTX 1060"
"Nvidia GeForce GTX1080" para "Nvidia GeForce GTX 1080"
"Nvidia GeForce MX130" para "Nvidia GeForce 130MX"
"Nvidia GeForce MX150" para "Nvidia GeForce 150MX"

OpSys: Sistema Operacional. Devemos manter, convertendo em numérica.

Weight: Como os investidores querem essa variável, temos que converter em numérica.

Price_euros: Variável-alvo. Vamos converter em Reais ao valor de 6,28 - cotação de 08/12/2021.

In [95]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1303 entries, 0 to 1302
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   laptop_ID         1303 non-null   int64  
 1   Company           1303 non-null   object 
 2   Product           1303 non-null   object 
 3   TypeName          1303 non-null   object 
 4   Inches            1303 non-null   float64
 5   ScreenResolution  1303 non-null   object 
 6   Cpu               1303 non-null   object 
 7   Ram               1303 non-null   object 
 8   Memory            1303 non-null   object 
 9   Gpu               1303 non-null   object 
 10  OpSys             1303 non-null   object 
 11  Weight            1303 non-null   object 
 12  Price_euros       1303 non-null   float64
dtypes: float64(2), int64(1), object(10)
memory usage: 132.5+ KB
```

Não há valores faltantes

Tratamento do DataFrame

Separação da variável-alvo (y) das variáveis para predição (X)

Já removendo as colunas indesejadas do DataFrame de predição.

Já separando os dados de treino e validação.

In [96]:
```
from sklearn.model_selection import train_test_split
```

In [97]:
```
var_alvo = 'Price_euros'
```

In [98]:
```
# Primeiro vamos converter os valores de Euros para Reais
df[var_alvo] = df[var_alvo] * 6.28
```

In [99]:
```
y = df[var_alvo]
y
```

Out[99]:
```
0        8413.2532
1        5645.3432
2        3611.0000
3       15935.1860
4       11326.6080
           ...
1298     4006.6400
1299     9413.7200
1300     1438.1200
1301     4797.9200
1302     2317.3200
Name: Price_euros, Length: 1303, dtype: float64
```

In [100…
```
X = df.drop(var_alvo, axis=1)
X.head(3)
```

Out[100]:

	laptop_ID	Company	Product	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight
0	1	Apple	MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8GB	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37kg
1	2	Apple	Macbook Air	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8GB	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34kg
2	3	HP	250 G6	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8GB	256GB SSD	Intel HD Graphics 620	No OS	1.86kg

In [152…
```
X_train, X_valid, y_train, y_valid = train_test_split(X,
                                                      y,
                                                      test_size=0.1,
                                                      random_state=12)

print(X_train.shape, y_train.shape)
print(X_valid.shape, y_valid.shape)
```

```
(1172, 12) (1172,)
(131, 12) (131,)
```

Tratamento das variáveis de predição

In [153…
```python
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OrdinalEncoder, FunctionTransformer
from sklearn.pipeline import Pipeline

import re
```

In [154…
```python
pd.options.mode.chained_assignment = None
```

In [155…
```python
# Dicts de De x Para, para corrigir itens com mais de uma descrição
cpu_de_para = {
    "Intel Celeron Dual Core N3060 1.60GHz":"Intel Celeron Dual Core N3060 1.6GHz",
    "Intel Core M m7-6Y75 1.2GHz":"Intel Core M M7-6Y75 1.2GHz",
    "Intel Core i3 6006U 2GHz":"Intel Core i3 6006U 2.0GHz",
    "Intel Core i5 7200U 2.50GHz":"Intel Core i5 7200U 2.5GHz",
    "Intel Core i5 7200U 2.70GHz":"Intel Core i5 7200U 2.7GHz",
    "Intel Core i7 6500U 2.50GHz":"Intel Core i7 6500U 2.5GHz"
}

gpu_de_para = {
    "AMD Radeon R2 Graphics":"AMD Radeon R2",
    "AMD Radeon R4 Graphics":"AMD Radeon R4",
    "AMD Radeon R7 Graphics":"AMD Radeon R7",
    "Intel Graphics 620":"Intel HD Graphics 620",
    "Intel HD Graphics 620 ":"Intel HD Graphics 620",
    "Nvidia GeForce 920MX ":"Nvidia GeForce 920MX",
    "Nvidia GeForce 930MX ":"Nvidia GeForce 930MX",
    "Nvidia GeForce GTX 1050 Ti":"Nvidia GeForce GTX 1050Ti",
    "Nvidia GeForce GTX 960<U+039C>":"Nvidia GeForce GTX 960",
    "Nvidia GeForce GTX1050 Ti":"Nvidia GeForce GTX 1050Ti",
    "Nvidia GeForce GTX1060":"Nvidia GeForce GTX 1060",
    "Nvidia GeForce GTX1080":"Nvidia GeForce GTX 1080",
    "Nvidia GeForce MX130":"Nvidia GeForce 130MX",
    "Nvidia GeForce MX150":"Nvidia GeForce 150MX"
}

# Função utilizada para executar o DE x PARA para uma coluna
def de_para(valor, dict):
    valor = valor.strip()
    try:
        # Se encontrar um PARA o retorna
        return dict[valor]
    except:
        # KeyError: Retorna o proprio valor
        return valor
```

```python
# Função para extrair um valor numérico (float) de um texto qualquer
def vlr_texto(vlrStr):
    return float(re.sub('[^0-9|.]','', vlrStr))    # Elimina tudo que não for dígito ou ponto


# Função que converte valores de GB ou TB em numéro e padroniza o retorno em GB
def gb_texto(valor):
    valor = valor.upper().strip()
    if valor.find('TB') >= 0:
        fator = 1024                # Se for TB converte para GB
    else:
        fator = 1                   # Se for GB mantem

    valor = vlr_texto(valor)   # Obtem o valor numérico apenas
    return float(valor) * fator


# Função para tratar resolução de tela especificamente (dois valores a serem multiplicados)
def trata_scr_res(valor):
    px = valor.find('x')              # Encontra o x que divide a resolução (todos os valores possuem o x)
    valor1 = vlr_texto(valor[0:px])   # Obtem o primeiro valor numérico de resolução
    valor2 = vlr_texto(valor[px:])    # Obtem o segundo valor numérico de resolução
    resol  = valor1 * valor2          # Multiplica os valores para obter o total de pixels
    return resol


# Retorna o índice de tipo de HD, sendo: 0=SSD, 1=Hybrid, 2=HDD, 3=Flash Storage
def indice_tipo_hd(x):
    if x.find('SSD') >= 0:
        return 0
    if x.find('HYBRID') >= 0:
        return 1
    if x.find('FLASH') >= 0:
        return 3
    return 2

# Trata o texto de especificação de HDs que pode conter até duas especificações.
# Retorna uma tupla com o tamanho de cada HD, de acordo com seu tipo -- vide indice_tipo_hd()
def trata_hd(x):
    x = x.upper()

    lista_res = [0,0,0,0]    # 0:SSD, 1:Hybrid, 2:HDD, 3:Flash Storage
    # Disco 1
    d1 = x
    i1 = -1
    # Disco 2
    d2 = None
    i2 = -1
    pmais = x.find('+')
    if pmais > 0:
```

```
        # Temos 2 HDs - separa o texto do primeiro do texto do segundo
        d1 = x[0:pmais]
        d2 = x[pmais+1:]

        # Avalia se o HD2 está em TB ou GB para converter em GB
        fator = 1024 if d2.find('TB') >= 0 else 1
        # Determina a posicao do HD na tupla de resultado através de sua tecnologia
        i2 = indice_tipo_hd(d2)
        # Converte em numérico, em GB e armazena na lista
        d2 = vlr_texto(d2) * fator
        lista_res[i2] = d2

    # Avalia se o HD1 está em TB ou GB para converter em GB
    fator = 1024 if d1.find('TB') >= 0 else 1
    # Determina a posicao do HD na tupla de resultado através de sua tecnologia
    i1 = indice_tipo_hd(d1)
    # Converte em numérico, em GB e armazena na lista
    d1 = vlr_texto(d1) * fator
    lista_res[i1] = d1

    return tuple(lista_res)

#
# Transforma as variáveis com tratamento especial do DataFrame
#
def transf_cols(df):
    # Colunas para aplicar OrdinalEncoder
    cols_categ = ['Company','TypeName','Inches','ScreenResolution','Cpu','Ram',
                  'DSK_SSD','DSK_Hybrid','DSK_HDD','DSK_Flash','Gpu','OpSys']
    df = df.copy()
    # Funções que tratam condições especiais
    df['Company'] = df['Company'].apply(lambda x : 1 if x.upper() == 'APPLE' else 0)
    df['ScreenResolution'] = df['ScreenResolution'].apply(lambda x: trata_scr_res(x))
    df['Cpu'] = df['Cpu'].apply(lambda x: de_para(x, cpu_de_para))
    df['Ram'] = df['Ram'].apply(lambda x: gb_texto(x))
    df['Gpu'] = df['Gpu'].apply(lambda x: de_para(x, gpu_de_para))
    df['Weight'] = df['Weight'].apply(lambda x: vlr_texto(x))

    # Nota: 'Memory' será substituída por 4 novas colunas, pois é importante considerar a quantidade e
    #       tecnologia dos HDs do computador como fatores de influencia no preço
    df['DSK_SSD'   ] = df['Memory'].apply(lambda x : trata_hd(x)[0])
    df['DSK_Hybrid'] = df['Memory'].apply(lambda x : trata_hd(x)[1])
    df['DSK_HDD'   ] = df['Memory'].apply(lambda x : trata_hd(x)[2])
    df['DSK_Flash' ] = df['Memory'].apply(lambda x : trata_hd(x)[3])

    # Variáveis indesejadas para o modelo
    df = df.drop(['laptop_ID','Product','Memory'], axis=1)

    # Encoding das colunas categóricas determinadas
    oc = OrdinalEncoder()
    df[cols_categ] = oc.fit_transform(df[cols_categ])
```

```
    return df
```

In [156…]
```
# Transformer a aplicar nas colunas
categoriz = ColumnTransformer(transformers=[('transform', FunctionTransformer(transf_cols), X.columns)])
```

In [157…]
```
# Teste do pipeline
pipe_transf = Pipeline(steps=[('categ', categoriz)])
pipe_transf
```

Out[157]:
```
Pipeline(steps=[('categ',
                 ColumnTransformer(transformers=[('transform',
                                                  FunctionTransformer(func=<function transf_cols at 0x000001DCC95CE5E0>),
                                                  Index(['laptop_ID', 'Company', 'Product', 'TypeName', 'Inches',
       'ScreenResolution', 'Cpu', 'Ram', 'Memory', 'Gpu', 'OpSys', 'Weight'],
      dtype='object'))]))])
```

In [158…]
```
# Conferindo o DataFrame resultante do Transformer
X_pipe_train = pd.DataFrame(pipe_transf.fit_transform(X_train))
X_pipe_train.sample(15)
```

Out[158]:

	0	1	2	3	4	5	6	7	8	9	10	11	12
237	0.0	3.0	16.0	3.0	73.0	3.0	61.0	5.0	2.50	5.0	0.0	3.0	0.0
230	0.0	3.0	16.0	3.0	84.0	3.0	76.0	4.0	2.69	5.0	0.0	3.0	0.0
241	0.0	3.0	14.0	0.0	68.0	3.0	47.0	5.0	1.90	0.0	0.0	2.0	0.0
98	0.0	3.0	16.0	3.0	65.0	3.0	61.0	5.0	2.63	8.0	0.0	0.0	0.0
585	0.0	3.0	14.0	0.0	99.0	1.0	37.0	5.0	2.00	0.0	0.0	2.0	0.0
120	0.0	4.0	7.0	3.0	41.0	3.0	40.0	5.0	1.20	9.0	0.0	0.0	0.0
1085	0.0	3.0	14.0	3.0	73.0	3.0	56.0	4.0	1.95	8.0	0.0	0.0	0.0
826	0.0	2.0	2.0	0.0	33.0	0.0	35.0	5.0	1.17	0.0	0.0	0.0	2.0
806	0.0	3.0	14.0	3.0	65.0	3.0	47.0	2.0	1.98	0.0	0.0	3.0	0.0
62	0.0	1.0	14.0	3.0	67.0	3.0	66.0	5.0	2.40	8.0	0.0	0.0	0.0
498	0.0	2.0	5.0	3.0	77.0	5.0	41.0	5.0	1.30	9.0	0.0	0.0	0.0
1133	0.0	4.0	10.0	3.0	65.0	3.0	47.0	5.0	1.13	0.0	0.0	0.0	5.0
567	0.0	3.0	16.0	3.0	93.0	3.0	6.0	5.0	2.80	5.0	0.0	3.0	0.0
545	0.0	1.0	16.0	3.0	89.0	3.0	66.0	5.0	3.00	0.0	0.0	3.0	0.0
323	0.0	3.0	14.0	3.0	65.0	1.0	57.0	2.0	2.10	0.0	0.0	3.0	0.0

In [159…]
```
def ren_cols(df):
```

```
    # 0       1        2     3               4   5   6   7     8      9       10         11      12
    # Company TypeName Inches ScreenResolution Cpu Ram Gpu OpSys Weight DSK_SSD DSK_Hybrid DSK_HDD DSK_Flash
    df.rename({0:'Company',1:'TypeName',2:'Inches',3:'Screen',4:'Cpu',5:'Ram',6:'Gpu',7:'SO',8:'Peso',9:'Dsk.SSD',
                    10:'Dsk.Hybrid',11:'Dsk.HDD',12:'Dsk.Flash'},axis=1,inplace=True)
    return df

X_pipe_train = ren_cols(X_pipe_train)
X_pipe_train.head(3)
```

Out[159]:

	Company	TypeName	Inches	Screen	Cpu	Ram	Gpu	SO	Peso	Dsk.SSD	Dsk.Hybrid	Dsk.HDD	Dsk.Flash
0	0.0	3.0	14.0	3.0	11.0	3.0	13.0	5.0	2.23	8.0	0.0	0.0	0.0
1	0.0	3.0	14.0	3.0	65.0	3.0	47.0	4.0	2.30	8.0	0.0	0.0	0.0
2	0.0	3.0	4.0	11.0	38.0	2.0	37.0	5.0	1.40	0.0	0.0	0.0	3.0

In [160…

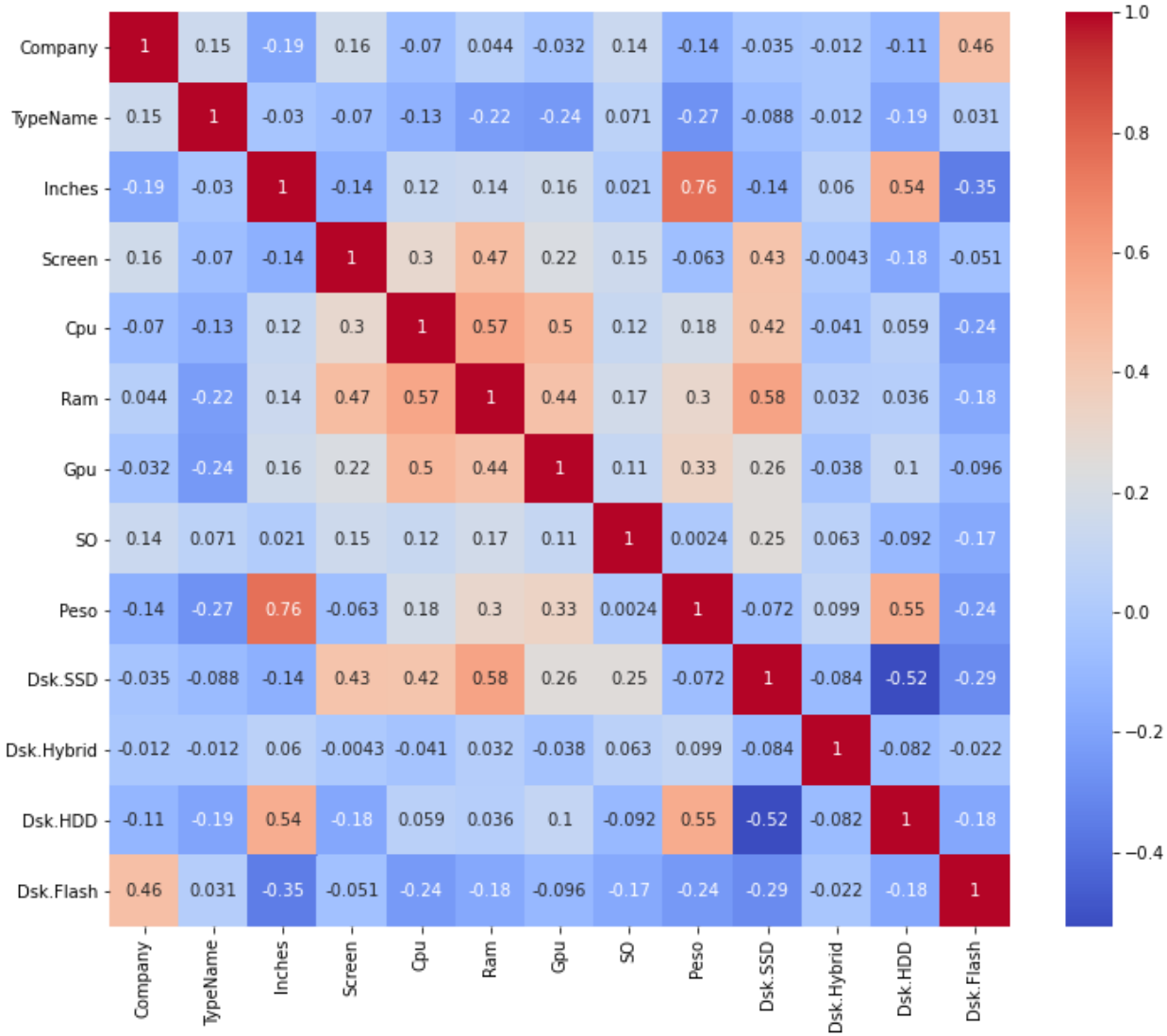
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [161…

```
# Análise da correlação das variáveis
plt.figure(figsize=(12,10))
sns.heatmap(X_pipe_train.corr(), cmap='coolwarm', annot=True)
```

Out[161]:

```
<AxesSubplot:>
```



In [162…
```
import numpy as np
import statsmodels.api as sm
from lightgbm import LGBMRegressor
```

In [163…
```
# Testando os parâmetros do modelo de regressão para a melhor combinação
from sklearn.model_selection import GridSearchCV
```

In [164…
```
modelo = LGBMRegressor()
parameters = {
    'boosting_type': ['gb','dart'],
    'max_depth': [3, 5, 7, 9],
```

```
    'n_estimators': [50, 100, 250],
    'learning_rate': [0.01, 0.05, 0.1, 0.5, 1],
    'min_samples_leaf': [3, 5, 10, 15, 20],
    'is_unbalance': [True, False]
}
clf = GridSearchCV(modelo, parameters, n_jobs=-1, verbose=0, scoring='neg_mean_squared_error')
clf.fit(X_pipe_train, y_train)
```

```
[LightGBM] [Warning] min_data_in_leaf is set with min_child_samples=20, will be overridden by min_samples_leaf=3. Current value: min_data_in_leaf=3
D:\LetsCode\anaconda3\lib\site-packages\sklearn\model_selection\_search.py:918: UserWarning: One or more of the test scores are non-finite: [           nan
nan               nan ...
 -3545595.54495033 -3384693.35583088 -3316618.79721882]
  warnings.warn(
```

Out[164]:
```
GridSearchCV(estimator=LGBMRegressor(), n_jobs=-1,
             param_grid={'boosting_type': ['gb', 'dart'],
                         'is_unbalance': [True, False],
                         'learning_rate': [0.01, 0.05, 0.1, 0.5, 1],
                         'max_depth': [3, 5, 7, 9],
                         'min_samples_leaf': [3, 5, 10, 15, 20],
                         'n_estimators': [50, 100, 250]},
             scoring='neg_mean_squared_error')
```

In [165…
```
# Melhores parâmetros
clf.best_params_
```

Out[165]:
```
{'boosting_type': 'dart',
 'is_unbalance': True,
 'learning_rate': 0.5,
 'max_depth': 3,
 'min_samples_leaf': 3,
 'n_estimators': 250}
```

In [146…
```
modelo = LGBMRegressor(boosting_type='dart',is_unbalance=True,learning_rate=0.5,max_depth=5,
                       n_estimators=250, min_samples_leaf=5)
modelo.fit(X_pipe_train, y_train)
```

```
[LightGBM] [Warning] min_data_in_leaf is set with min_child_samples=20, will be overridden by min_samples_leaf=5. Current value: min_data_in_leaf=5
```

Out[146]:
```
LGBMRegressor(boosting_type='dart', is_unbalance=True, learning_rate=0.5,
              max_depth=5, min_samples_leaf=5, n_estimators=250)
```

In [147…
```
importances = pd.DataFrame({
    'feature': X_pipe_train.columns,
    'importance': modelo.feature_importances_}).sort_values('importance', ascending=False)
importances.head(7)
```

Out[147]:

	feature	importance
8	Peso	1388
6	Gpu	861
4	Cpu	834
9	Dsk.SSD	374
5	Ram	365
3	Screen	319
2	Inches	282

In [148…
```
# Predições de treino e validação
X_pipe_valid = ren_cols(pd.DataFrame(pipe_transf.fit_transform(X_valid)))
```

In [149…
```
print('X_pipe_train:')
display(X_pipe_train.head(3))
print('X_pipe_valid:')
display(X_pipe_valid.head(3))
```

X_pipe_train:

	Company	TypeName	Inches	Screen	Cpu	Ram	Gpu	SO	Peso	Dsk.SSD	Dsk.Hybrid	Dsk.HDD	Dsk.Flash
0	0.0	5.0	14.0	3.0	83.0	3.0	83.0	7.0	2.59	8.0	0.0	0.0	0.0
1	0.0	3.0	14.0	3.0	0.0	2.0	14.0	5.0	2.40	0.0	0.0	4.0	0.0
2	0.0	3.0	14.0	3.0	63.0	3.0	16.0	5.0	2.50	0.0	2.0	0.0	0.0

X_pipe_valid:

	Company	TypeName	Inches	Screen	Cpu	Ram	Gpu	SO	Peso	Dsk.SSD	Dsk.Hybrid	Dsk.HDD	Dsk.Flash
0	0.0	4.0	7.0	6.0	36.0	5.0	21.0	4.0	1.10	5.0	0.0	0.0	0.0
1	0.0	3.0	8.0	3.0	40.0	3.0	30.0	4.0	2.20	0.0	0.0	2.0	0.0
2	0.0	3.0	8.0	0.0	29.0	3.0	25.0	4.0	2.36	0.0	0.0	3.0	0.0

In [150…
```
y_train_pred = modelo.predict(X_pipe_train)
y_valid_pred = modelo.predict(X_pipe_valid)
```

In [151…
```
# Métricas
from sklearn import metrics
import math

rmse_train = metrics.mean_squared_error(y_train, y_train_pred, squared=False)
r2_train   = metrics.r2_score(          y_train, y_train_pred)
```

```
rmse_valid = metrics.mean_squared_error(y_valid, y_valid_pred, squared=False)
r2_valid   = metrics.r2_score(          y_valid, y_valid_pred)

print(f'Treino...: RMSE: {rmse_train:.2f} R2: {r2_train:.2f}%')
print(f'Validação: RMSE: {rmse_valid:.2f} R2: {r2_valid:.2f}%')
```

```
Treino...: RMSE: 601.79 R2: 0.98%
Validação: RMSE: 4106.71 R2: -0.05%
```

In [131…
```
# Pipeline Final
pipe = Pipeline(steps = [
    ('categ', categoriz),
    ('modelo', LGBMRegressor(boosting_type='dart',is_unbalance=True,learning_rate=0.5,max_depth=3,min_samples_leaf=5,
                             n_estimators=250))
])
```

Esperamos erro residual 0 (zero)

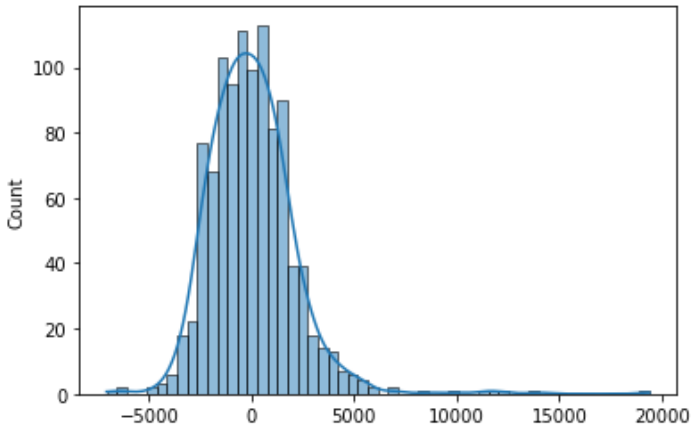
In [132…
```
model.resid.mean()
```

Out[132]:
```
-6.873580399675442e-13
```

Esperamos que o erro residual siga uma distribuição normal

In [133…
```
sns.histplot(model.resid, kde=True)
```

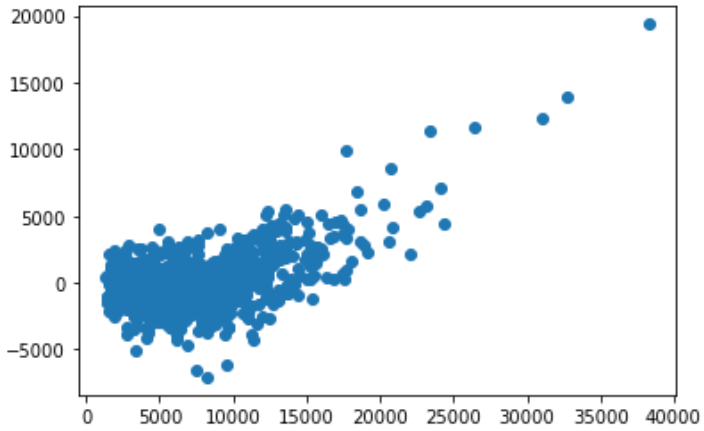
Out[133]:
```
<AxesSubplot:ylabel='Count'>
```



O Erro residual deve ser descorrelacionado

In [134…
```
plt.scatter(y_train_pred, model.resid)
```

Out[134]:
```
<matplotlib.collections.PathCollection at 0x1dcdad61460>
```



In [135…
```
# Realizando o Fit da pipeline com os dados de treino
pipe.fit(X_train, y_train)
```

Out[135]:
```
[LightGBM] [Warning] min_data_in_leaf is set with min_child_samples=20, will be overridden by min_samples_leaf=5. Current value: min_data_in_leaf=5
Pipeline(steps=[('categ',
                 ColumnTransformer(transformers=[('transform',
                                                  FunctionTransformer(func=<function transf_cols at 0x000001DCC95CE550>),
                                                  Index(['laptop_ID', 'Company', 'Product', 'TypeName', 'Inches',
       'ScreenResolution', 'Cpu', 'Ram', 'Memory', 'Gpu', 'OpSys', 'Weight'],
      dtype='object'))])),
                ('modelo',
                 LGBMRegressor(boosting_type='dart', is_unbalance=True,
                               learning_rate=0.5, max_depth=3,
                               min_samples_leaf=5, n_estimators=250))])
```

In [136…
```
# Predições de treino e validação
y_train_pred = pipe.predict(X_train)
y_valid_pred = pipe.predict(X_valid)
```

In [137…
```
# Métricas
from sklearn import metrics
import math

mse_train  = metrics.mean_squared_error(y_train, y_train_pred)
rmse_train = metrics.mean_squared_error(y_train, y_train_pred, squared=False)
r2_train   = metrics.r2_score(          y_train, y_train_pred)

mse_valid  = metrics.mean_squared_error(y_valid, y_valid_pred)
rmse_valid = metrics.mean_squared_error(y_valid, y_valid_pred, squared=False)
r2_valid   = metrics.r2_score(          y_valid, y_valid_pred)

print(f'Treino...: MSE: {mse_train:11.2f} RMSE: {rmse_train:.2f} R2: {r2_train:.2f}%')
print(f'Validação: MSE: {mse_valid:11.2f} RMSE: {rmse_valid:.2f} R2: {r2_valid:.2f}%')
```

```
Treino...: MSE:   924450.17 RMSE: 961.48 R2: 0.95%
Validação: MSE: 16357894.41 RMSE: 4044.49 R2: -0.02%
```

In [138…
```
importances = pd.DataFrame({
    'feature': X_pipe_train.columns,
    'importance': modelo.feature_importances_}).sort_values('importance', ascending=False)
importances.head(7)
```

Out[138]:

	feature	importance
8	Peso	374
6	Gpu	314
4	Cpu	237
5	Ram	159
9	Dsk.SSD	142
3	Screen	115
2	Inches	79

In [139…
```
import pickle
```

In [140…
```
with open('modelo_computadores.pkl', 'wb') as file:
    pickle.dump(pipe, file)
```

Testando o modelo

Carga do modelo

In [141…
```
pipe_teste = pickle.load(open('modelo_computadores.pkl', 'rb'))
```

Input das variáveis

In [142…
```
Company = 'HP'
# Apple ou Qualquer outra marca

TypeName = 'Notebook'
# Valores válidos: 'Ultrabook', 'Notebook', 'Netbook', 'Gaming', '2 in 1 Convertible','Workstation'

Inches = 15.6
# Valor da resolução da tela, com .
# Valores usuais: 10.1,11.3,11.6,12.0,12.3,12.5,13.0,13.3,13.5,13.9,14.0,14.1,15.0,15.4,15.6,17.0,17.3,18.4
```

```
Screen = 'Full HD 1920x1080'
# Descrição da tela. Valores usuais:
#       '1366x768', '1440x900', '1600x900', '1920x1080', '2560x1440',
#       '4K Ultra HD / Touchscreen 3840x2160',
#       '4K Ultra HD 3840x2160',
#       'Full HD / Touchscreen 1920x1080', 'Full HD 1920x1080',
#       'IPS Panel 1366x768',
#       'IPS Panel 2560x1440',
Cpu = 'Intel Core i5 7200U 2.5GHz'
# Descrição da CPU. Valores usuais:
#       'AMD A10-Series 9600P 2.4GHz',
#       'AMD A10-Series 9620P 2.5GHz',
#       'AMD A10-Series A10-9620P 2.5GHz'
#       'Intel Celeron Dual Core 3205U 1.5GHz',
#       'Intel Core i3 6006U 2.0GHz',
#       'Intel Core i3 6006U 2.2GHz',
#       'Intel Core i5 1.6GHz',
#       'Intel Core i5 1.8GHz',
#       'Intel Core i7 7660U 2.5GHz'
Ram = '16GB'
# Valores válidos: '12GB', '16GB', '24GB', '2GB', '32GB', '4GB', '64GB', '6GB', '8GB'
Disks = '256GB SSD'
# Disco(s). Valores usuais:
#       '1.0TB HDD', '1.0TB Hybrid', '128GB Flash Storage', '128GB HDD', '256GB SSD', '16GB Flash Storage'
#       '128GB SSD', '128GB SSD +  1TB HDD', '128GB SSD +  2TB HDD', '180GB SSD', '1TB HDD'
Gpu = 'Intel HD Graphics 620'
# Placa gráfica. Valores usuais:
#       'AMD FirePro W4190M', 'AMD FirePro W4190M ', 'AMD FirePro W5130M',
#       'AMD FirePro W6150M', 'AMD R17M-M1-70', 'AMD R4 Graphics',
#       'AMD Radeon 520', 'AMD Radeon 530', 'AMD Radeon 540',
#       'Intel Graphics 620', 'Intel HD Graphics', 'Intel HD Graphics 400',
#       'Intel HD Graphics 405', 'Intel HD Graphics 500',
#       'Nvidia GTX 980 SLI', 'Nvidia GeForce 150MX', 'Nvidia GeForce 920',
#       'Nvidia GeForce GTX 1050', 'Nvidia GeForce GTX 1050 Ti',
#       'Nvidia GeForce GTX 1060', 'Nvidia GeForce GTX 1070',
#       'Nvidia GeForce GTX 960M', 'Nvidia GeForce GTX 965M',
#       'Nvidia Quadro 3000M', 'Nvidia Quadro M1000M',
OpSys = 'No OS'
# 'Android','Chrome OS','Linux','Mac OS X','No OS','Windows 10','Windows 10 S','Windows 7','macOS'
Weight = '1.86kg'
# Peso entre 0.50kg e 5kg
```

Predição do preço

In [143…

```
df_tst = pd.DataFrame([{'laptop_ID':0,
                        'Company':Company,
                        'Product':0,
                        'TypeName':TypeName,
                        'Inches':Inches,
                        'ScreenResolution':Screen,
```

```
                    'Cpu':Cpu,
                    'Ram':Ram,
                    'Memory':Disks,
                    'Gpu':Gpu,
                    'OpSys':OpSys,
                    'Weight':Weight}])

preco = pipe_teste.predict(df_tst)[0]
print('Preço estimado do computador:', preco)
```

```
Preço estimado do computador: 3837.2788649784134
```

In [144…

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

Out[144]:

	laptop_ID	Company	Product	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price_euros
0	1	Apple	MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8GB	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37kg	8413.2532
1	2	Apple	Macbook Air	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8GB	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34kg	5645.3432
2	3	HP	250 G6	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8GB	256GB SSD	Intel HD Graphics 620	No OS	1.86kg	3611.0000

Comentários sobre a estratégia

"...deseja explorar combinações de tamanho e resolução da tela, versão do CPU e o peso do equipamento"

In [145…

```
importances = pd.DataFrame({
    'feature': X_pipe_train.columns,
    'importance': modelo.feature_importances_}).sort_values('importance', ascending=False)
importances.head(7)
```

Out[145]:

	feature	importance
8	Peso	374
6	Gpu	314
4	Cpu	237
5	Ram	159
9	Dsk.SSD	142
3	Screen	115
2	Inches	79

Os investidores não estão errados nas variáveis selecionadas. Inclusive "Peso" é a variável mais relevante, contudo devem ser consideradas também as variáveis **Gpu**, "Ram" e **Memory**, pois têm alta significância no preço.
Sobre **Memory**, que na verdade é o conjunto de discos do computador, a presença ou não de SSD é relevante para o preço.

In [ ]: