laptop-price-prediction

September 16, 2024

1 Importing Libraries

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
# Drive mounting

from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[3]:
```
# Reading dataset and checking top 5 and last 5 records

df = pd.read_csv('/content/drive/MyDrive/Colab/laptop_price.csv',␣
 ↪encoding='latin-1')
df.head()
```

[3]:

	laptop_ID	Company	Product	TypeName	Inches	ScreenResolution	Cpu	Ram
0	1	Apple	MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8GB
1	2	Apple	Macbook Air	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8GB
2	3	HP	250 G6	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8GB
3	4	Apple	MacBook Pro	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16GB
4	5	Apple	MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8GB

```
                Memory                           Gpu  OpSys  Weight  \
0            128GB SSD  Intel Iris Plus Graphics 640  macOS  1.37kg
1  128GB Flash Storage        Intel HD Graphics 6000  macOS  1.34kg
2            256GB SSD         Intel HD Graphics 620  No OS  1.86kg
3            512GB SSD            AMD Radeon Pro 455  macOS  1.83kg
4            256GB SSD  Intel Iris Plus Graphics 650  macOS  1.37kg

   Price_euros
0      1339.69
1       898.94
2       575.00
3      2537.45
4      1803.60
```

[4]:
```
# checking shape

print('DataFrame Shape : ', df.shape)
```

```
DataFrame Shape :  (1303, 13)
```

[5]:
```
# Checking columns

print('DataFrame Column: ',df.columns)
print('\n')
print('No. of  features: ',len(df.columns))
```

```
DataFrame Column:  Index(['laptop_ID', 'Company', 'Product', 'TypeName',
'Inches',
       'ScreenResolution', 'Cpu', 'Ram', 'Memory', 'Gpu', 'OpSys', 'Weight',
       'Price_euros'],
      dtype='object')


No. of  features:  13
```

- **Dataset has 1303 samples data and 13 features. There are 12 independent features and 1 dependent feature. And Target feature is 'Price_euros.'**

[6]:
```
# Removing column 'laptop_ID' as it doesn't affect much the price and unique␣
 ↪for each laptop

df.drop('laptop_ID', axis = 1, inplace = True)

df.head()
```

[6]:
```
  Company      Product   TypeName  Inches                    ScreenResolution  \
0   Apple  MacBook Pro  Ultrabook    13.3  IPS Panel Retina Display 2560x1600
```

```
1   Apple  Macbook Air  Ultrabook    13.3                          1440x900
2      HP       250 G6   Notebook    15.6                   Full HD 1920x1080
3   Apple  MacBook Pro  Ultrabook    15.4  IPS Panel Retina Display 2880x1800
4   Apple  MacBook Pro  Ultrabook    13.3  IPS Panel Retina Display 2560x1600

                           Cpu   Ram               Memory  \
0        Intel Core i5 2.3GHz   8GB            128GB SSD
1        Intel Core i5 1.8GHz   8GB  128GB Flash Storage
2  Intel Core i5 7200U 2.5GHz   8GB            256GB SSD
3        Intel Core i7 2.7GHz  16GB            512GB SSD
4        Intel Core i5 3.1GHz   8GB            256GB SSD

                             Gpu  OpSys  Weight  Price_euros
0  Intel Iris Plus Graphics 640  macOS  1.37kg      1339.69
1        Intel HD Graphics 6000  macOS  1.34kg       898.94
2         Intel HD Graphics 620  No OS  1.86kg       575.00
3            AMD Radeon Pro 455  macOS  1.83kg      2537.45
4  Intel Iris Plus Graphics 650  macOS  1.37kg      1803.60
```

[7]:
```
# Basic information about dataset

print('Laptop_Price_Prediction_Dataset: \n')
df.info()
```

```
Laptop_Price_Prediction_Dataset:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1303 entries, 0 to 1302
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Company           1303 non-null   object
 1   Product           1303 non-null   object
 2   TypeName          1303 non-null   object
 3   Inches            1303 non-null   float64
 4   ScreenResolution  1303 non-null   object
 5   Cpu               1303 non-null   object
 6   Ram               1303 non-null   object
 7   Memory            1303 non-null   object
 8   Gpu               1303 non-null   object
 9   OpSys             1303 non-null   object
 10  Weight            1303 non-null   object
 11  Price_euros       1303 non-null   float64
dtypes: float64(2), object(10)
memory usage: 122.3+ KB
```

[8]:
```
# Check missing values

print('Column-wise missing values count: \n')
df.isnull().sum()
```

```
Column-wise missing values count:
```

[8]:
```
Company             0
Product             0
TypeName            0
Inches              0
ScreenResolution    0
Cpu                 0
Ram                 0
Memory              0
Gpu                 0
OpSys               0
Weight              0
Price_euros         0
dtype: int64
```

- There is not any missing values in the dataset.

[9]:
```
# Check any duplicated rows

print('No of duplicated records :',df.duplicated().sum())
```

```
No of duplicated records : 28
```

[10]:
```
# Getting categorical and numerical features

cat_features = df.select_dtypes(include = 'object').columns
# alternate way : [feature for feature in df.columns if df[feature].dtypes ==␣
 ↪'object']
num_features = df.select_dtypes(include =␣
 ↪['int32','int64','float32','float64']).columns
# alternate way : [feature for feature in df.columns if df[feature].dtypes !=␣
 ↪'object']
```

[11]:
```
print('Categorical Features list:\n')
print(cat_features)
print('-' * 100)
print('Numerical Features list:\n')
print(num_features)
```

```
Categorical Features list:
```

```
Index(['Company', 'Product', 'TypeName', 'ScreenResolution', 'Cpu', 'Ram',
       'Memory', 'Gpu', 'OpSys', 'Weight'],
      dtype='object')
--------------------------------------------------------------------------------
--------------------
Numerical Features list:

Index(['Inches', 'Price_euros'], dtype='object')
```

[12]:
```
# Function to get unique values present in cols

def uniqueValues(feature):

    print(f'Unique values in {feature} is : {df[feature].unique()}')

for feature in cat_features:
    uniqueValues(feature)
    print('\n')
    print('-'*100)
```

```
Unique values in Company is : ['Apple' 'HP' 'Acer' 'Asus' 'Dell' 'Lenovo'
'Chuwi' 'MSI' 'Microsoft'
 'Toshiba' 'Huawei' 'Xiaomi' 'Vero' 'Razer' 'Mediacom' 'Samsung' 'Google'
 'Fujitsu' 'LG']


--------------------------------------------------------------------------------
--------------------
Unique values in Product is : ['MacBook Pro' 'Macbook Air' '250 G6' 'Aspire 3'
'ZenBook UX430UN'
 'Swift 3' 'Inspiron 3567' 'MacBook 12"' 'IdeaPad 320-15IKB' 'XPS 13'
 'Vivobook E200HA' 'Legion Y520-15IKBN' '255 G6' 'Inspiron 5379'
 '15-BS101nv (i7-8550U/8GB/256GB/FHD/W10)' 'MacBook Air' 'Inspiron 5570'
 'Latitude 5590' 'ProBook 470' 'LapBook 15.6"'
 'E402WA-GA010T (E2-6110/2GB/32GB/W10)'
 '17-ak001nv (A6-9220/4GB/500GB/Radeon' 'IdeaPad 120S-14IAP'
 'Inspiron 5770' 'ProBook 450' 'X540UA-DM186 (i3-6006U/4GB/1TB/FHD/Linux)'
 'Inspiron 7577' 'X542UQ-GO005 (i5-7200U/8GB/1TB/GeForce'
 'Aspire A515-51G' 'Inspiron 7773' 'IdeaPad 320-15ISK' 'Rog Strix'
 'X751NV-TY001T (N4200/4GB/1TB/GeForce' 'Yoga Book' 'ProBook 430'
 'Inspiron 3576' '15-bs002nv (i3-6006U/4GB/128GB/FHD/W10)' 'VivoBook Max'
 'GS73VR 7RG' 'X541UA-DM1897 (i3-6006U/4GB/256GB/FHD/Linux)' 'Vostro 5471'
 'IdeaPad 520S-14IKB' 'UX410UA-GV350T (i5-8250U/8GB/256GB/FHD/W10)'
 'ZenBook Pro' 'Stream 14-AX040wm' 'V310-15ISK (i5-7200U/4GB/1TB/FHD/W10)'
 'FX753VE-GC093 (i7-7700HQ/12GB/1TB/GeForce' 'Surface Laptop'
 'Inspiron 5370' 'GL72M 7RDX' 'Aspire E5-475'
 'FX503VD-E4022T (i7-7700HQ/8GB/1TB/GeForce' 'IdeaPad 320-15IKBN'
 'Aspire A515-51G-32MX' 'ProBook 440' 'IdeaPad 320-15AST'
```

'Pavilion 15-CK000nv' 'FX503VM-E4007T (i7-7700HQ/16GB/1TB'
'FX550IK-DM018T (FX-9830P/8GB/1TB/Radeon' 'Aspire 5' 'Probook 430'
'Zenbook UX430UA' 'Spin 5' 'X541UV-DM1439T (i3-7100U/6GB/256GB/GeForce'
'Omen 15-ce007nv' '15-bs017nv (i7-7500U/8GB/256GB/Radeon'
'15-bw000nv (E2-9000e/4GB/500GB/Radeon' 'Envy 13-ad009n'
'Pavilion 14-BK001nv' 'Ideapad 310-15ISK'
'UX430UQ-GV209R (i7-7500U/8GB/256GB/GeForce' 'GP62M 7REX' 'Thinkpad T470'
'VivoBook S15' 'ThinkPad Yoga' 'Probook 440' 'Spectre x360'
'Inspiron 7570' 'X705UV-BX074T (i3-6006U/4GB/1TB/GeForce' 'Spin 3'
'GS63VR 7RG' 'Probook 470' 'E402WA-GA007T (E2-6110/4GB/64GB/W10'
'Inspiron 5567' 'Aspire A515-51G-37JS'
'15-BS078nr (i7-7500U/8GB/1TB/W10)' 'V110-15IAP (N3350/4GB/1TB/No'
'FX753VD-GC086T (i5-7300HQ/8GB/1TB' 'Envy 13-AD007nv' 'ThinkPad E480'
'Satellite Pro' 'ZenBook UX430UA' 'EliteBook Folio'
'X541NA (N3350/4GB/1TB/FHD/W10)' 'GE72MVR 7RG' 'Aspire A315-51'
'Inspiron 5577' 'Inspiron 7567' 'V110-15IKB (i5-7200U/4GB/128GB/W10)'
'GE73VR 7RE' 'EliteBook 840' '15-BS103nv (i5-8250U/6GB/256GB/Radeon'
'Yoga 520-14IKB' 'ZenBook Flip' 'Inspiron 5579'
'X555BP-XX180T (A9-9420/4GB/1TB/Radeon' 'Aspire A517-51G'
'Aspire A315-31' 'GE63VR 7RE' 'MateBook X'
'17-bs001nv (i5-7200U/6GB/2TB/Radeon' 'GT80S 6QF-074US'
'V310-15IKB (i5-7200U/8GB/1TB' 'Yoga 920-13IKB' 'Mi Notebook' 'XPS 15'
'Swift 7' 'Thinkpad Yoga' 'K147 (N3350/4GB/32GB/FHD/W10)'
'IdeaPad 320-17IKBR' 'Blade Pro' 'Omen 17-W295'
'V110-15ISK (i5-6200U/4GB/128GB/W10)' 'Aspire E5-576G'
'Legion Y720-15IKB' 'Precision 7520' 'Aspire 7' 'ROG GL703VD-GC028T'
'15-bs018nq (i3-6006U/4GB/500GB/FHD/No' 'IdeaPad 320-17IKB'
'Latitude 5490' 'Portege Z30-C-16L' 'Alienware 17'
'Vivobook X541UV-DM1217T' 'K756UX-T4340T (i5-7200U/8GB/500GB' 'ZBook 15u'
'Pro P2540UA-XO0198T' '15-rb013nv (E2-9000e/4GB/500GB/W10)' 'Vostro 5468'
'Aspire R7' 'X555QG-DM242T (A10-9620P/4GB/1TB' 'ROG G703VI-E5062T'
'Nitro AN515-51' 'VivoBook Pro' 'F756UX-T4201D (i7-7500U/8GB/128GB'
'Yoga 910-13IKB' '15-bs015dx (i5-7200U/8GB/1TB/W10)' 'Rog G701VIK-BA060T'
'ROG G752VSK-GC493T' 'X505BP-BR019T (A9-9420/4GB/1TB/Radeon'
'Vostro 5370' '15-BW094nd (A6-9220/8GB/128GB/W10)' 'Envy 17-U275cl'
'GT73EVR 7RE' 'Yoga 720-15IKB' 'Vostro 3568'
'V330-15IKB (i7-8550U/8GB/256GB/FHD/W10)' 'ThinkPad X1'
'IdeaPad 320-17ISK' 'Ideapad 320-15IKBN'
'SP315-51 (i7-7500U/12GB/1TB/FHD/W10)' 'Thinkpad T570'
'Chromebook C910-C2ST' 'FX753VD-GC071T (i7-7700HQ/8GB/1TB/GeForce'
'17-BS037cl (i3-6006U/8GB/1TB/W10)'
'V330-15IKB (i5-8250U/8GB/256GB/FHD/W10)' 'Aspire A715-71G'
'Precision 7720' 'IdeaPad 310-15ABR' 'ZenBook UX530UQ-PRO' 'VivoBook S14'
'Rog GL702VS-GC095T' 'GL553VE-FY082T (i7-7700HQ/8GB/1TB'
'IdeaPad 320-15IAP' 'EliteBook x360' 'IdeaPad 720S-13IKB' 'GE63VR 7RF'
'ES1-523-84K7 (A8-7410/8GB/256GB/FHD/W10)' 'VivoBook Flip' 'ThinkPad 13'
'ProBook 640' 'TravelMate B' 'Elitebook 840' 'ZenBook UX410UA-GV183T'
'Aspire E5-575' 'Elitebook 820' 'GL72M 7REX'

'UX510UX-CN269T (i7-7500U/8GB/256GB'
'V310-15ISK (i3-6006U/4GB/1TB/FHD/W10)'
'FX553VD-FY647T (i7-7700HQ/8GB/256GB/GeForce' 'Elitebook 850'
'X541NA (N3350/4GB/1TB/Linux)' 'Inspiron 3552' 'IdeaPad 320-15ABR'
'Stream 14-AX001nv' 'GP72MVR 7RFX' 'Zbook 15' 'Tecra A50-C-21G'
'Latitude 7480' 'Zenbook UX410UA-GV027T' '15-AY023na (N3710/8GB/2TB/W10)'
'Elitebook 1040' 'IdeaPad 110-17ACL'
'15-bw003nv (A9-Series-9420/4GB/256GB/FHD/W10)' 'Yoga 11e'
'VivoBook E403NA' 'Omen 17-w212nv'
'V310-15ISK (i3-6006U/4GB/128GB/FHD/No' 'ROG Strix' 'IdeaPad 720S-14IKB'
'Zenbook Flip' 'Thinkpad X1' 'Ideapad 510S-13IKB' 'Precision 3510'
'Precision 5520' 'Rog GL753VD-GC042T' 'Rog GL753VE-GC070T'
'Leopard GP72M' '15-BW004nv (A9-9420/4GB/256GB/Radeon' 'ThinkPad E580'
'ThinkPad L470' 'Precision M5520' 'FX753VD-GC461T (i7-7700HQ/16GB/1TB'
'GE73VR 7RF' 'Zenbook 3' 'Portege Z30-C-16P' 'Lenovo IdeaPad'
'ThinkPad P51' 'Thinkpad T470p' '15-BS028nv (i3-6006U/4GB/1TB/Radeon'
'Latitude 3380' 'EliteBook 1040' 'LapBook 12.3' 'ProBook 650'
'X542UQ-DM117 (i3-7100U/8GB/1TB/GeForce' 'Latitude 5480' 'Omen 17-w207nv'
'FlexBook Edge' 'Chromebook 3' 'Thinkpad 13' 'IdeaPad 320s-14IKB'
'Thinkpad P51' '15-ra044nv (N3060/4GB/500GB/W10)' 'Pixelbook (Core'
'ThinkPad T470s' 'ThinkPad X270' 'Omen 15-AX205na' 'Aspire ES1-572'
'Precision 3520' 'GV62 7RD-1686NL' '15-bs024nv (i5-7200U/8GB/128GB/W10)'
'ThinkPad T470' 'Inspiron 3168' '17-BS092ND (i3-6006U/8GB/256GB/W10)'
'Pro P2540UA-AB51' 'IdeaPad 510s-14IKB'
'X541NA-PD1003Y (N4200/4GB/500GB/W10)' 'Omen 17-an006nv' 'Thinkpad T460s'
'Latitude 7390' 'Latitude E5470' 'Portege X30-D-10J' 'Lapbook 15,6'
'ThinkPad E570' 'Thinkpad X270' 'Zenbook UX390UA' 'Thinkpad E570'
'Portege X30-D-10L' 'Rog G752VL-UH71T' 'Thinkpad X260'
'Ideapad 520-15IKBR' 'ThinkPad L570' 'VivoBook E201NA'
'15-BS026nv (i5-7200U/8GB/256GB/Radeon' 'IdeaPad 320-14IAP'
'Chromebook N23' 'ZenBook UX510UX-CN211T' 'Aspire A515-51G-59QF'
'Envy 13-AB002nv' 'Vostro 5568' 'VivoBook E12'
'15-bs190od (i5-8250U/4GB/1TB/W10)' 'ROG Zephyrus' 'Probook 450'
'FX753VE-GC155T (i7-7700HQ/16GB/1TB' 'Spectre X360' 'Latitude 5580'
'Zenbook UX510UW-FI095T' 'SmartBook Edge' 'Omen 15-ce006nv'
'Thinkpad E470' 'Envy 13-AB020nr' 'VivoBook X540YA-XX519T'
'ThinkPad E470' 'V310-15ISK (i5-6200U/4GB/1TB/FHD/No' 'ThinkPad T570'
'17-X047na (i3-6006U/8GB/1TB/W10)' 'A541NA-GO342 (N3350/4GB/500GB/Linux)'
'SmartBook 130' '15-bw007nv (A10-9620P/6GB/128GB/Radeon' 'Spin SP111-31'
'V330-15IKB (i3-7130U/4GB/128GB/FHD/W10)' 'EliteBook 1030' 'Thinkpad P71'
'FX553VD-DM627T (i5-7300HQ/8GB/1TB' 'Lifebook A557' 'ZBook 17'
'14-am079na (N3710/8GB/2TB/W10)' '15-cd005nv (A9-9420/6GB/256GB/Radeon'
'V330-15IKB (i5-8250U/4GB/500GB/FHD/W10)' 'SmartBook 141'
'Tecra X40-D-10H' 'IdeaPad Y910-17ISK' 'GT73VR Titan' 'Chromebook 11'
'GT80S 6QE' 'Omen 17-AN010nv' 'Ideapad 320-15IKBR'
'TP501UA-CJ131T (i5-7200U/8GB/1TB/W10)' 'Inspiron 3179'
'Notebook Odyssey' 'V320-17ISK (i3-6006U/4GB/500GB/FHD/No'
'IdeaPad 110-15ISK' 'Latitude 5289' 'EliteBook 850' 'Aspire 1'

'Laptop MSI' 'GS63VR 7RF' 'Tecra Z50-C-144' 'IdeaPad 310-15IKB'
'Swift SF114-31-P5HY' 'Inspiron 7559' 'FX753VD-GC007T (i7-7700HQ/8GB/1TB'
'GT62VR 7RE' 'CB5-132T-C9KK (N3160/4GB/32GB/Chrome' 'LifeBook A557'
'SmartBook 140' 'Q304UA-BHI5T11 (i5-7200U/6GB/1TB/FHD/W10)' 'ZenBook 3'
'V330-15IKB (i5-8250U/4GB/256GB/FHD/W10)' 'Ideapad 320-15ISK'
'X541NA-GO414T (N3350/8GB/1TB/W10)' 'IdeaPad 100S-14IBR'
'17-AK091ND (A9-9420/8GB/1TB/W10)' 'ROG GL553VE-FY022' 'Extensa EX2540'
'Portege Z30-C-16J' 'ROG G701VI'
'A715-71G-59DH (i5-7300HQ/8GB/1TB/GeForce' 'GL62M 7REX' 'Tecra A50-D-11M'
'IdeaPad Y700-15ISK' 'Latitude E7470' 'Ideapad 320-15IAP'
'15-ay047nv (i3-6006U/6GB/1TB/Radeon' 'GP72VR Leopard' 'Latitude 3580'
'15-bs012nv (i7-7500U/8GB/1TB/Radeon' 'Tecra Z50-D-10E'
'V310-15ISK (i5-7200U/8GB/1TB' 'Yoga 720-13IKB' 'Pavilion X360'
'GP62 7RDX' 'Chromebook X360' 'Gram 15Z975' 'Aspire VX5-591G' 'GV62M 7RD'
'L502NA-GO052T (N3350/4GB/128GB/W10)' 'Alienware 15' '17-bs000nv I3'
'Yoga 730' '17-Y002nv (A10-9600P/6GB/2TB/Radeon'
'V110-15ISK (3855U/4GB/500GB/W10)' 'Chromebook 14' 'IdeaPad 520s-14IKB'
'TravelMate B117-M' 'Chromebook Flip' 'Portege Z30T-C-133'
'15-bs011nv (i7-7500U/4GB/500GB/Radeon'
'V310-15IKB (i5-7200U/4GB/1TB/FHD/W10)'
'V310-15ISK (i3-6006U/4GB/500GB/No' 'ThinkPad P51s' 'Thinkpad T460p'
'17-ak002nv (A10-9620P/6GB/2TB/Radeon'
'110-15ACL (A6-7310/4GB/500GB/W10)' 'Smartbook 142'
'V310-15IKB (i5-7200U/4GB/1TB/No' 'Inspiron 5378'
'15-BW037na (A9-9420/4GB/1TB/Radeon' 'Predator 17'
'15-BW091ND (A9-9420/6GB/1TB' 'Extensa EX2540-58KR'
'V310-15IKB (i7-7500U/4GB/1TB/FHD/W10)' 'ZBook 15' 'Inspiron 7560'
'Tecra X40-D-10G' 'Flex 5' 'Thinkpad P51s' 'Notebook 9' 'Zbook 17'
'N23 (N3060/4GB/128GB/W10)' 'X550VX-XX015D (i5-6300HQ/4GB/1TB/GeForce'
'Thinkpad T460' 'Pro P2540UA-XO0192R' 'Yoga 900-13ISK'
'15-cb003na (i5-7300HQ/8GB/1TB' 'Latitude 7280' 'Zenbook UX330UA-AH5Q'
'TravelMate P238-M' 'X751NV-TY001 (N4200/4GB/1TB/GeForce'
'Tecra A40-C-1E5' 'EliteBook 820'
'Q524UQ-BHI7T15 (i7-7500U/12GB/2TB/GeForce' 'Thinkpad P50' 'Vivobook Max'
'Rog G752VS-BA171T' 'Tecra Z40-C-161' 'IdeaPad 110-15IBR' 'GS43VR 7RE'
'GL62M (i5-7300HQ/8GB/1TB' 'Predator G9-793'
'FX502VM-DM560T (i7-7700HQ/8GB/1TB' 'K146 (N3350/4GB/32GB/W10)'
'Yoga 510-15IKB' 'R417NA-RS01 (N3350/4GB/32GB/W10)' 'Pro P2540UA-XS51'
'Latitude 3180' '15-ba043na (A12-9700P/8GB/2TB/W10)' 'Omen 17-an012dx'
'Thinkpad T470s' 'Blade Stealth' 'Latitude 3480'
'V110-15ISK (i3-6006U/4GB/500GB/W10)' 'Tecra X40-D-10Z' 'GL62M 7RD'
'Rog GL702VS-BA023T' 'N42-20 Chromebook'
'R558UA-DM966T (i5-7200U/8GB/128GB/FHD/W10)' 'Rog GL702VM-GC017T'
'ZenBook UX310UQ-GL026T' 'Rog GL502VM-DS74' 'Inspiron 5767'
'ThinkPad T470p' 'K556UR-DM621T (i7-7500U/8GB/256GB/GeForce'
'X541NA (N4200/4GB/1TB/W10)' 'Inspiron 5368' 'Portege X30-D-10X'
'Portégé Z30-C-188' 'TMX349-G2-M-50FS (i5-7200U/8GB/256GB/FHD/W10)'
'Tecra A50-D-11D' 'X541NA-GO121 (N4200/4GB/1TB/Linux)' 'Pavilion x360'

'VivoBook L402NA' 'IdeaPad 510-15ISK' 'Rog GL753VD-GC082T'
'Chromebook C731-C78G' 'Probook 640' 'Envy x360' 'GS73VR Stealth'
'Portege X30-D-10V' 'G701VO-IH74K (i7-6820HK/32GB/2x' 'Gram 15Z970'
'Chromebook CB5-571-C1DZ' 'Gram 14Z970' 'Elitebook Folio'
'IdeaPad 510-15IKB' 'GE72VR 6RF' 'Envy 13-AB077cl' 'Tecra Z50-C-140'
'Probook 650' 'Tecra Z40-C-12X' 'GP62M Leopard' 'Omen 17-W006na'
'X751SV-TY001T (N3710/4GB/1TB/GeForce' 'TravelMate P259-G2'
'Tecra A50-C-1ZV' 'Yoga 700-11ISK' 'IdeaPad Y700-15ACZ' 'Insprion 5767'
'ZBook Studio' 'Portege Z30-C-1CW' 'ProBook x360' 'Chromebook C738T-C2EJ'
'Portege Z30-C-16Z' 'Aspire F5-573G-510L' 'Portege X20W-D-10V'
'Tecra A40-C-1DF' 'ThinkPad T460' 'Q534UX-BHI7T19 (i7-7500U/16GB/2TB'
'15-bs053od (i7-7500U/6GB/1TB/W10)' 'Rog GL753VE-DS74' 'Inspiron 7579'
'Portege Z30-C-1CV' 'LifeBook A556' 'Tecra A40-C-1KF'
'15-bs005nv (i3-6006U/4GB/1TB' 'V110-15IAP (N3350/4GB/128GB/No'
'ThinkPad T560' 'ZenBook UX310UA-FB485T' 'Spectre 13-V111dx'
'Aspire ES1-533' 'Rog GL553VE-DS74' 'Nitro 5' 'ENVY -'
'Portege Z30-C-16H' 'Portege A30-C-1CZ' 'ThinkPad P70' 'Tecra Z40-C-12Z'
'Inspiron 5568' 'Portégé Z30-C-16K' 'Spectre 13-V100nv' 'Latitude E5570'
'Tecra Z40-C-136' 'Yoga 500-15ISK' 'V142 (X5-Z8350/2GB/32GB/W10)'
'Tecra A50-C-218' 'Thinkpad L560' 'GT72S Dominator' 'IdeaPad Y900-17ISK'
'Chromebook C202SA' 'Noteb Pav' 'Inspiron 5578' '250 G5' 'Aspire ES1-523'
'Inspiron 7378' 'GT62VR 6RD' 'Rog G752VL-GC088D' 'GS63VR 6RF'
'ROG G701VO' 'Latitude 3570' 'IdeaPad 300-17ISK' 'Ideapad 700-15ISK'
'GT72VR Dominator' 'V110-15ISK (i5-6200U/4GB/500GB/W10)'
'Yoga 900S-12ISK' 'Chromebook 13' 'Rog GL702VM-GC354T' 'Aspire F5-573G'
'GS70 Stealth' 'G752VY-GC162T (i7-6700HQ/16GB/1TB' 'Latitude E5270'
'Chromebook 15' 'GE72 Apache' '15-bw011nv (A6-9220/4GB/1TB/FHD/W10)'
'Rog GL552VW-CN470T' 'Vostro 3559' 'V110-15ISK (i3-6006U/4GB/128GB/W10)'
'Spectre Pro' 'Portege X30-D-10K' 'Rog GL752VW-T4308T'
'V131 (X5-Z8350/4GB/32GB/FHD/W10)' 'Omen -'
'15-bs078cl (i7-7500U/8GB/2TB/W10)' 'ThinkPad P40'
'L403NA-GA013TS (N3350/4GB/32GB/W10)' 'IdeaPad 500-15ISK' 'GP62M 7RDX'
'V110-15ISK (i3-6006U/4GB/1TB/No' '15-BA015wm (E2-7110/4GB/500GB/W10)'
'B51-80 (i5-6200U/8GB/1TB/Radeon' '15-bw002nv (A6-9220/4GB/256GB/Radeon'
'GP72M 7REX' 'ThinkPad T460s' 'B51-80 (i5-6200U/8GB/1008GB/Radeon'
'GS40 Phantom' 'Pavilion 15-cb003nv' 'IdeaPad 310-15ISK' '250 G4'
'320-15ISK (i3-6006U/4GB/1TB/GeForce' 'Stream 14-AX000nv' 'PL60 7RD'
'X553SA-XX021T (N3050/4GB/500GB/W10)' 'V110-15ISK (i5-6200U/4GB/500GB/No'
'UX410UA-GV097T (i3-7100U/4GB/256GB/FHD/W10)'
'B51-80 (i7-6500U/4GB/1008GB/FHD/W7)' 'GS60 Ghost' 'Pavilion 15-BC000nv'
'Rog GL552VW-DM201T' 'Chromebook Plus' 'Pavilion Power'
'V110-15ISK (i3-6006U/4GB/1TB/Radeon' 'Rog G752VY-GC229T' 'GS73VR 7RF'
'FX502VM-DM105T (i7-6700HQ/8GB/1TB/GeForce'
'15-bs025nv (i5-7200U/8GB/256GB/W10)' 'Aspire E5-774G'
'FX502VM-AS73 (i7-7700HQ/16GB/1TB' 'C740-C9QX (3205U/2GB/32GB/Chrome'
'E5 774G' 'SP714-51 (i7-7Y75/8GB/256GB/FHD/W10)' 'Thinkpad T560'
'GP62MVR 6RF' '15-bw009nv (A12-9720P/6GB/1TB/Radeon' 'Latitude E7270'
'X540SA-RBPDN09 (N3710/4GB/1TB/W10)' 'GL62M 7RDX' 'GE72VR Apache'

```
 '15-bs023nv (i3-6006U/4GB/1TB/FHD/W10)' 'GL62 6QF' 'ZenBook UX310UA-WB71'
 'Inspiron 7779' 'Rog GL553VE-FY052T' 'Rog GL502VS'
 'V510-15IKB (i5-7200U/8GB/256GB/FHD/No' 'ThinkPad L460'
 'X541NA-GO020T (N3350/4GB/1TB/W10)' 'Rog G752VT-GC073T'
 'B51-80 (i7-6500U/8GB/1008GB/Radeon' 'GE62 Apache' 'Yoga 500-14IBD'
 'ZenBook UX305CA-UBM1' 'Aspire ES1-531' 'Pavilion 15-AW003nv'
 'Stream 11-Y000na' 'X556UJ-XO044T (i7-6500U/4GB/500GB/GeForce'
 'Yoga 500-14ISK' '15-AC110nv (i7-6500U/6GB/1TB/Radeon'
 'X553SA-XX031T (N3050/4GB/500GB/W10)']


--------------------------------------------------------------------------------
--------------------
Unique values in TypeName is : ['Ultrabook' 'Notebook' 'Netbook' 'Gaming' '2 in
1 Convertible'
 'Workstation']


--------------------------------------------------------------------------------
--------------------
Unique values in ScreenResolution is : ['IPS Panel Retina Display 2560x1600'
'1440x900' 'Full HD 1920x1080'
 'IPS Panel Retina Display 2880x1800' '1366x768'
 'IPS Panel Full HD 1920x1080' 'IPS Panel Retina Display 2304x1440'
 'IPS Panel Full HD / Touchscreen 1920x1080'
 'Full HD / Touchscreen 1920x1080' 'Touchscreen / Quad HD+ 3200x1800'
 'IPS Panel Touchscreen 1920x1200' 'Touchscreen 2256x1504'
 'Quad HD+ / Touchscreen 3200x1800' 'IPS Panel 1366x768'
 'IPS Panel 4K Ultra HD / Touchscreen 3840x2160'
 'IPS Panel Full HD 2160x1440' '4K Ultra HD / Touchscreen 3840x2160'
 'Touchscreen 2560x1440' '1600x900' 'IPS Panel 4K Ultra HD 3840x2160'
 '4K Ultra HD 3840x2160' 'Touchscreen 1366x768'
 'IPS Panel Full HD 1366x768' 'IPS Panel 2560x1440'
 'IPS Panel Full HD 2560x1440' 'IPS Panel Retina Display 2736x1824'
 'Touchscreen 2400x1600' '2560x1440' 'IPS Panel Quad HD+ 2560x1440'
 'IPS Panel Quad HD+ 3200x1800'
 'IPS Panel Quad HD+ / Touchscreen 3200x1800'
 'IPS Panel Touchscreen 1366x768' '1920x1080'
 'IPS Panel Full HD 1920x1200'
 'IPS Panel Touchscreen / 4K Ultra HD 3840x2160'
 'IPS Panel Touchscreen 2560x1440' 'Touchscreen / Full HD 1920x1080'
 'Quad HD+ 3200x1800' 'Touchscreen / 4K Ultra HD 3840x2160'
 'IPS Panel Touchscreen 2400x1600']


--------------------------------------------------------------------------------
--------------------
Unique values in Cpu is : ['Intel Core i5 2.3GHz' 'Intel Core i5 1.8GHz'
```

'Intel Core i5 7200U 2.5GHz' 'Intel Core i7 2.7GHz'
'Intel Core i5 3.1GHz' 'AMD A9-Series 9420 3GHz' 'Intel Core i7 2.2GHz'
'Intel Core i7 8550U 1.8GHz' 'Intel Core i5 8250U 1.6GHz'
'Intel Core i3 6006U 2GHz' 'Intel Core i7 2.8GHz'
'Intel Core M m3 1.2GHz' 'Intel Core i7 7500U 2.7GHz'
'Intel Core i7 2.9GHz' 'Intel Core i3 7100U 2.4GHz'
'Intel Atom x5-Z8350 1.44GHz' 'Intel Core i5 7300HQ 2.5GHz'
'AMD E-Series E2-9000e 1.5GHz' 'Intel Core i5 1.6GHz'
'Intel Core i7 8650U 1.9GHz' 'Intel Atom x5-Z8300 1.44GHz'
'AMD E-Series E2-6110 1.5GHz' 'AMD A6-Series 9220 2.5GHz'
'Intel Celeron Dual Core N3350 1.1GHz' 'Intel Core i3 7130U 2.7GHz'
'Intel Core i7 7700HQ 2.8GHz' 'Intel Core i5 2.0GHz'
'AMD Ryzen 1700 3GHz' 'Intel Pentium Quad Core N4200 1.1GHz'
'Intel Atom x5-Z8550 1.44GHz' 'Intel Celeron Dual Core N3060 1.6GHz'
'Intel Core i5 1.3GHz' 'AMD FX 9830P 3GHz' 'Intel Core i7 7560U 2.4GHz'
'AMD E-Series 6110 1.5GHz' 'Intel Core i5 6200U 2.3GHz'
'Intel Core M 6Y75 1.2GHz' 'Intel Core i5 7500U 2.7GHz'
'Intel Core i3 6006U 2.2GHz' 'AMD A6-Series 9220 2.9GHz'
'Intel Core i7 6920HQ 2.9GHz' 'Intel Core i5 7Y54 1.2GHz'
'Intel Core i7 7820HK 2.9GHz' 'Intel Xeon E3-1505M V6 3GHz'
'Intel Core i7 6500U 2.5GHz' 'AMD E-Series 9000e 1.5GHz'
'AMD A10-Series A10-9620P 2.5GHz' 'AMD A6-Series A6-9220 2.5GHz'
'Intel Core i5 2.9GHz' 'Intel Core i7 6600U 2.6GHz'
'Intel Core i3 6006U 2.0GHz' 'Intel Celeron Dual Core 3205U 1.5GHz'
'Intel Core i7 7820HQ 2.9GHz' 'AMD A10-Series 9600P 2.4GHz'
'Intel Core i7 7600U 2.8GHz' 'AMD A8-Series 7410 2.2GHz'
'Intel Celeron Dual Core 3855U 1.6GHz'
'Intel Pentium Quad Core N3710 1.6GHz' 'AMD A12-Series 9720P 2.7GHz'
'Intel Core i5 7300U 2.6GHz' 'AMD A12-Series 9720P 3.6GHz'
'Intel Celeron Quad Core N3450 1.1GHz'
'Intel Celeron Dual Core N3060 1.60GHz' 'Intel Core i5 6440HQ 2.6GHz'
'Intel Core i7 6820HQ 2.7GHz' 'AMD Ryzen 1600 3.2GHz'
'Intel Core i7 7Y75 1.3GHz' 'Intel Core i5 7440HQ 2.8GHz'
'Intel Core i7 7660U 2.5GHz' 'Intel Core i7 7700HQ 2.7GHz'
'Intel Core M m3-7Y30 2.2GHz' 'Intel Core i5 7Y57 1.2GHz'
'Intel Core i7 6700HQ 2.6GHz' 'Intel Core i3 6100U 2.3GHz'
'AMD A10-Series 9620P 2.5GHz' 'AMD E-Series 7110 1.8GHz'
'Intel Celeron Dual Core N3350 2.0GHz' 'AMD A9-Series A9-9420 3GHz'
'Intel Core i7 6820HK 2.7GHz' 'Intel Core M 7Y30 1.0GHz'
'Intel Xeon E3-1535M v6 3.1GHz' 'Intel Celeron Quad Core N3160 1.6GHz'
'Intel Core i5 6300U 2.4GHz' 'Intel Core i3 6100U 2.1GHz'
'AMD E-Series E2-9000 2.2GHz' 'Intel Celeron Dual Core N3050 1.6GHz'
'Intel Core M M3-6Y30 0.9GHz' 'AMD A9-Series 9420 2.9GHz'
'Intel Core i5 6300HQ 2.3GHz' 'AMD A6-Series 7310 2GHz'
'Intel Atom Z8350 1.92GHz' 'Intel Xeon E3-1535M v5 2.9GHz'
'Intel Core i5 6260U 1.8GHz' 'Intel Pentium Dual Core N4200 1.1GHz'
'Intel Celeron Quad Core N3710 1.6GHz' 'Intel Core M 1.2GHz'
'AMD A12-Series 9700P 2.5GHz' 'Intel Core i7 7500U 2.5GHz'

```
 'Intel Pentium Dual Core 4405U 2.1GHz' 'AMD A4-Series 7210 2.2GHz'
 'Intel Core i7 6560U 2.2GHz' 'Intel Core M m7-6Y75 1.2GHz'
 'AMD FX 8800P 2.1GHz' 'Intel Core M M7-6Y75 1.2GHz'
 'Intel Core i5 7200U 2.50GHz' 'Intel Core i5 7200U 2.70GHz'
 'Intel Atom X5-Z8350 1.44GHz' 'Intel Core i5 7200U 2.7GHz'
 'Intel Core M 1.1GHz' 'Intel Pentium Dual Core 4405Y 1.5GHz'
 'Intel Pentium Quad Core N3700 1.6GHz' 'Intel Core M 6Y54 1.1GHz'
 'Intel Core i7 6500U 2.50GHz' 'Intel Celeron Dual Core N3350 2GHz'
 'Samsung Cortex A72&A53 2.0GHz' 'AMD E-Series 9000 2.2GHz'
 'Intel Core M 6Y30 0.9GHz' 'AMD A9-Series 9410 2.9GHz']


--------------------------------------------------------------------------------
--------------------
Unique values in Ram is : ['8GB' '16GB' '4GB' '2GB' '12GB' '6GB' '32GB' '24GB'
'64GB']


--------------------------------------------------------------------------------
--------------------
Unique values in Memory is : ['128GB SSD' '128GB Flash Storage' '256GB SSD'
'512GB SSD' '500GB HDD'
 '256GB Flash Storage' '1TB HDD' '32GB Flash Storage'
 '128GB SSD +  1TB HDD' '256GB SSD +  256GB SSD' '64GB Flash Storage'
 '256GB SSD +  1TB HDD' '256GB SSD +  2TB HDD' '32GB SSD' '2TB HDD'
 '64GB SSD' '1.0TB Hybrid' '512GB SSD +  1TB HDD' '1TB SSD'
 '256GB SSD +  500GB HDD' '128GB SSD +  2TB HDD' '512GB SSD +  512GB SSD'
 '16GB SSD' '16GB Flash Storage' '512GB SSD +  256GB SSD'
 '512GB SSD +  2TB HDD' '64GB Flash Storage +  1TB HDD' '180GB SSD'
 '1TB HDD +  1TB HDD' '32GB HDD' '1TB SSD +  1TB HDD'
 '512GB Flash Storage' '128GB HDD' '240GB SSD' '8GB SSD' '508GB Hybrid'
 '1.0TB HDD' '512GB SSD +  1.0TB Hybrid' '256GB SSD +  1.0TB Hybrid']


--------------------------------------------------------------------------------
--------------------
Unique values in Gpu is : ['Intel Iris Plus Graphics 640' 'Intel HD Graphics
6000'
 'Intel HD Graphics 620' 'AMD Radeon Pro 455'
 'Intel Iris Plus Graphics 650' 'AMD Radeon R5' 'Intel Iris Pro Graphics'
 'Nvidia GeForce MX150' 'Intel UHD Graphics 620' 'Intel HD Graphics 520'
 'AMD Radeon Pro 555' 'AMD Radeon R5 M430' 'Intel HD Graphics 615'
 'AMD Radeon Pro 560' 'Nvidia GeForce 940MX' 'Intel HD Graphics 400'
 'Nvidia GeForce GTX 1050' 'AMD Radeon R2' 'AMD Radeon 530'
 'Nvidia GeForce 930MX' 'Intel HD Graphics' 'Intel HD Graphics 500'
 'Nvidia GeForce 930MX ' 'Nvidia GeForce GTX 1060' 'Nvidia GeForce 150MX'
 'Intel Iris Graphics 540' 'AMD Radeon RX 580' 'Nvidia GeForce 920MX'
 'AMD Radeon R4 Graphics' 'AMD Radeon 520' 'Nvidia GeForce GTX 1070'
```

```
 'Nvidia GeForce GTX 1050 Ti' 'Nvidia GeForce MX130' 'AMD R4 Graphics'
 'Nvidia GeForce GTX 940MX' 'AMD Radeon RX 560' 'Nvidia GeForce 920M'
 'AMD Radeon R7 M445' 'AMD Radeon RX 550' 'Nvidia GeForce GTX 1050M'
 'Intel HD Graphics 515' 'AMD Radeon R5 M420' 'Intel HD Graphics 505'
 'Nvidia GTX 980 SLI' 'AMD R17M-M1-70' 'Nvidia GeForce GTX 1080'
 'Nvidia Quadro M1200' 'Nvidia GeForce 920MX ' 'Nvidia GeForce GTX 950M'
 'AMD FirePro W4190M ' 'Nvidia GeForce GTX 980M' 'Intel Iris Graphics 550'
 'Nvidia GeForce 930M' 'Intel HD Graphics 630' 'AMD Radeon R5 430'
 'Nvidia GeForce GTX 940M' 'Intel HD Graphics 510' 'Intel HD Graphics 405'
 'AMD Radeon RX 540' 'Nvidia GeForce GT 940MX' 'AMD FirePro W5130M'
 'Nvidia Quadro M2200M' 'AMD Radeon R4' 'Nvidia Quadro M620'
 'AMD Radeon R7 M460' 'Intel HD Graphics 530' 'Nvidia GeForce GTX 965M'
 'Nvidia GeForce GTX1080' 'Nvidia GeForce GTX1050 Ti'
 'Nvidia GeForce GTX 960M' 'AMD Radeon R2 Graphics' 'Nvidia Quadro M620M'
 'Nvidia GeForce GTX 970M' 'Nvidia GeForce GTX 960<U+039C>'
 'Intel Graphics 620' 'Nvidia GeForce GTX 960' 'AMD Radeon R5 520'
 'AMD Radeon R7 M440' 'AMD Radeon R7' 'Nvidia Quadro M520M'
 'Nvidia Quadro M2200' 'Nvidia Quadro M2000M' 'Intel HD Graphics 540'
 'Nvidia Quadro M1000M' 'AMD Radeon 540' 'Nvidia GeForce GTX 1070M'
 'Nvidia GeForce GTX1060' 'Intel HD Graphics 5300' 'AMD Radeon R5 M420X'
 'AMD Radeon R7 Graphics' 'Nvidia GeForce 920' 'Nvidia GeForce 940M'
 'Nvidia GeForce GTX 930MX' 'AMD Radeon R7 M465' 'AMD Radeon R3'
 'Nvidia GeForce GTX 1050Ti' 'AMD Radeon R7 M365X' 'AMD Radeon R9 M385'
 'Intel HD Graphics 620 ' 'Nvidia Quadro 3000M' 'Nvidia GeForce GTX 980 '
 'AMD Radeon R5 M330' 'AMD FirePro W4190M' 'AMD FirePro W6150M'
 'AMD Radeon R5 M315' 'Nvidia Quadro M500M' 'AMD Radeon R7 M360'
 'Nvidia Quadro M3000M' 'Nvidia GeForce 960M' 'ARM Mali T860 MP4']


--------------------------------------------------------------------------------
--------------------
Unique values in OpSys is : ['macOS' 'No OS' 'Windows 10' 'Mac OS X' 'Linux'
'Android' 'Windows 10 S'
 'Chrome OS' 'Windows 7']


--------------------------------------------------------------------------------
--------------------
Unique values in Weight is : ['1.37kg' '1.34kg' '1.86kg' '1.83kg' '2.1kg'
'2.04kg' '1.3kg' '1.6kg'
 '2.2kg' '0.92kg' '1.22kg' '0.98kg' '2.5kg' '1.62kg' '1.91kg' '2.3kg'
 '1.35kg' '1.88kg' '1.89kg' '1.65kg' '2.71kg' '1.2kg' '1.44kg' '2.8kg'
 '2kg' '2.65kg' '2.77kg' '3.2kg' '0.69kg' '1.49kg' '2.4kg' '2.13kg'
 '2.43kg' '1.7kg' '1.4kg' '1.8kg' '1.9kg' '3kg' '1.252kg' '2.7kg' '2.02kg'
 '1.63kg' '1.96kg' '1.21kg' '2.45kg' '1.25kg' '1.5kg' '2.62kg' '1.38kg'
 '1.58kg' '1.85kg' '1.23kg' '1.26kg' '2.16kg' '2.36kg' '2.05kg' '1.32kg'
 '1.75kg' '0.97kg' '2.9kg' '2.56kg' '1.48kg' '1.74kg' '1.1kg' '1.56kg'
 '2.03kg' '1.05kg' '4.4kg' '1.90kg' '1.29kg' '2.0kg' '1.95kg' '2.06kg'
```

```
'1.12kg' '1.42kg' '3.49kg' '3.35kg' '2.23kg' '4.42kg' '2.69kg' '2.37kg'
'4.7kg' '3.6kg' '2.08kg' '4.3kg' '1.68kg' '1.41kg' '4.14kg' '2.18kg'
'2.24kg' '2.67kg' '2.14kg' '1.36kg' '2.25kg' '2.15kg' '2.19kg' '2.54kg'
'3.42kg' '1.28kg' '2.33kg' '1.45kg' '2.79kg' '1.84kg' '2.6kg' '2.26kg'
'3.25kg' '1.59kg' '1.13kg' '1.78kg' '1.10kg' '1.15kg' '1.27kg' '1.43kg'
'2.31kg' '1.16kg' '1.64kg' '2.17kg' '1.47kg' '3.78kg' '1.79kg' '0.91kg'
'1.99kg' '4.33kg' '1.93kg' '1.87kg' '2.63kg' '3.4kg' '3.14kg' '1.94kg'
'1.24kg' '4.6kg' '4.5kg' '2.73kg' '1.39kg' '2.29kg' '2.59kg' '2.94kg'
'1.14kg' '3.8kg' '3.31kg' '1.09kg' '3.21kg' '1.19kg' '1.98kg' '1.17kg'
'4.36kg' '1.71kg' '2.32kg' '4.2kg' '1.55kg' '0.81kg' '1.18kg' '2.72kg'
'1.31kg' '0.920kg' '3.74kg' '1.76kg' '1.54kg' '2.83kg' '2.07kg' '2.38kg'
'3.58kg' '1.08kg' '2.20kg' '2.75kg' '1.70kg' '2.99kg' '1.11kg' '2.09kg'
'4kg' '3.0kg' '0.99kg' '3.52kg' '2.591kg' '2.21kg' '3.3kg' '2.191kg'
'2.34kg' '4.0kg']
```

Observations

- It is clear visible from the above data that features have alot text and for prediction we need numerical data.
- So to do so will need to do alot data(text) preprocessing.

[13]:
```
# Let's start with feature 'RAM' and 'Weight' : from the observations if we remove GB and kg the easialy can be converted into int.

df['Ram'] = df['Ram'].str.replace('GB','').astype('int32')

df['Weight'] = df['Weight'].str.replace('kg','').astype('float32')
```

[14]:
```
df.head()
```

[14]:

	Company	Product	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory
0	Apple	MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8	128GB SSD
1	Apple	Macbook Air	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8	128GB Flash Storage
2	HP	250 G6	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8	256GB SSD
3	Apple	MacBook Pro	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16	512GB SSD
4	Apple	MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8	256GB SSD

```
                            Gpu  OpSys  Weight  Price_euros
0  Intel Iris Plus Graphics 640  macOS    1.37      1339.69
1        Intel HD Graphics 6000  macOS    1.34       898.94
2         Intel HD Graphics 620  No OS    1.86       575.00
3            AMD Radeon Pro 455  macOS    1.83      2537.45
4  Intel Iris Plus Graphics 650  macOS    1.37      1803.60
```

[15]:
```
# Rename the 'Ram' -- 'Ram_GB' and 'Weight' -- 'Weight_Kg'

df.rename(columns = {'Ram':'Ram_GB','Weight':'Weight_KG'}, inplace = True)

df.head()
```

[15]:
```
  Company      Product   TypeName  Inches                    ScreenResolution  \
0   Apple  MacBook Pro  Ultrabook    13.3  IPS Panel Retina Display 2560x1600
1   Apple  Macbook Air  Ultrabook    13.3                              1440x900
2      HP       250 G6   Notebook    15.6                    Full HD 1920x1080
3   Apple  MacBook Pro  Ultrabook    15.4  IPS Panel Retina Display 2880x1800
4   Apple  MacBook Pro  Ultrabook    13.3  IPS Panel Retina Display 2560x1600

                          Cpu  Ram_GB              Memory  \
0        Intel Core i5 2.3GHz       8           128GB SSD
1        Intel Core i5 1.8GHz       8  128GB Flash Storage
2  Intel Core i5 7200U 2.5GHz       8           256GB SSD
3        Intel Core i7 2.7GHz      16           512GB SSD
4        Intel Core i5 3.1GHz       8           256GB SSD

                            Gpu  OpSys  Weight_KG  Price_euros
0  Intel Iris Plus Graphics 640  macOS       1.37      1339.69
1        Intel HD Graphics 6000  macOS       1.34       898.94
2         Intel HD Graphics 620  No OS       1.86       575.00
3            AMD Radeon Pro 455  macOS       1.83      2537.45
4  Intel Iris Plus Graphics 650  macOS       1.37      1803.60
```

2 Exploratory Data Analysis

[16]:
```
# Statistical Analysis: 5-Point Summary

df.describe()
```

[16]:
```
            Inches       Ram_GB    Weight_KG  Price_euros
count  1303.000000  1303.000000  1303.000000  1303.000000
mean     15.017191     8.382195     2.038734  1123.686992
std       1.426304     5.084665     0.665475   699.009043
min      10.100000     2.000000     0.690000   174.000000
25%      14.000000     4.000000     1.500000   599.000000
```

```
50%       15.600000      8.000000      2.040000    977.000000
75%       15.600000      8.000000      2.300000   1487.880000
max       18.400000     64.000000      4.700000   6099.000000
```

- From above info it is clear that 'Price_euro' has skewness in data.

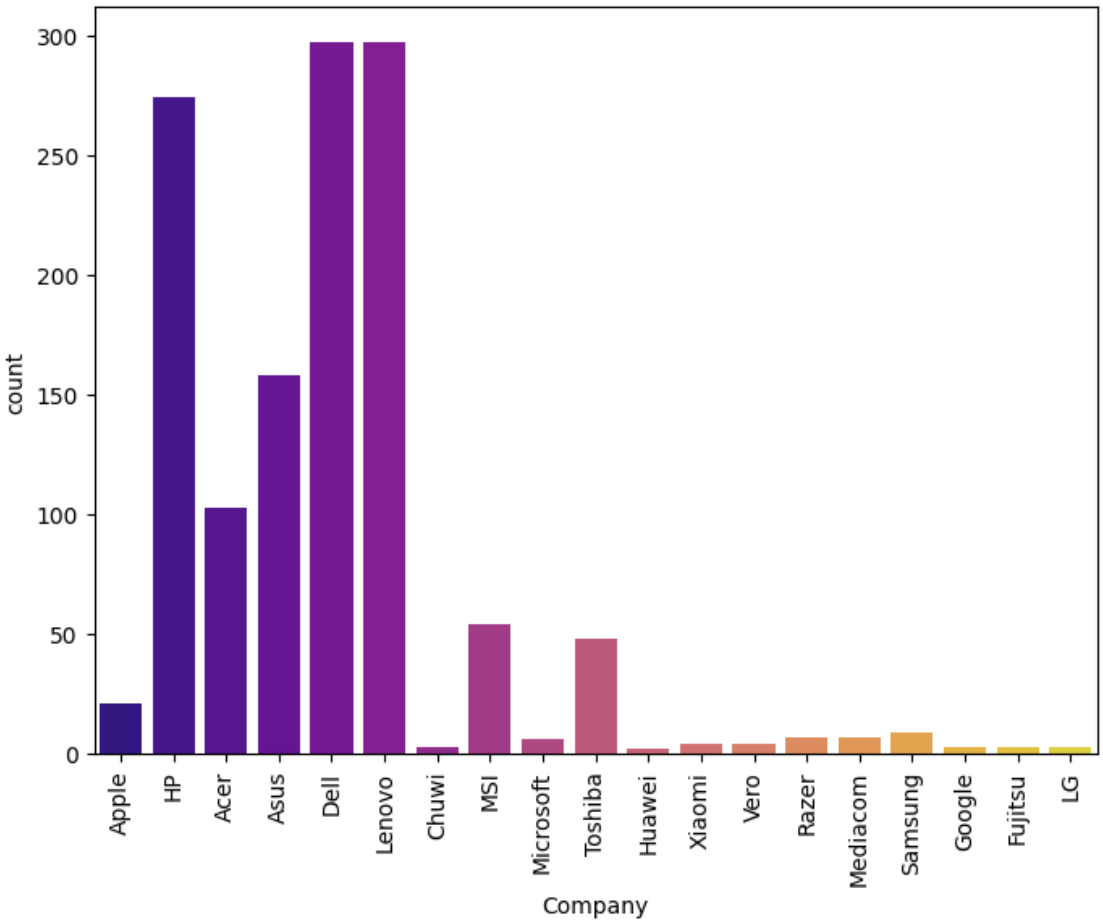
[17]:
```
# From above analysis, it is clear that 'Price' feature has some skewness in␣
↪data. So let's analyse the Feature 'Price_euro'

plt.figure(figsize = (12,6))
sns.distplot(df['Price_euros'], color = 'red')
plt.xlabel('Price')
plt.title('Price Distribution')
plt.show()
```

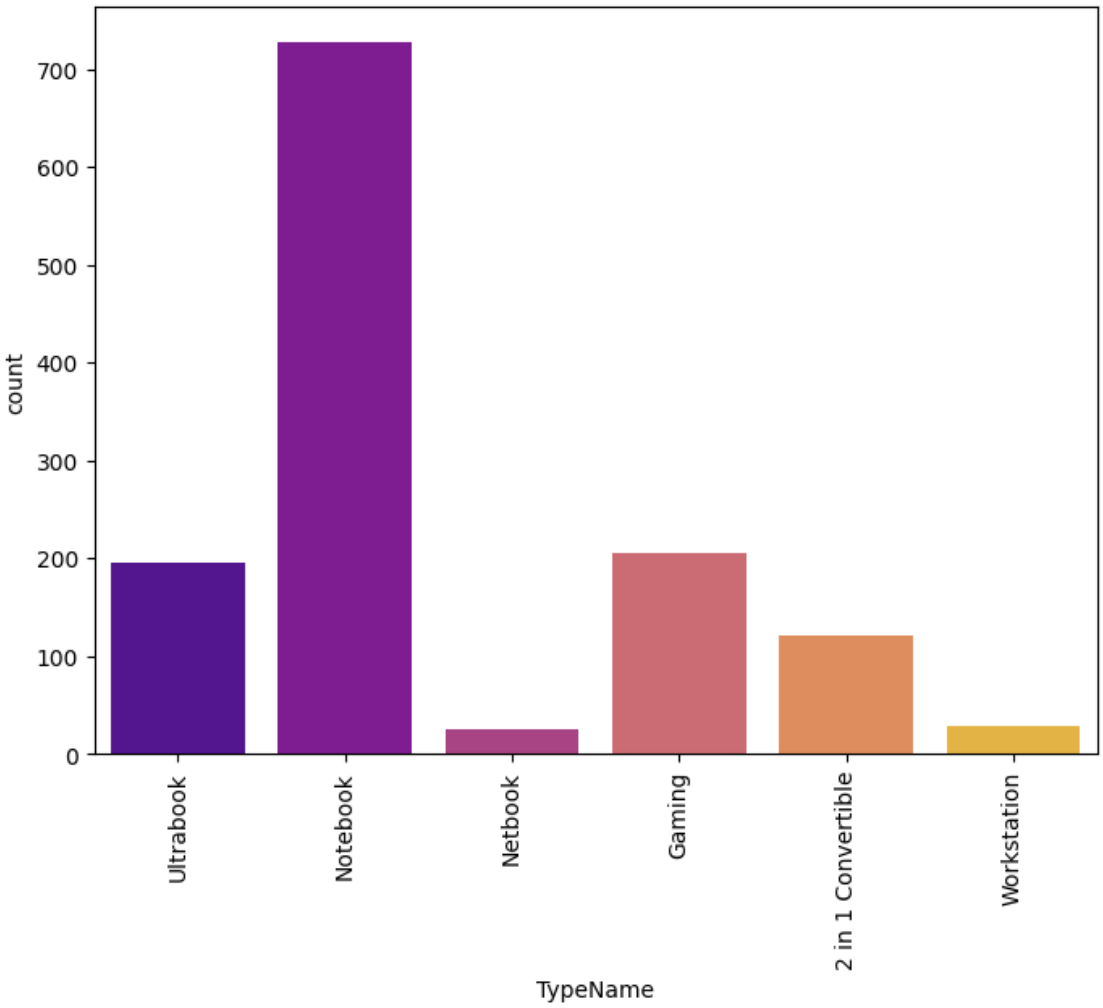
[18]:
```
df['Price_euros'].skew()
```

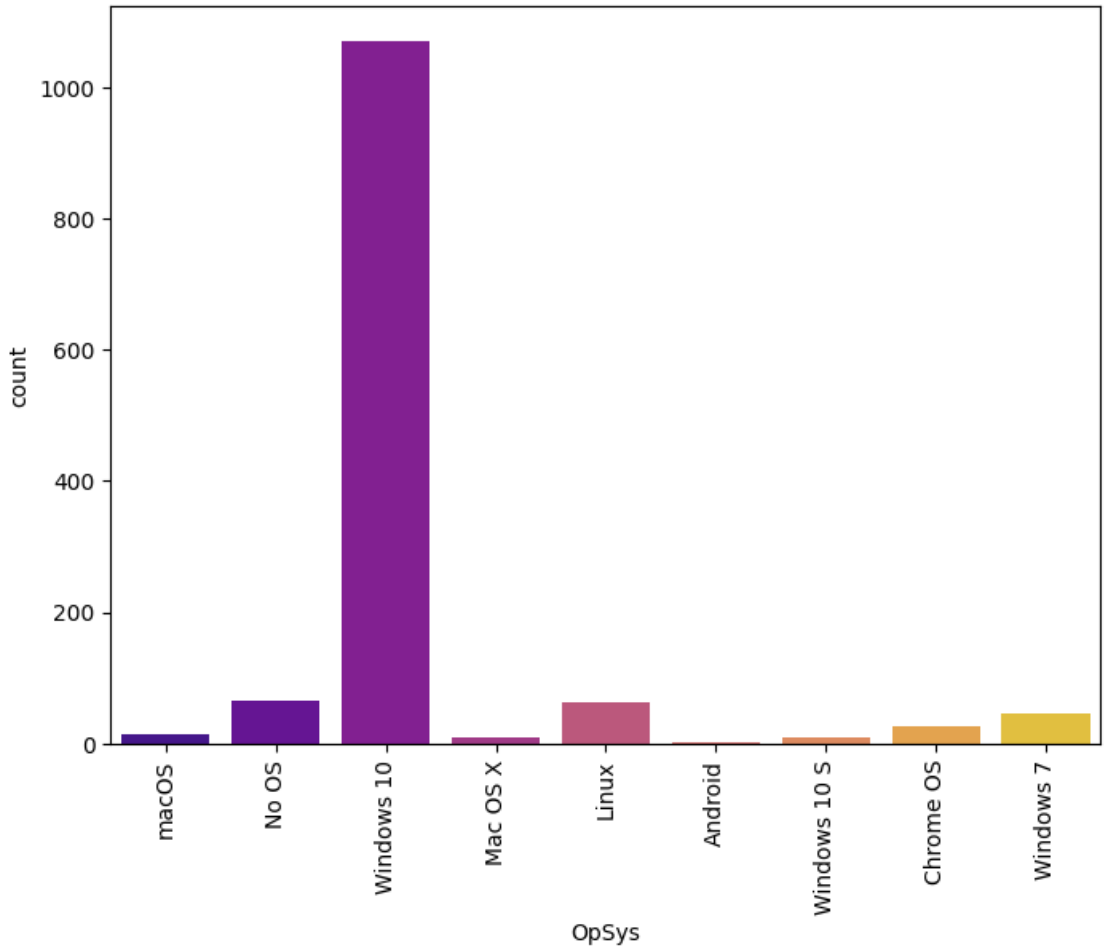
[18]:
```
1.5208655681688525
```

- Distribustion is slightly positive skeweed.

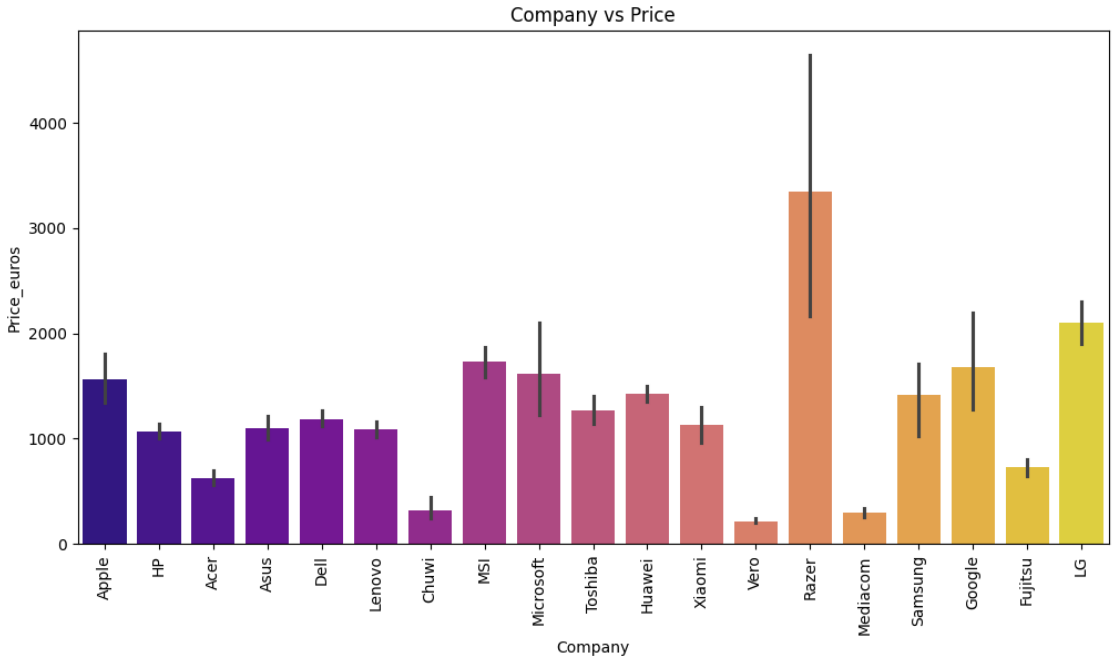
[19]:
```
# Countplot for categorical feature having very less unique values

for feature in ['Company', 'TypeName', 'OpSys']:

    plt.figure(figsize = (8,6))
    sns.countplot(data = df, x = df[feature], palette = 'plasma')
    plt.xticks(rotation = 'vertical')
```


300
250
200
150
100
50
0
count
Apple
HP
Acer
Asus
Dell
Lenovo
Chuwi
MSI
Microsoft
Toshiba
Huawei
Xiaomi
Vero
Razer
Mediacom
Samsung
Google
Fujitsu
LG
Company


700
600
500
400
300
200
100
0
count
Ultrabook
Notebook
Netbook
Gaming
2 in 1 Convertible
Workstation
TypeName



[20]:
```
# Inflation check with 'Company'

plt.figure(figsize = (12,6))
sns.barplot(x = df.Company, y = df.Price_euros, palette = 'plasma')
plt.title('Company vs Price')
plt.xticks(rotation = 'vertical')
plt.show()
```

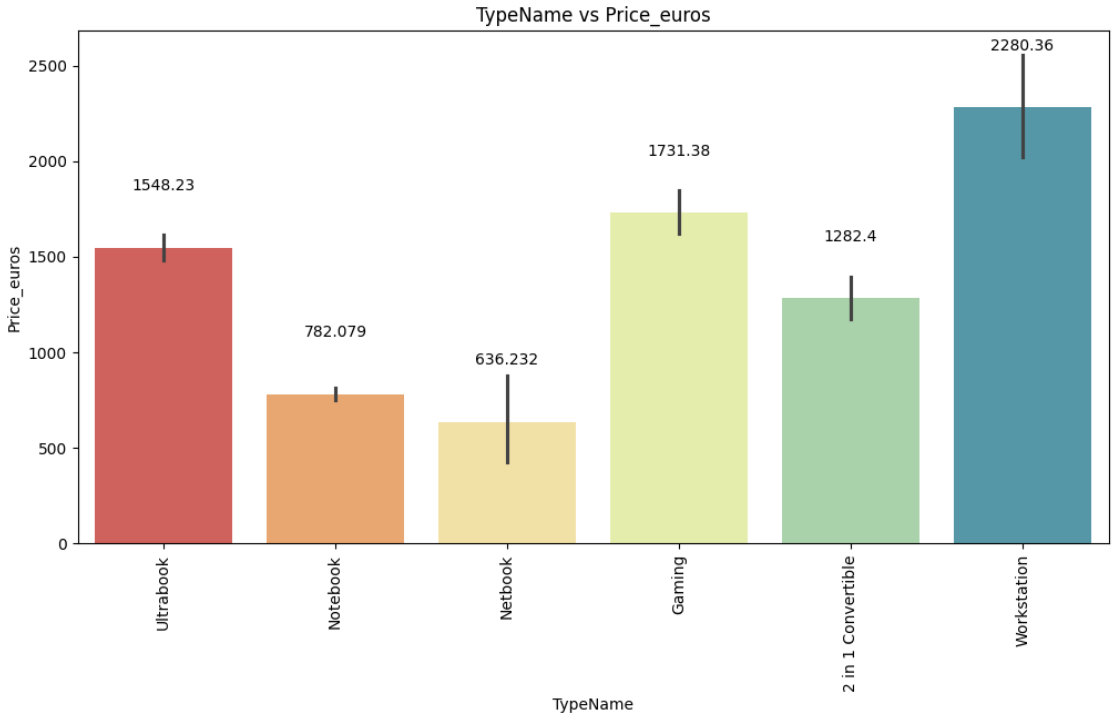


- From here we can get the price range of each brand laptops.

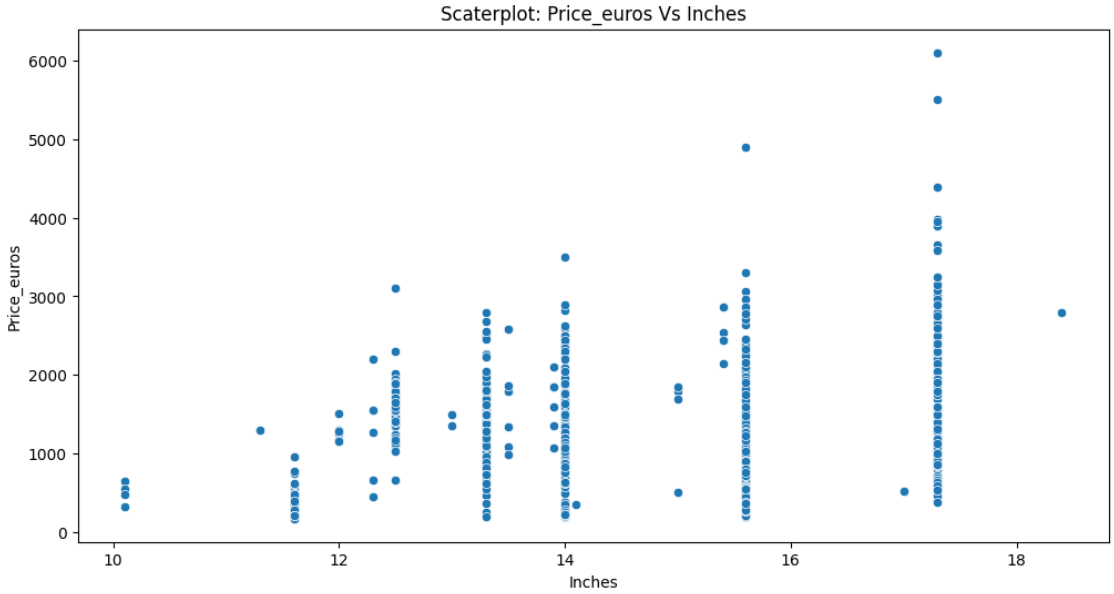
[21]:
```
df['TypeName'].value_counts()
```

[21]:
```
TypeName
Notebook              727
Gaming                205
Ultrabook             196
2 in 1 Convertible    121
Workstation            29
Netbook                25
Name: count, dtype: int64
```

[22]:
```
plt.figure(figsize = (12,6))
ax = sns.barplot(data = df, x = df['TypeName'], y = df['Price_euros'], palette␣
 ↪= 'Spectral')

for label in ax.containers:
    ax.bar_label(label, padding=35)
plt.title('TypeName vs Price_euros')
plt.xticks(rotation = 'vertical')
plt.show()
```



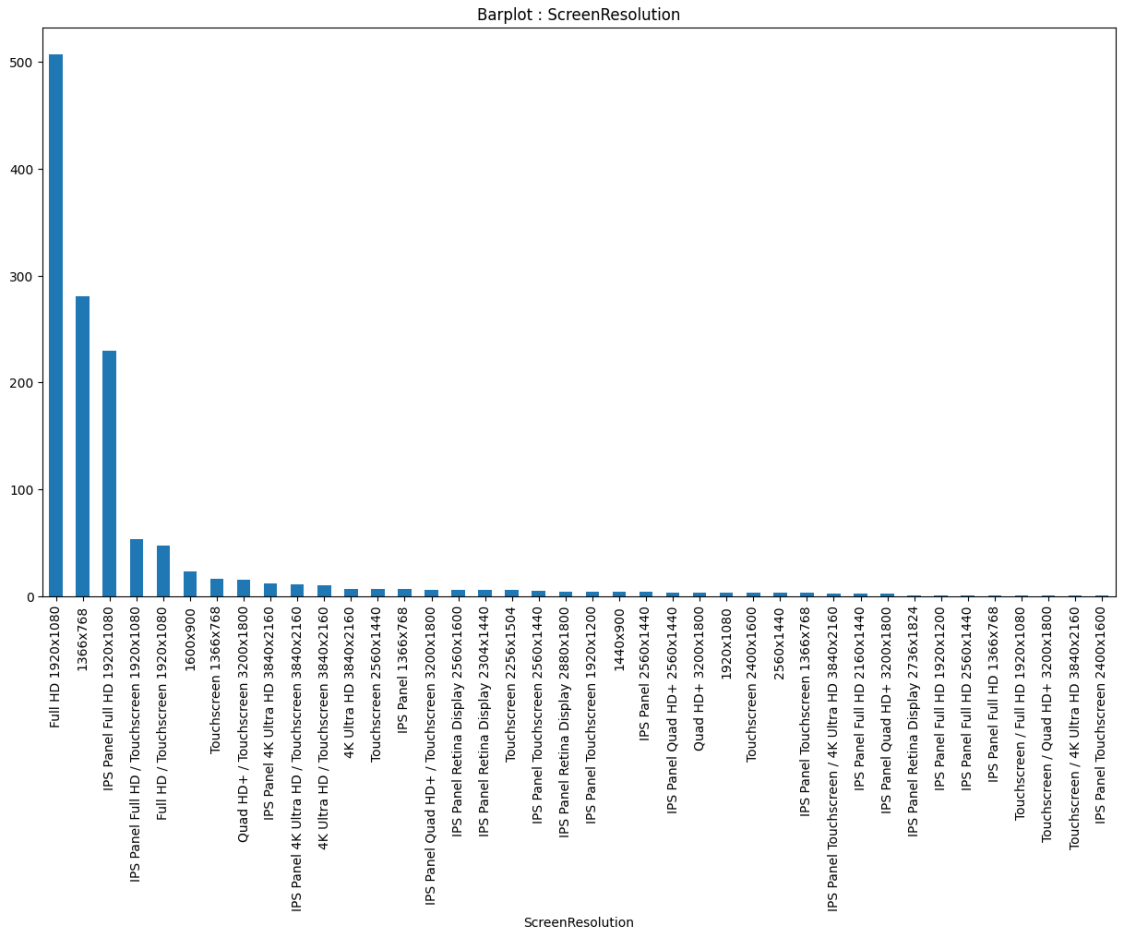
[23]:
```
# Price_euros vs Inches

plt.figure(figsize = (12,6))
sns.scatterplot(data = df, x = df['Inches'], y = df['Price_euros'])
plt.title('Scaterplot: Price_euros Vs Inches')
plt.show()
```



[24]:
```
df['ScreenResolution'].value_counts()
```

[24]:
```
ScreenResolution
Full HD 1920x1080                                507
1366x768                                         281
IPS Panel Full HD 1920x1080                      230
IPS Panel Full HD / Touchscreen 1920x1080         53
Full HD / Touchscreen 1920x1080                   47
1600x900                                          23
Touchscreen 1366x768                              16
Quad HD+ / Touchscreen 3200x1800                  15
IPS Panel 4K Ultra HD 3840x2160                   12
IPS Panel 4K Ultra HD / Touchscreen 3840x2160     11
4K Ultra HD / Touchscreen 3840x2160               10
4K Ultra HD 3840x2160                              7
Touchscreen 2560x1440                              7
IPS Panel 1366x768                                 7
IPS Panel Quad HD+ / Touchscreen 3200x1800         6
IPS Panel Retina Display 2560x1600                 6
IPS Panel Retina Display 2304x1440                 6
Touchscreen 2256x1504                              6
IPS Panel Touchscreen 2560x1440                    5
IPS Panel Retina Display 2880x1800                 4
IPS Panel Touchscreen 1920x1200                    4
1440x900                                           4
IPS Panel 2560x1440                                4
IPS Panel Quad HD+ 2560x1440                       3
Quad HD+ 3200x1800                                 3
1920x1080                                          3
Touchscreen 2400x1600                              3
2560x1440                                          3
IPS Panel Touchscreen 1366x768                     3
IPS Panel Touchscreen / 4K Ultra HD 3840x2160      2
IPS Panel Full HD 2160x1440                        2
IPS Panel Quad HD+ 3200x1800                       2
IPS Panel Retina Display 2736x1824                 1
IPS Panel Full HD 1920x1200                        1
IPS Panel Full HD 2560x1440                        1
IPS Panel Full HD 1366x768                         1
Touchscreen / Full HD 1920x1080                    1
Touchscreen / Quad HD+ 3200x1800                   1
Touchscreen / 4K Ultra HD 3840x2160                1
IPS Panel Touchscreen 2400x1600                    1
Name: count, dtype: int64
```

[25]:
```
plt.figure(figsize = (15,8))
df['ScreenResolution'].value_counts().plot(kind='bar')
plt.title('Barplot : ScreenResolution')
plt.show()
```



[26]:
```
# Create a feature 'TouchScreen'

df['TouchScreen'] = df['ScreenResolution'].apply(lambda element:1 if␣
 ↪'Touchscreen' in element else 0)

df.head()
```

[26]:

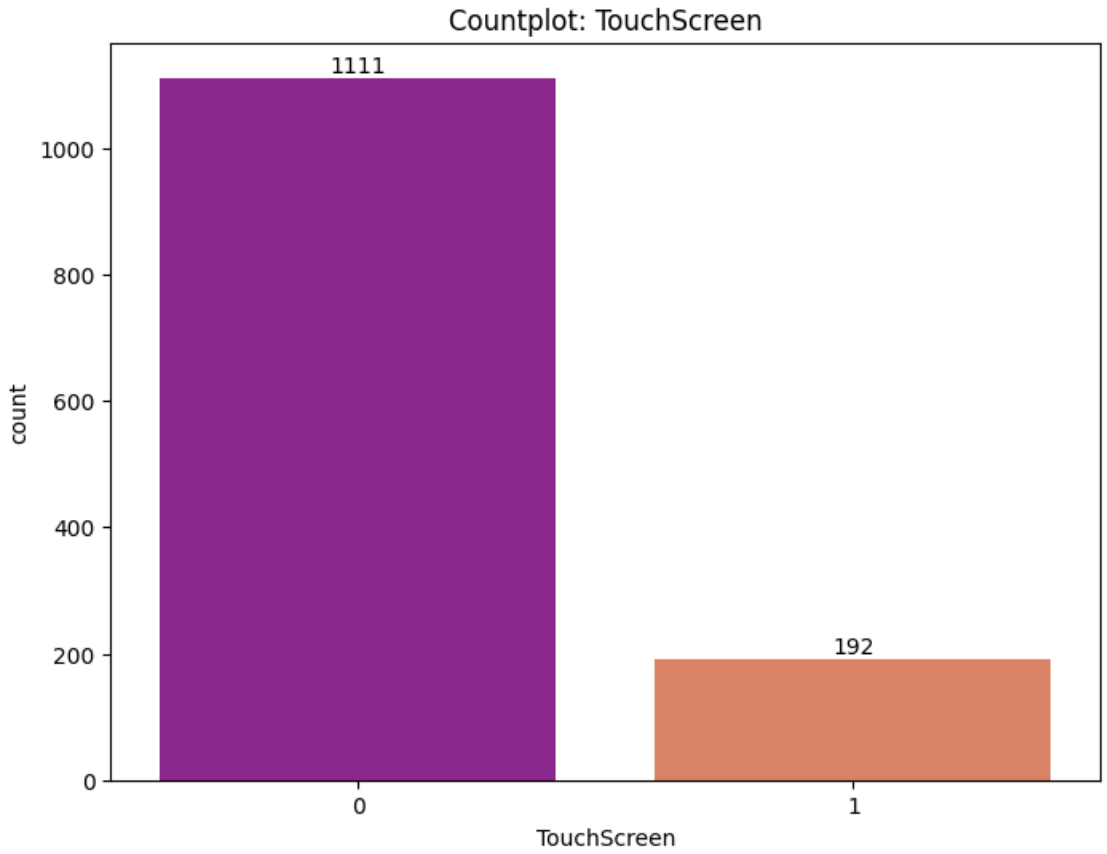
	Company	Product	TypeName	Inches	ScreenResolution	\
0	Apple	MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	
1	Apple	Macbook Air	Ultrabook	13.3	1440x900	
2	HP	250 G6	Notebook	15.6	Full HD 1920x1080	
3	Apple	MacBook Pro	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	
4	Apple	MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	

```
                          Cpu  Ram_GB               Memory  \
0        Intel Core i5 2.3GHz       8            128GB SSD
1        Intel Core i5 1.8GHz       8  128GB Flash Storage
2  Intel Core i5 7200U 2.5GHz       8            256GB SSD
3        Intel Core i7 2.7GHz      16            512GB SSD
4        Intel Core i5 3.1GHz       8            256GB SSD

                            Gpu  OpSys  Weight_KG  Price_euros  TouchScreen
0  Intel Iris Plus Graphics 640  macOS       1.37      1339.69            0
1        Intel HD Graphics 6000  macOS       1.34       898.94            0
2         Intel HD Graphics 620  No OS       1.86       575.00            0
3            AMD Radeon Pro 455  macOS       1.83      2537.45            0
4  Intel Iris Plus Graphics 650  macOS       1.37      1803.60            0
```

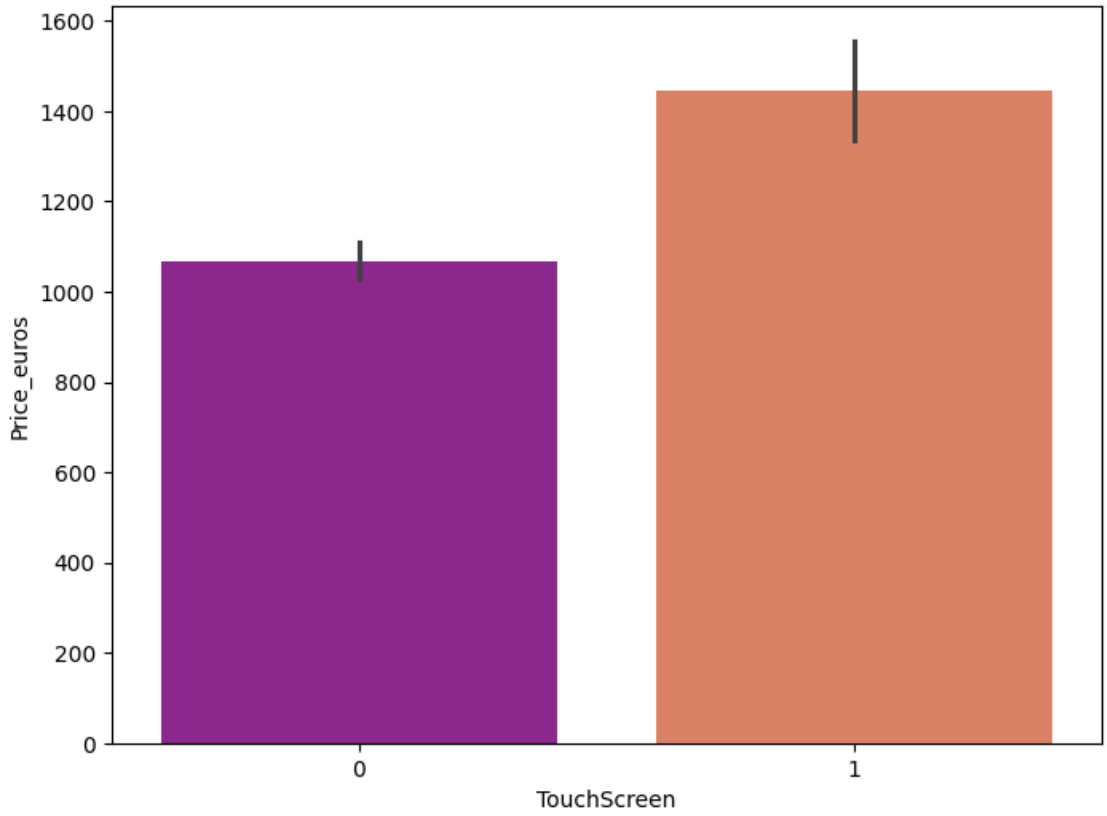
[32]:
```
df['TouchScreen'].value_counts()
```

[32]:
```
TouchScreen
0    1111
1     192
Name: count, dtype: int64
```

[27]:
```
plt.figure(figsize = (8,6))
ax = sns.countplot(data = df, x = df['TouchScreen'], palette = 'plasma')
plt.title('Countplot: TouchScreen')

for label in ax.containers:
    ax.bar_label(label)

plt.show()
```



[28]:
```
plt.figure(figsize = (8,6))
sns.barplot(data = df, x = df['TouchScreen'], y = df['Price_euros'],palette =␣
↪'plasma')
plt.show()
```



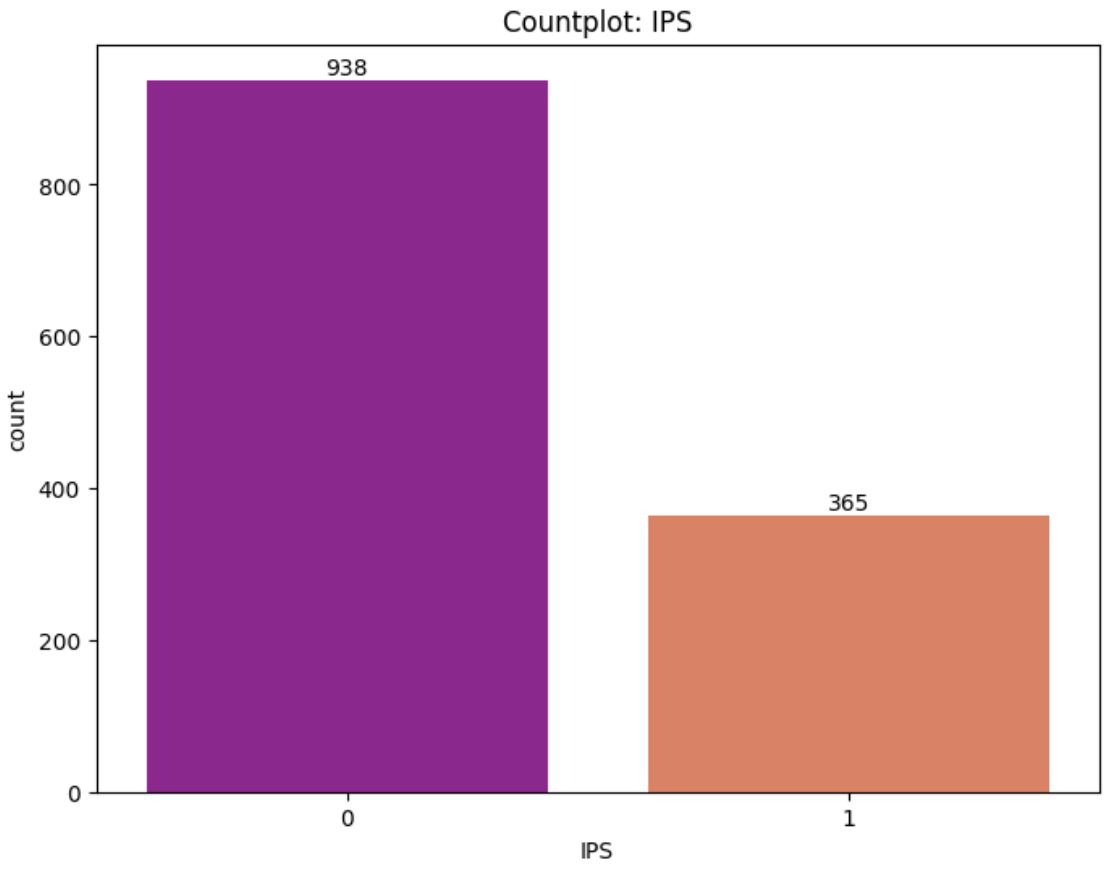
[29]:
```
# Create a feature 'IPS'

df['IPS'] = df['ScreenResolution'].apply(lambda element:1 if 'IPS' in element␣
 ↪else 0)

df.head()
```

[29]:

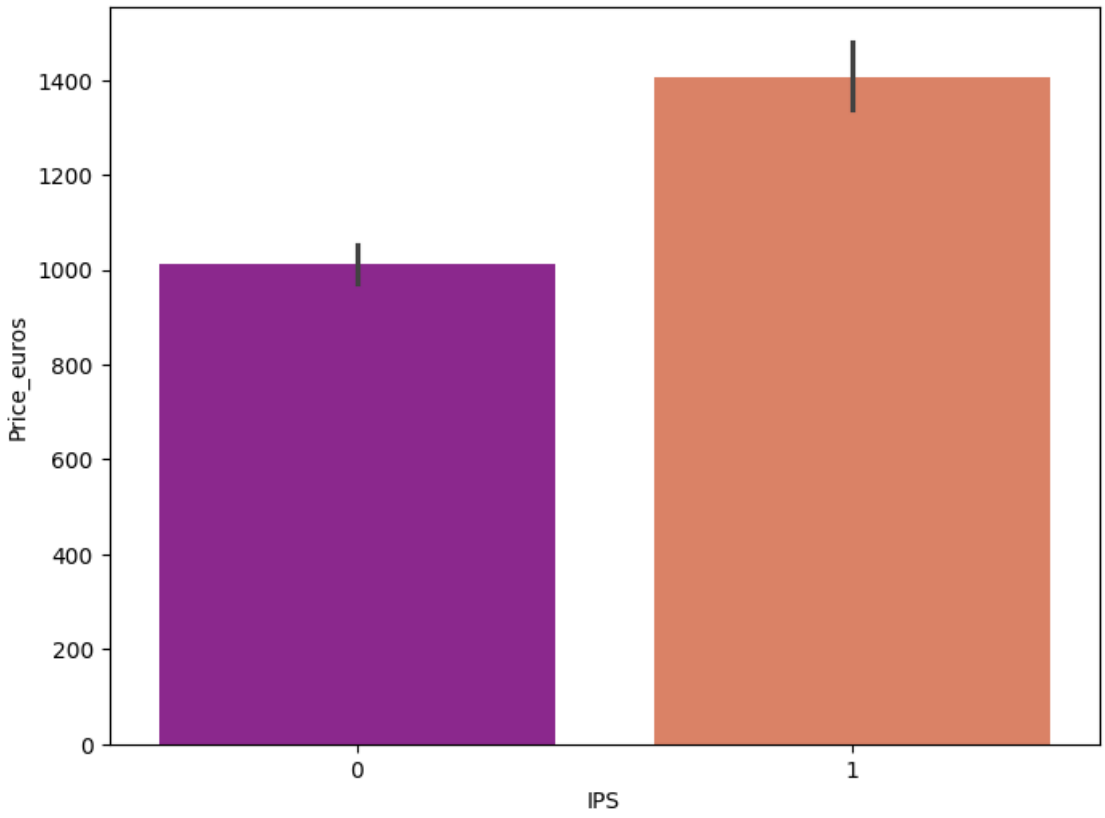
	Company	Product	TypeName	Inches	ScreenResolution	Cpu	Ram_GB	Memory
0	Apple	MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8	128GB SSD
1	Apple	Macbook Air	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8	128GB Flash Storage
2	HP	250 G6	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8	256GB SSD
3	Apple	MacBook Pro	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16	512GB SSD
4	Apple	MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8	256GB SSD

```
                           Gpu  OpSys  Weight_KG  Price_euros  TouchScreen  \
0  Intel Iris Plus Graphics 640  macOS       1.37      1339.69            0
1        Intel HD Graphics 6000  macOS       1.34       898.94            0
2         Intel HD Graphics 620  No OS       1.86       575.00            0
3            AMD Radeon Pro 455  macOS       1.83      2537.45            0
4  Intel Iris Plus Graphics 650  macOS       1.37      1803.60            0

   IPS
0    1
1    0
2    0
3    1
4    1
```

[30]:
```
plt.figure(figsize = (8,6))
ax = sns.countplot(data = df, x = df['IPS'], palette = 'plasma')
plt.title('Countplot: IPS')

for label in ax.containers:
    ax.bar_label(label)
plt.show()
```



[31]:
```
plt.figure(figsize = (8,6))
sns.barplot(data = df, x = df['IPS'], y = df['Price_euros'],palette = 'plasma')
plt.show()
```



[32]:
```
# Extracting the X Resolution and Y Resolutions

df['ScreenResolution'].str.split('x',n=1,expand=True)
```

[32]:

	0	1
0	IPS Panel Retina Display 2560	1600
1	1440	900
2	Full HD 1920	1080
3	IPS Panel Retina Display 2880	1800
4	IPS Panel Retina Display 2560	1600
…	…	…
1298	IPS Panel Full HD / Touchscreen 1920	1080
1299	IPS Panel Quad HD+ / Touchscreen 3200	1800
1300	1366	768

```
1301                                  1366   768
1302                                  1366   768

[1303 rows x 2 columns]
```

[33]:
```
df['X_res'] = df['ScreenResolution'].str.split('x',n=1,expand=True)[0]
df['Y_res'] = df['ScreenResolution'].str.split('x',n=1,expand=True)[1]
```

[34]:
```
df.head()
```

[34]:
```
  Company      Product   TypeName  Inches                    ScreenResolution  \
0   Apple  MacBook Pro  Ultrabook    13.3  IPS Panel Retina Display 2560x1600
1   Apple  Macbook Air  Ultrabook    13.3                              1440x900
2      HP       250 G6   Notebook    15.6                    Full HD 1920x1080
3   Apple  MacBook Pro  Ultrabook    15.4  IPS Panel Retina Display 2880x1800
4   Apple  MacBook Pro  Ultrabook    13.3  IPS Panel Retina Display 2560x1600

                          Cpu  Ram_GB               Memory  \
0        Intel Core i5 2.3GHz       8            128GB SSD
1        Intel Core i5 1.8GHz       8  128GB Flash Storage
2  Intel Core i5 7200U 2.5GHz       8            256GB SSD
3        Intel Core i7 2.7GHz      16            512GB SSD
4        Intel Core i5 3.1GHz       8            256GB SSD

                           Gpu  OpSys  Weight_KG  Price_euros  TouchScreen  \
0  Intel Iris Plus Graphics 640  macOS       1.37      1339.69            0
1        Intel HD Graphics 6000  macOS       1.34       898.94            0
2         Intel HD Graphics 620  No OS       1.86       575.00            0
3            AMD Radeon Pro 455  macOS       1.83      2537.45            0
4  Intel Iris Plus Graphics 650  macOS       1.37      1803.60            0

   IPS                         X_res Y_res
0    1  IPS Panel Retina Display 2560  1600
1    0                          1440   900
2    0                   Full HD 1920  1080
3    1  IPS Panel Retina Display 2880  1800
4    1  IPS Panel Retina Display 2560  1600
```

[35]:
```
df['X_res'] = df['X_res'].str.replace(',','').str.findall(r'(\d+\.?\d+)').
↪apply(lambda x : x[0])
```

[36]:
```
df.head()
```

[36]:
```
  Company      Product   TypeName  Inches                    ScreenResolution  \
0   Apple  MacBook Pro  Ultrabook    13.3  IPS Panel Retina Display 2560x1600
1   Apple  Macbook Air  Ultrabook    13.3                              1440x900
2      HP       250 G6   Notebook    15.6                    Full HD 1920x1080
```

```
3   Apple  MacBook Pro  Ultrabook    15.4  IPS Panel Retina Display 2880x1800
4   Apple  MacBook Pro  Ultrabook    13.3  IPS Panel Retina Display 2560x1600

                          Cpu  Ram_GB              Memory  \
0        Intel Core i5 2.3GHz       8           128GB SSD
1        Intel Core i5 1.8GHz       8 128GB Flash Storage
2  Intel Core i5 7200U 2.5GHz       8           256GB SSD
3        Intel Core i7 2.7GHz      16           512GB SSD
4        Intel Core i5 3.1GHz       8           256GB SSD

                            Gpu  OpSys  Weight_KG  Price_euros  TouchScreen  \
0  Intel Iris Plus Graphics 640  macOS       1.37      1339.69            0
1        Intel HD Graphics 6000  macOS       1.34       898.94            0
2         Intel HD Graphics 620  No OS       1.86       575.00            0
3            AMD Radeon Pro 455  macOS       1.83      2537.45            0
4  Intel Iris Plus Graphics 650  macOS       1.37      1803.60            0

   IPS X_res Y_res
0    1  2560  1600
1    0  1440   900
2    0  1920  1080
3    1  2880  1800
4    1  2560  1600
```

[37]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1303 entries, 0 to 1302
Data columns (total 16 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Company           1303 non-null   object
 1   Product           1303 non-null   object
 2   TypeName          1303 non-null   object
 3   Inches            1303 non-null   float64
 4   ScreenResolution  1303 non-null   object
 5   Cpu               1303 non-null   object
 6   Ram_GB            1303 non-null   int32
 7   Memory            1303 non-null   object
 8   Gpu               1303 non-null   object
 9   OpSys             1303 non-null   object
 10  Weight_KG         1303 non-null   float32
 11  Price_euros       1303 non-null   float64
 12  TouchScreen       1303 non-null   int64
 13  IPS               1303 non-null   int64
 14  X_res             1303 non-null   object
 15  Y_res             1303 non-null   object
```

```
dtypes: float32(1), float64(2), int32(1), int64(2), object(10)
memory usage: 152.8+ KB
```

- from above it is clear that 'X_res' and 'Y_res' are categorical so needs to change in int.

[38]:
```
df['X_res'] = df['X_res'].astype('int')
df['Y_res'] = df['Y_res'].astype('int')
```

[39]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1303 entries, 0 to 1302
Data columns (total 16 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Company           1303 non-null   object
 1   Product           1303 non-null   object
 2   TypeName          1303 non-null   object
 3   Inches            1303 non-null   float64
 4   ScreenResolution  1303 non-null   object
 5   Cpu               1303 non-null   object
 6   Ram_GB            1303 non-null   int32
 7   Memory            1303 non-null   object
 8   Gpu               1303 non-null   object
 9   OpSys             1303 non-null   object
 10  Weight_KG         1303 non-null   float32
 11  Price_euros       1303 non-null   float64
 12  TouchScreen       1303 non-null   int64
 13  IPS               1303 non-null   int64
 14  X_res             1303 non-null   int64
 15  Y_res             1303 non-null   int64
dtypes: float32(1), float64(2), int32(1), int64(4), object(8)
memory usage: 152.8+ KB
```

[40]:
```
df['Product']
```

[40]:
```
0                                  MacBook Pro
1                                  Macbook Air
2                                      250 G6
3                                  MacBook Pro
4                                  MacBook Pro
                          …
1298                            Yoga 500-14ISK
1299                            Yoga 900-13ISK
1300                         IdeaPad 100S-14IBR
1301    15-AC110nv (i7-6500U/6GB/1TB/Radeon
1302    X553SA-XX031T (N3050/4GB/500GB/W10)
Name: Product, Length: 1303, dtype: object
```

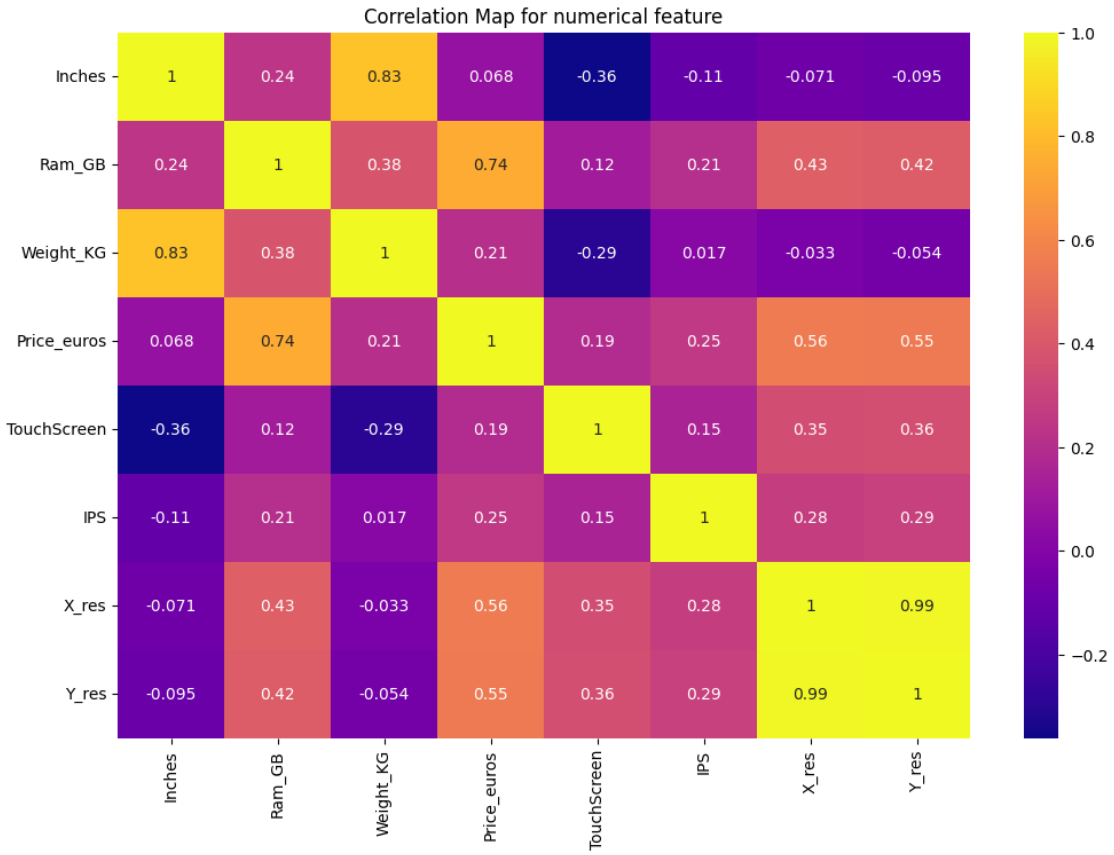
[41]:
```
df['Product'].value_counts()
```

[41]:
```
Product
XPS 13                                      30
Inspiron 3567                               29
250 G6                                      21
Legion Y520-15IKBN                          19
Vostro 3568                                 19
                                            ..
15-bw007nv (A10-9620P/6GB/128GB/Radeon       1
SmartBook 130                                1
A541NA-GO342 (N3350/4GB/500GB/Linux)         1
17-X047na (i3-6006U/8GB/1TB/W10)             1
V330-15IKB (i5-8250U/4GB/500GB/FHD/W10)      1
Name: count, Length: 618, dtype: int64
```

- As there are 618 unique values in 'Product' in 1303 total observations, so will directly drop as it will not much affect the price.

[42]:
```
# Droping feature 'Product'

df.drop('Product', axis=1,inplace=True)
```

[43]:
```
# Correlation

df_num = df[[feature for feature in df.columns if df[feature].dtypes !=␣
 ↪'object']]

plt.figure(figsize = (12,8))
sns.heatmap(df_num.corr(), annot=True,cmap='plasma')
plt.title('Correlation Map for numerical feature')
plt.xticks(rotation='vertical')
plt.show()
```



[44]:
```
df_num.corr()['Price_euros']
```

[44]:
```
Inches          0.068197
Ram_GB          0.743007
Weight_KG       0.210370
Price_euros     1.000000
TouchScreen     0.191226
IPS             0.252208
X_res           0.556529
Y_res           0.552809
Name: Price_euros, dtype: float64
```

- From above observations, it is clear that 'X_res' and 'Y_res' both are highly corelated and can affect the decision
- So I have created a new column named 'PPI(Pixel Per Inch). –> this is done by using following mathematical relation.

[45]:
```
df['PPI'] = (np.round((df['X_res']**2 + df['Y_res']**2)**(1/2))/df['Inches']).
↪astype('float')
```

[46]:
```
df.head()
```

[46]:

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram_GB	Memory	Gpu	OpSys	Weight_KG	Price_euros	TouchScreen	IPS	X_res	Y_res	PPI
0	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37	1339.69	0	1	2560	1600	226.992481
1	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34	898.94	0	0	1440	900	127.669173
2	HP	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	No OS	1.86	575.00	0	0	1920	1080	141.217949
3	Apple	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16	512GB SSD	AMD Radeon Pro 455	macOS	1.83	2537.45	0	1	2880	1800	220.519481
4	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37	1803.60	0	1	2560	1600	226.992481

[47]:
```
df[['PPI','Price_euros']].corr()
```

[47]:

	PPI	Price_euros
PPI	1.000000	0.473497
Price_euros	0.473497	1.000000

[48]:
```
#Now we can delete 'Inches', 'X_res' and 'Y_res'

df.drop(columns = ['Inches','ScreenResolution', 'X_res','Y_res'],␣
 ↪axis=1,inplace=True)
df.head()
```

[48]:

	Company	TypeName	Cpu	Ram_GB	Memory	Gpu	OpSys	Weight_KG	Price_euros	TouchScreen	IPS	X_res	Y_res	PPI
0	Apple	Ultrabook	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37	1339.69	0	1	2560	1600	226.992481
1	Apple	Ultrabook	Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34	898.94	0	0	1440	900	127.669173
2	HP	Notebook	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	No OS	1.86	575.00	0	0	1920	1080	141.217949
3	Apple	Ultrabook	Intel Core i7 2.7GHz	16	512GB SSD	AMD Radeon Pro 455	macOS	1.83	2537.45	0	1	2880	1800	220.519481

```
4   Apple  Ultrabook        Intel Core i5 3.1GHz       8            256GB SSD

                            Gpu  OpSys  Weight_KG  Price_euros  TouchScreen  \
0  Intel Iris Plus Graphics 640  macOS       1.37      1339.69            0
1        Intel HD Graphics 6000  macOS       1.34       898.94            0
2         Intel HD Graphics 620  No OS       1.86       575.00            0
3            AMD Radeon Pro 455  macOS       1.83      2537.45            0
4  Intel Iris Plus Graphics 650  macOS       1.37      1803.60            0

   IPS         PPI
0    1  226.992481
1    0  127.669173
2    0  141.217949
3    1  220.519481
4    1  226.992481
```

[49]:
```
df['Cpu'].value_counts()
```

[49]:
```
Cpu
Intel Core i5 7200U 2.5GHz       190
Intel Core i7 7700HQ 2.8GHz      146
Intel Core i7 7500U 2.7GHz       134
Intel Core i7 8550U 1.8GHz        73
Intel Core i5 8250U 1.6GHz        72
                                 …
Intel Core M M3-6Y30 0.9GHz        1
AMD A9-Series 9420 2.9GHz          1
Intel Core i3 6006U 2.2GHz         1
AMD A6-Series 7310 2GHz            1
Intel Xeon E3-1535M v6 3.1GHz      1
Name: count, Length: 118, dtype: int64
```

[50]:
```
df['Cpu_name'] = df['Cpu'].apply(lambda text:" ".join(text.split()[:3]))
df.head()
```

[50]:
```
  Company   TypeName                         Cpu  Ram_GB               Memory  \
0   Apple  Ultrabook       Intel Core i5 2.3GHz       8            128GB SSD
1   Apple  Ultrabook       Intel Core i5 1.8GHz       8  128GB Flash Storage
2      HP   Notebook  Intel Core i5 7200U 2.5GHz       8            256GB SSD
3   Apple  Ultrabook       Intel Core i7 2.7GHz      16            512GB SSD
4   Apple  Ultrabook       Intel Core i5 3.1GHz       8            256GB SSD

                            Gpu  OpSys  Weight_KG  Price_euros  TouchScreen  \
0  Intel Iris Plus Graphics 640  macOS       1.37      1339.69            0
1        Intel HD Graphics 6000  macOS       1.34       898.94            0
2         Intel HD Graphics 620  No OS       1.86       575.00            0
3            AMD Radeon Pro 455  macOS       1.83      2537.45            0
```

```
4  Intel Iris Plus Graphics 650  macOS       1.37      1803.60            0

   IPS         PPI       Cpu_name
0    1  226.992481  Intel Core i5
1    0  127.669173  Intel Core i5
2    0  141.217949  Intel Core i5
3    1  220.519481  Intel Core i7
4    1  226.992481  Intel Core i5
```

[51]:
```
df['Cpu_name'].value_counts()
```

[51]:
```
Cpu_name
Intel Core i7                527
Intel Core i5                423
Intel Core i3                136
Intel Celeron Dual            80
Intel Pentium Quad            27
Intel Core M                  19
AMD A9-Series 9420            12
Intel Celeron Quad             8
AMD A6-Series 9220             8
AMD A12-Series 9720P           7
Intel Atom x5-Z8350            5
AMD A8-Series 7410             4
Intel Atom x5-Z8550            4
Intel Pentium Dual             3
AMD A9-Series 9410             3
AMD Ryzen 1700                 3
AMD A9-Series A9-9420          2
AMD A10-Series 9620P           2
Intel Atom X5-Z8350            2
AMD E-Series E2-9000e          2
Intel Xeon E3-1535M            2
Intel Xeon E3-1505M            2
AMD E-Series 7110              2
AMD A10-Series 9600P           2
AMD A6-Series A6-9220          2
AMD A10-Series A10-9620P       2
AMD Ryzen 1600                 1
Intel Atom x5-Z8300            1
AMD E-Series E2-6110           1
AMD FX 9830P                   1
AMD E-Series E2-9000           1
AMD A6-Series 7310             1
Intel Atom Z8350               1
AMD A12-Series 9700P           1
AMD A4-Series 7210             1
```

```
AMD FX 8800P                    1
AMD E-Series 9000e              1
Samsung Cortex A72&A53          1
AMD E-Series 9000               1
AMD E-Series 6110               1
Name: count, dtype: int64
```

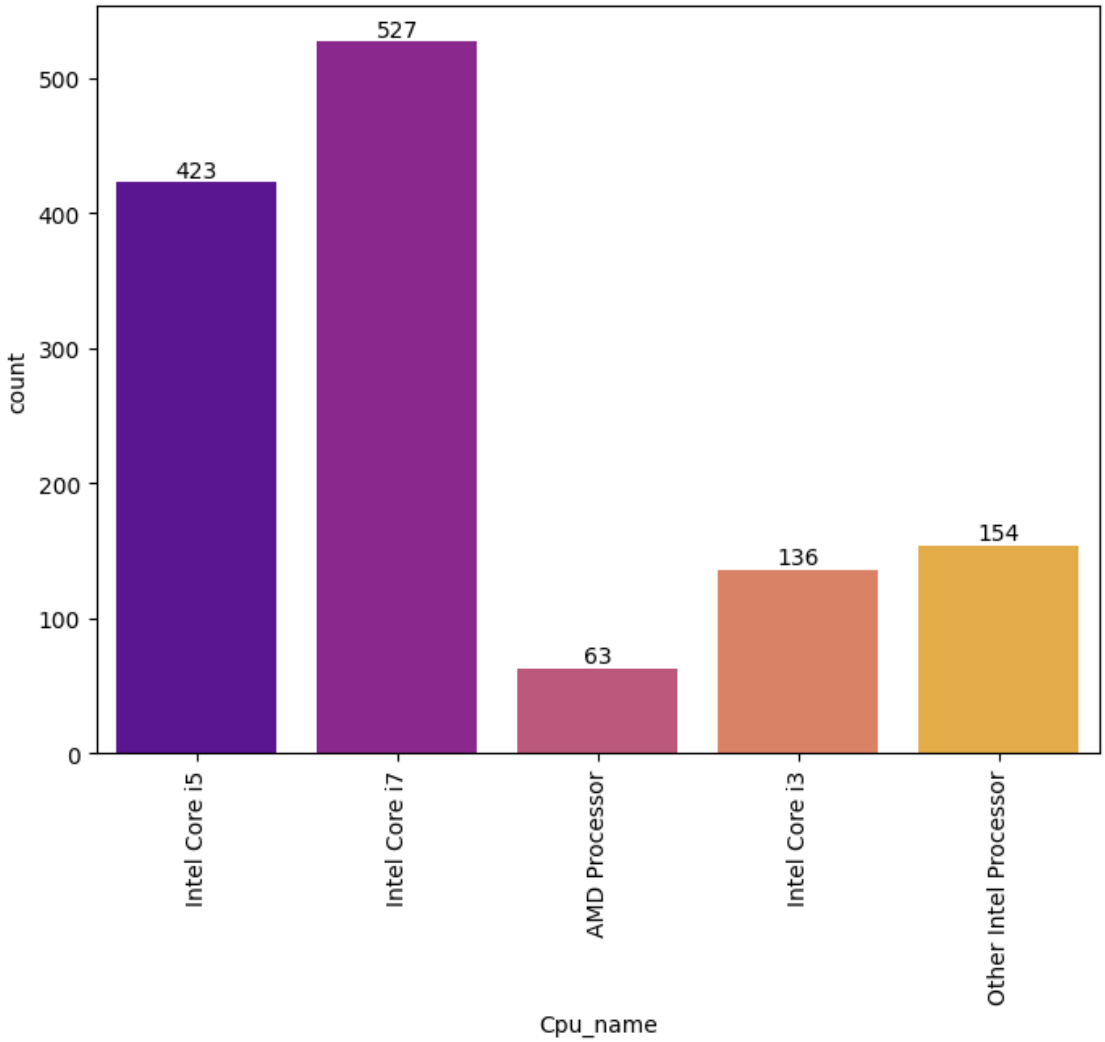
[52]:
```
def processortype(text):

    if text == 'Intel Core i5' or text == 'Intel Core i7' or text == 'Intel␣
 ↪Core i3':
        return text

    else:
        if text.split()[0] == 'Intel':
            return 'Other Intel Processor'
        else:
            return 'AMD Processor'


df['Cpu_name'] = df['Cpu_name'].apply(lambda text:processortype(text))
df.head()
```

[52]:

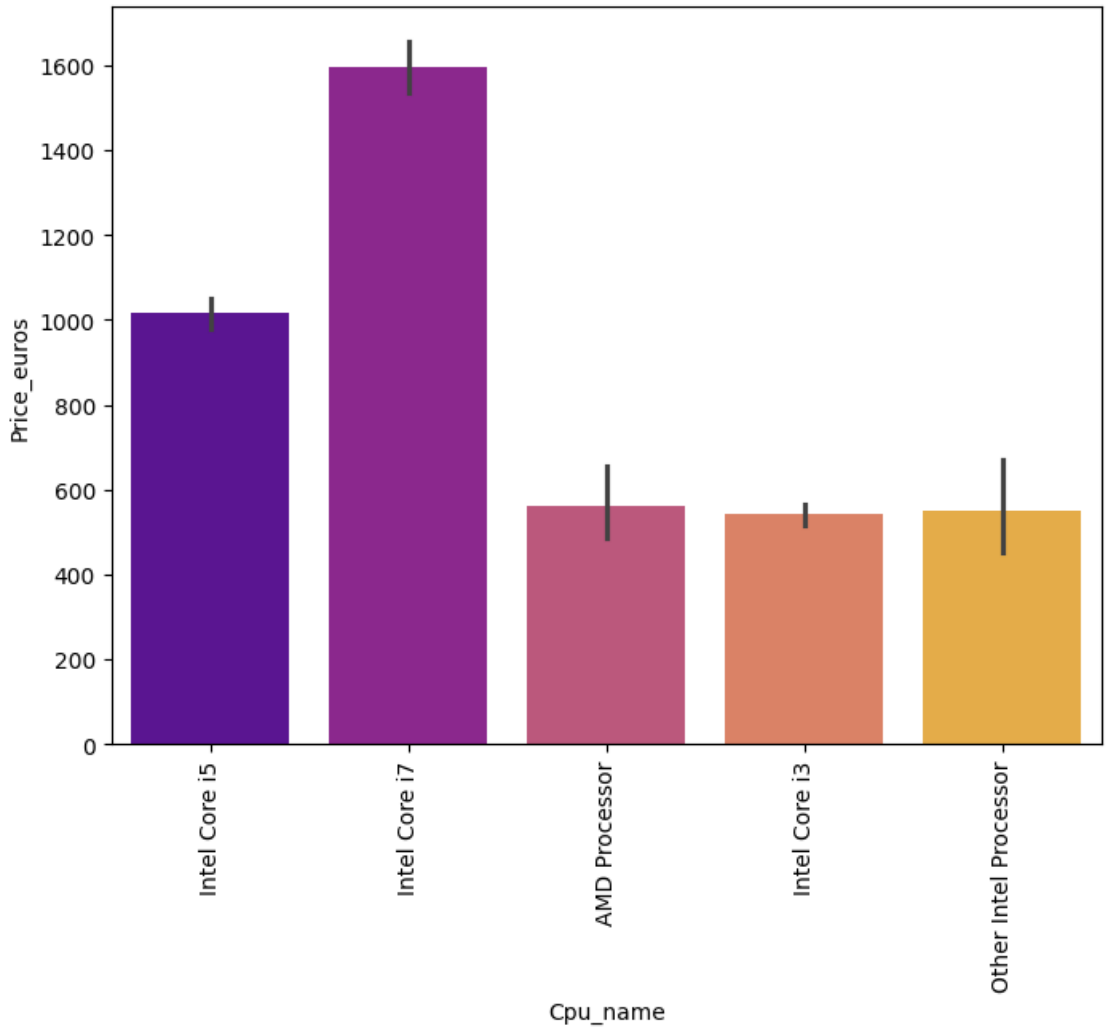
	Company	TypeName	Cpu	Ram_GB	Memory	Gpu	OpSys	Weight_KG	Price_euros	TouchScreen	IPS	PPI	Cpu_name
0	Apple	Ultrabook	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37	1339.69	0	1	226.992481	Intel Core i5
1	Apple	Ultrabook	Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34	898.94	0	0	127.669173	Intel Core i5
2	HP	Notebook	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	No OS	1.86	575.00	0	0	141.217949	Intel Core i5
3	Apple	Ultrabook	Intel Core i7 2.7GHz	16	512GB SSD	AMD Radeon Pro 455	macOS	1.83	2537.45	0	1	220.519481	Intel Core i7
4	Apple	Ultrabook	Intel Core i5 3.1GHz	8	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37	1803.60	0	1	226.992481	Intel Core i5

[53]:
```
plt.figure(figsize = (8,6))
ax = sns.countplot(data = df, x = df['Cpu_name'], palette='plasma')
plt.xticks(rotation = 'vertical')
```

```
for label in ax.containers:
    ax.bar_label(label)
plt.show()
```



[54]:
```
plt.figure(figsize = (8,6))
sns.barplot(data = df, x = df['Cpu_name'], y = df['Price_euros'],␣
 ↪palette='plasma')
plt.xticks(rotation = 'vertical')
plt.show()
```



[55]:
```
# Now we will drop the Cpu column

df.drop(columns = ['Cpu'], axis=1,inplace=True)
```

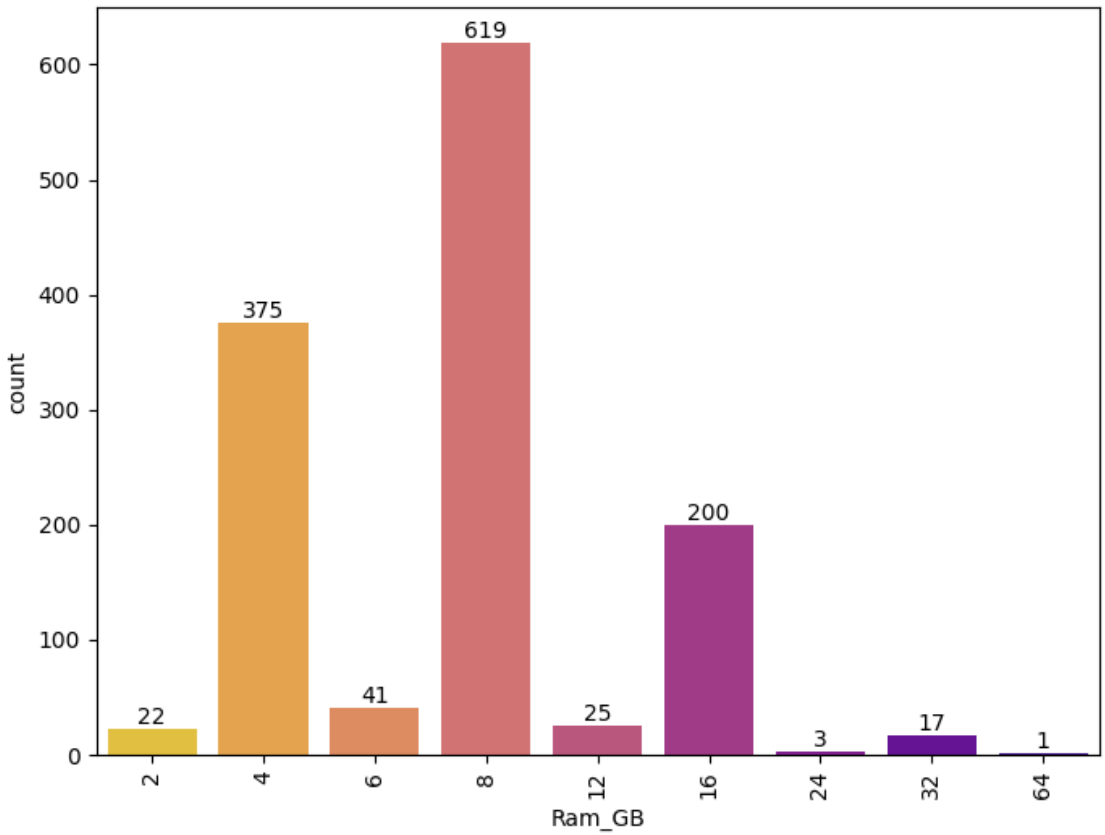
[56]:
```
df.head()
```

[56]:

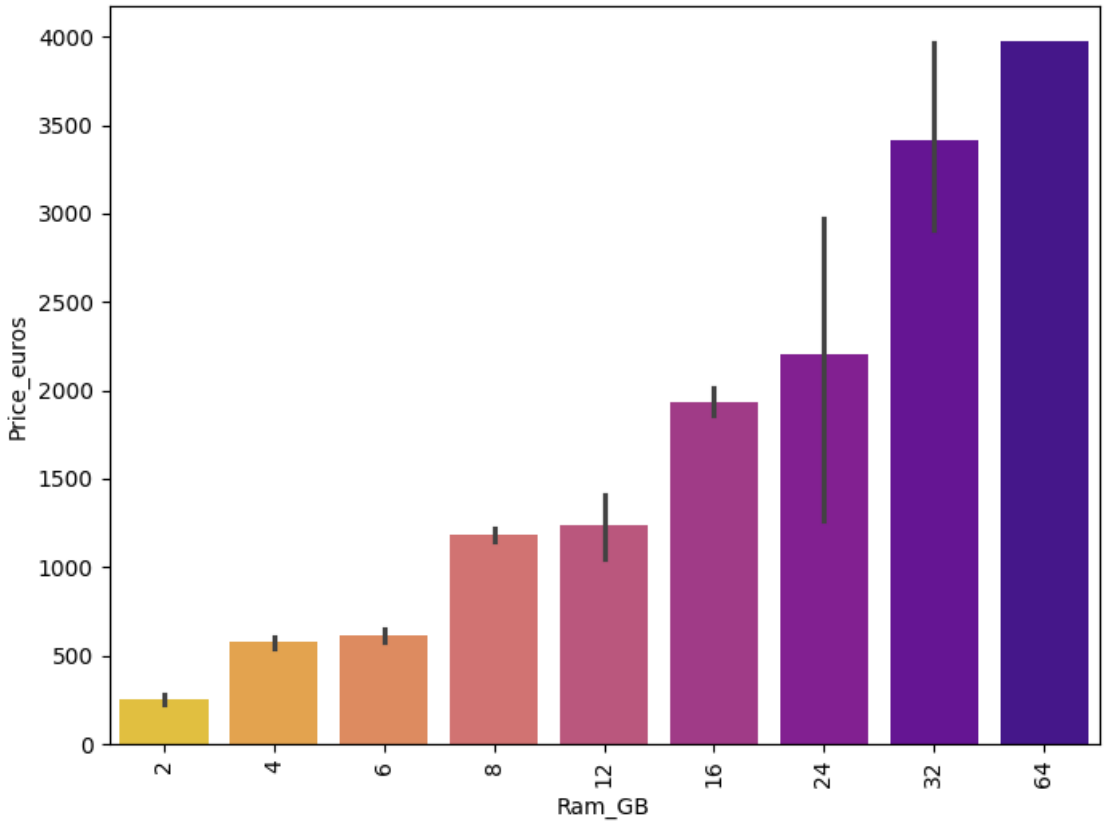
	Company	TypeName	Ram_GB	Memory	Gpu	OpSys	Weight_KG	Price_euros	TouchScreen
0	Apple	Ultrabook	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37	1339.69	0
1	Apple	Ultrabook	8	128GB Flash Storage					
2	HP	Notebook	8	256GB SSD					
3	Apple	Ultrabook	16	512GB SSD					
4	Apple	Ultrabook	8	256GB SSD					

```
1        Intel HD Graphics 6000  macOS       1.34       898.94            0
2         Intel HD Graphics 620  No OS       1.86       575.00            0
3            AMD Radeon Pro 455  macOS       1.83      2537.45            0
4  Intel Iris Plus Graphics 650  macOS       1.37      1803.60            0

   IPS         PPI       Cpu_name
0    1  226.992481  Intel Core i5
1    0  127.669173  Intel Core i5
2    0  141.217949  Intel Core i5
3    1  220.519481  Intel Core i7
4    1  226.992481  Intel Core i5
```

[57]:
```
# Countplot of Ram

plt.figure(figsize = (8,6))
ax = sns.countplot(data = df, x = df['Ram_GB'], palette='plasma_r')
plt.xticks(rotation = 'vertical')

for label in ax.containers:
    ax.bar_label(label)
plt.show()
```



[58]:
```
# Price variations w.r.t RAM

plt.figure(figsize = (8,6))
sns.barplot(data = df, x = df['Ram_GB'], y = df['Price_euros'],␣
 ↪palette='plasma_r')
plt.xticks(rotation = 'vertical')
plt.show()
```



[59]:
```
# 'Memory' Column

df['Memory'].value_counts()
```

[59]:
```
Memory
256GB SSD                        412
1TB HDD                          223
500GB HDD                        132
512GB SSD                        118
128GB SSD +  1TB HDD              94
128GB SSD                         76
```

```
256GB SSD +  1TB HDD             73
32GB Flash Storage               38
2TB HDD                          16
64GB Flash Storage               15
512GB SSD +  1TB HDD             14
1TB SSD                          14
256GB SSD +  2TB HDD             10
1.0TB Hybrid                      9
256GB Flash Storage               8
16GB Flash Storage                7
32GB SSD                          6
180GB SSD                         5
128GB Flash Storage               4
512GB SSD +  2TB HDD              3
16GB SSD                          3
512GB Flash Storage               2
1TB SSD +  1TB HDD                2
256GB SSD +  500GB HDD            2
128GB SSD +  2TB HDD              2
256GB SSD +  256GB SSD            2
512GB SSD +  256GB SSD            1
512GB SSD +  512GB SSD            1
64GB Flash Storage +  1TB HDD     1
1TB HDD +  1TB HDD                1
32GB HDD                          1
64GB SSD                          1
128GB HDD                         1
240GB SSD                         1
8GB SSD                           1
508GB Hybrid                      1
1.0TB HDD                         1
512GB SSD +  1.0TB Hybrid         1
256GB SSD +  1.0TB Hybrid         1
Name: count, dtype: int64
```

[60]:
```python
# Preocess 'Memory' column

# Remove the decimal for example 1.0 = 1
df['Memory'] = df['Memory'].astype(str).replace('\.0','',regex=True)

# Remove GB
df['Memory'] = df['Memory'].str.replace('GB','')

# Removve TB with 000
df['Memory'] = df['Memory'].str.replace('TB','000')
```

[61]:
```python
df.Memory
```

[61]:
```
0                  128 SSD
1       128 Flash Storage
2                  256 SSD
3                  512 SSD
4                  256 SSD
               …
1298               128 SSD
1299               512 SSD
1300     64 Flash Storage
1301             1000 HDD
1302              500 HDD
Name: Memory, Length: 1303, dtype: object
```

[62]:
```
# split the text across '+'
newdf = df['Memory'].str.split('+',n=1,expand=True)
newdf
```

[62]:

	0	1
0	128 SSD	None
1	128 Flash Storage	None
2	256 SSD	None
3	512 SSD	None
4	256 SSD	None
…	…	…
1298	128 SSD	None
1299	512 SSD	None
1300	64 Flash Storage	None
1301	1000 HDD	None
1302	500 HDD	None

[1303 rows x 2 columns]

[63]:
```
df['Memory_first'] = newdf[0]
df['Memory_first'] = df['Memory_first'].str.strip()
df.head()
```

[63]:

	Company	TypeName	Ram_GB	Memory	Gpu	OpSys	Weight_KG	Price_euros	TouchScreen	IPS	PPI	Cpu_name
0	Apple	Ultrabook	8	128 SSD	Intel Iris Plus Graphics 640	macOS	1.37	1339.69	0	1	226.992481	Intel Core i5
1	Apple	Ultrabook	8	128 Flash Storage	Intel HD Graphics 6000	macOS	1.34	898.94	0	0	127.669173	Intel Core i5
2	HP	Notebook	8	256 SSD	Intel HD Graphics 620	No OS	1.86	575.00	0	0	141.217949	Intel Core i5

```
3  macOS       1.83      2537.45             0    1  220.519481  Intel Core i7
4  macOS       1.37      1803.60             0    1  226.992481  Intel Core i5

        Memory_first
0            128 SSD
1  128 Flash Storage
2            256 SSD
3            512 SSD
4            256 SSD
```

[64]:
```
def applychanges(value):

    df['Memory_first-'+value] = df['Memory_first'].apply(lambda x:1 if value in␣
 ↪x else 0)

valueList = ['SSD', 'HDD', 'Hybrid', 'Flash Storage']
for value in valueList:
    applychanges(value)

df.sample(10)
```

[64]:
```
      Company   TypeName  Ram_GB    Memory                         Gpu  \
999        HP  Ultrabook       8   512 SSD         Intel HD Graphics 620
227      Asus   Notebook       8   256 SSD         Nvidia GeForce 920MX
376    Lenovo    Netbook       4   128 SSD         Intel HD Graphics 500
634      Asus   Notebook       8  1000 HDD         Intel HD Graphics 500
984   Toshiba   Notebook       4   500 HDD         Intel HD Graphics 520
1301       HP   Notebook       6  1000 HDD            AMD Radeon R5 M330
952   Toshiba   Notebook       8   256 SSD         Intel HD Graphics 520
1279   Lenovo   Notebook       8  1000 HDD       Nvidia GeForce GTX 960M
864      Dell  Ultrabook      16   512 SSD  Intel Iris Plus Graphics 640
1028     Dell  Ultrabook       8   256 SSD         Intel HD Graphics 620

           OpSys  Weight_KG  Price_euros  TouchScreen  IPS         PPI  \
999   Windows 10       1.48       2089.0            0    0  157.357143
227   Windows 10       2.00        769.0            0    0  141.217949
376   Windows 10       1.59        553.0            0    1  135.086207
634   Windows 10       2.00        399.0            0    0  100.448718
984   Windows 10       1.75        915.0            0    0  111.928571
1301  Windows 10       2.19        764.0            0    0  100.448718
952    Windows 7       1.20       1460.0            0    0  165.639098
1279  Windows 10       2.60        899.0            0    1  141.217949
864   Windows 10       1.29       2240.0            1    0  276.090226
1028  Windows 10       1.29       1624.0            1    0  276.090226

                  Cpu_name Memory_first  Memory_first-SSD  Memory_first-HDD  \
999           Intel Core i7      512 SSD                 1                 0
```

```
227           Intel Core i5      256 SSD                1                   0
376   Other Intel Processor      128 SSD                1                   0
634   Other Intel Processor     1000 HDD                0                   1
984           Intel Core i5      500 HDD                0                   1
1301          Intel Core i7     1000 HDD                0                   1
952           Intel Core i5      256 SSD                1                   0
1279          Intel Core i7     1000 HDD                0                   1
864           Intel Core i7      512 SSD                1                   0
1028          Intel Core i5      256 SSD                1                   0

      Memory_first-Hybrid  Memory_first-Flash Storage
999                     0                          0
227                     0                          0
376                     0                          0
634                     0                          0
984                     0                          0
1301                    0                          0
952                     0                          0
1279                    0                          0
864                     0                          0
1028                    0                          0
```

[65]:
```
# Remove all the characters just keep the numbers
df['Memory_first'] = df['Memory_first'].str.replace(r'\D','',regex=True)
df['Memory_first'].value_counts()
```

[65]:
```
Memory_first
256     508
1000    250
128     177
512     140
500     132
32       45
64       17
2000     16
16       10
180       5
240       1
8         1
508       1
Name: count, dtype: int64
```

[66]:
```
df['Memory_second'] = newdf[1]
df.head()
```

[66]:
```
   Company   TypeName  Ram_GB             Memory                          Gpu  \
0   Apple  Ultrabook       8          128 SSD  Intel Iris Plus Graphics 640
```

```
1   Apple  Ultrabook       8  128 Flash Storage        Intel HD Graphics 6000
2      HP   Notebook       8            256 SSD         Intel HD Graphics 620
3   Apple  Ultrabook      16            512 SSD            AMD Radeon Pro 455
4   Apple  Ultrabook       8            256 SSD  Intel Iris Plus Graphics 650

   OpSys  Weight_KG  Price_euros  TouchScreen  IPS         PPI       Cpu_name  \
0  macOS       1.37      1339.69            0    1  226.992481  Intel Core i5
1  macOS       1.34       898.94            0    0  127.669173  Intel Core i5
2  No OS       1.86       575.00            0    0  141.217949  Intel Core i5
3  macOS       1.83      2537.45            0    1  220.519481  Intel Core i7
4  macOS       1.37      1803.60            0    1  226.992481  Intel Core i5

  Memory_first  Memory_first-SSD  Memory_first-HDD  Memory_first-Hybrid  \
0          128                 1                 0                    0
1          128                 0                 0                    0
2          256                 1                 0                    0
3          512                 1                 0                    0
4          256                 1                 0                    0

   Memory_first-Flash Storage Memory_second
0                           0          None
1                           1          None
2                           0          None
3                           0          None
4                           0          None
```

[67]:
```
df['Memory_second'].isnull().sum()
```

[67]:
```
1095
```

[68]:
```
df['Memory_second'] = df['Memory_second'].fillna("0")
df['Memory_second'].value_counts()
```

[68]:
```
Memory_second
0               1095
  1000 HDD       185
  2000 HDD        15
  256 SSD          3
  500 HDD          2
  1000 Hybrid      2
  512 SSD          1
Name: count, dtype: int64
```

[69]:
```
def applychanges2(value):

    df['Memory_second-'+value] = df['Memory_second'].apply(lambda x:1 if value␣
 ↪in x else 0)
```

```
valueList = ['SSD', 'HDD', 'Hybrid', 'Flash Storage']

for value in valueList:
    applychanges2(value)
```

[70]:
```
# Remove all the characters just keep the numbers
df['Memory_second'] = df['Memory_second'].str.replace(r'\D','',regex=True)
df['Memory_second'].value_counts()
```

[70]:
```
Memory_second
0       1095
1000     187
2000      15
256        3
500        2
512        1
Name: count, dtype: int64
```

[71]:
```
df['Memory_first'] = df['Memory_first'].astype('int')
df['Memory_second'] = df['Memory_second'].astype('int')
```

[72]:
```
df.sample(5)
```

[72]:
```
     Company   TypeName  Ram_GB    Memory                        Gpu  \
813     Dell   Notebook       8  1000 HDD  Nvidia GeForce GT 940MX
641       HP   Notebook       8  1000 HDD            AMD Radeon R5
1093    Dell  Ultrabook       4   256 SSD           AMD Radeon 530
24        HP  Ultrabook       8   256 SSD    Intel HD Graphics 620
318       HP   Notebook       8   256 SSD    Intel HD Graphics 620

           OpSys  Weight_KG  Price_euros  TouchScreen  IPS  …  Memory_first  \
813   Windows 10       1.98       961.00            0    0  …          1000
641   Windows 10       2.60       520.90            0    0  …          1000
1093  Windows 10       1.40       818.35            0    1  …           256
24    Windows 10       1.91       659.00            0    0  …           256
318   Windows 10       1.95       980.00            0    0  …           256

     Memory_first-SSD  Memory_first-HDD  Memory_first-Hybrid  \
813                 0                 1                    0
641                 0                 1                    0
1093                1                 0                    0
24                  1                 0                    0
318                 1                 0                    0

      Memory_first-Flash Storage  Memory_second  Memory_second-SSD  \
813                            0              0                  0
```

```
641                              0                0                    0
1093                             0                0                    0
24                               0                0                    0
318                              0                0                    0

      Memory_second-HDD  Memory_second-Hybrid  Memory_second-Flash Storage
813                   0                     0                            0
641                   0                     0                            0
1093                  0                     0                            0
24                    0                     0                            0
318                   0                     0                            0

[5 rows x 22 columns]
```

[73]:
```
# Multiplying the column and storing the result in sunsequent column

df['HDD'] =␣
 ↪(df['Memory_first']*df['Memory_first-HDD']+df['Memory_second']*df['Memory_second-HDD'])
df['SSD'] =␣
 ↪(df['Memory_first']*df['Memory_first-SSD']+df['Memory_second']*df['Memory_second-SSD'])
df['Hybrid'] =␣
 ↪(df['Memory_first']*df['Memory_first-Hybrid']+df['Memory_second']*df['Memory_second-Hybrid']
df['Flash Storage'] = (df['Memory_first']*df['Memory_first-Flash␣
 ↪Storage']+df['Memory_second']*df['Memory_second-Flash Storage'])
```

[74]:
```
df.head()
```

[74]:
```
   Company   TypeName  Ram_GB              Memory                           Gpu  \
0    Apple  Ultrabook       8           128 SSD  Intel Iris Plus Graphics 640
1    Apple  Ultrabook       8  128 Flash Storage        Intel HD Graphics 6000
2       HP   Notebook       8           256 SSD         Intel HD Graphics 620
3    Apple  Ultrabook      16           512 SSD            AMD Radeon Pro 455
4    Apple  Ultrabook       8           256 SSD  Intel Iris Plus Graphics 650

    OpSys  Weight_KG  Price_euros  TouchScreen  IPS  …  \
0  macOS       1.37      1339.69            0    1  …
1  macOS       1.34       898.94            0    0  …
2  No OS       1.86       575.00            0    0  …
3  macOS       1.83      2537.45            0    1  …
4  macOS       1.37      1803.60            0    1  …

   Memory_first-Flash Storage Memory_second  Memory_second-SSD  \
0                           0             0                  0
1                           1             0                  0
2                           0             0                  0
3                           0             0                  0
4                           0             0                  0
```

```
   Memory_second-HDD  Memory_second-Hybrid  Memory_second-Flash Storage  HDD  \
0                  0                     0                            0    0
1                  0                     0                            0    0
2                  0                     0                            0    0
3                  0                     0                            0    0
4                  0                     0                            0    0

   SSD  Hybrid  Flash Storage
0  128       0              0
1    0       0            128
2  256       0              0
3  512       0              0
4  256       0              0

[5 rows x 26 columns]
```

[75]:
```
df.columns
```

[75]:
```
Index(['Company', 'TypeName', 'Ram_GB', 'Memory', 'Gpu', 'OpSys', 'Weight_KG',
       'Price_euros', 'TouchScreen', 'IPS', 'PPI', 'Cpu_name', 'Memory_first',
       'Memory_first-SSD', 'Memory_first-HDD', 'Memory_first-Hybrid',
       'Memory_first-Flash Storage', 'Memory_second', 'Memory_second-SSD',
       'Memory_second-HDD', 'Memory_second-Hybrid',
       'Memory_second-Flash Storage', 'HDD', 'SSD', 'Hybrid', 'Flash Storage'],
      dtype='object')
```

[76]:
```
df.drop(columns = ['Memory_first','Memory_second','Memory_first-SSD',␣
↪'Memory_first-HDD', 'Memory_first-Hybrid',
                   'Memory_first-Flash␣
↪Storage','Memory_second-SSD','Memory_second-HDD', 'Memory_second-Hybrid',
                   'Memory_second-Flash Storage'], axis = 1,inplace = True)
```

[77]:
```
df.head()
```

[77]:
```
   Company   TypeName  Ram_GB             Memory                         Gpu  \
0    Apple  Ultrabook       8          128 SSD  Intel Iris Plus Graphics 640
1    Apple  Ultrabook       8  128 Flash Storage        Intel HD Graphics 6000
2       HP   Notebook       8          256 SSD         Intel HD Graphics 620
3    Apple  Ultrabook      16          512 SSD            AMD Radeon Pro 455
4    Apple  Ultrabook       8          256 SSD  Intel Iris Plus Graphics 650

    OpSys  Weight_KG  Price_euros  TouchScreen  IPS         PPI       Cpu_name  \
0   macOS       1.37      1339.69            0    1  226.992481  Intel Core i5
1   macOS       1.34       898.94            0    0  127.669173  Intel Core i5
2  No OS        1.86       575.00            0    0  141.217949  Intel Core i5
3   macOS       1.83      2537.45            0    1  220.519481  Intel Core i7
```

```
4  macOS       1.37      1803.60            0    1  226.992481  Intel Core i5

   HDD  SSD  Hybrid  Flash Storage
0    0  128       0              0
1    0    0       0            128
2    0  256       0              0
3    0  512       0              0
4    0  256       0              0
```

[78]:
```
df.drop(columns = ['Memory'],axis = 1, inplace = True)
```

[79]:
```
df.shape
```

[79]:
```
(1303, 15)
```

[80]:
```
df.head()
```

[80]:
```
  Company   TypeName  Ram_GB                           Gpu  OpSys  Weight_KG  \
0   Apple  Ultrabook       8  Intel Iris Plus Graphics 640  macOS       1.37
1   Apple  Ultrabook       8        Intel HD Graphics 6000  macOS       1.34
2      HP   Notebook       8         Intel HD Graphics 620  No OS       1.86
3   Apple  Ultrabook      16            AMD Radeon Pro 455  macOS       1.83
4   Apple  Ultrabook       8  Intel Iris Plus Graphics 650  macOS       1.37

   Price_euros  TouchScreen  IPS         PPI       Cpu_name  HDD  SSD  Hybrid  \
0      1339.69            0    1  226.992481  Intel Core i5    0  128       0
1       898.94            0    0  127.669173  Intel Core i5    0    0       0
2       575.00            0    0  141.217949  Intel Core i5    0  256       0
3      2537.45            0    1  220.519481  Intel Core i7    0  512       0
4      1803.60            0    1  226.992481  Intel Core i5    0  256       0

   Flash Storage
0              0
1            128
2              0
3              0
4              0
```

[81]:
```
df[['HDD','SSD','Hybrid','Flash Storage','Price_euros']].corr()
```

[81]:
```
                    HDD       SSD    Hybrid  Flash Storage  Price_euros
HDD            1.000000 -0.399896 -0.076596      -0.117658    -0.096441
SSD           -0.399896  1.000000 -0.059750      -0.147991     0.670799
Hybrid        -0.076596 -0.059750  1.000000      -0.014368     0.007989
Flash Storage -0.117658 -0.147991 -0.014368       1.000000    -0.040511
Price_euros   -0.096441  0.670799  0.007989      -0.040511     1.000000
```

- From the above relation it is clear that “Hybrid”, “Flash Storage” has almost negligible corelation with price.
- Also it has negative correlation with “HDD” as day by day people preferring ‘SSD’ over ‘HDD’

[82]:
```
# so drop the Hybrid and Flash Storagre column

df.drop(columns = ['Hybrid','Flash Storage'], axis = 1, inplace = True)
```

[85]:
```
df.head()
```

[85]:

	Company	TypeName	Ram_GB	Gpu	OpSys	Weight_KG	Price_euros	TouchScreen	IPS	PPI	Cpu_name	HDD	SSD
0	Apple	Ultrabook	8	Intel Iris Plus Graphics 640	macOS	1.37	1339.69	0	1	226.992481	Intel Core i5	0	128
1	Apple	Ultrabook	8	Intel HD Graphics 6000	macOS	1.34	898.94	0	0	127.669173	Intel Core i5	0	0
2	HP	Notebook	8	Intel HD Graphics 620	No OS	1.86	575.00	0	0	141.217949	Intel Core i5	0	256
3	Apple	Ultrabook	16	AMD Radeon Pro 455	macOS	1.83	2537.45	0	1	220.519481	Intel Core i7	0	512
4	Apple	Ultrabook	8	Intel Iris Plus Graphics 650	macOS	1.37	1803.60	0	1	226.992481	Intel Core i5	0	256

[86]:
```
# Analysing 'Gpu'

df['Gpu'].value_counts()
```

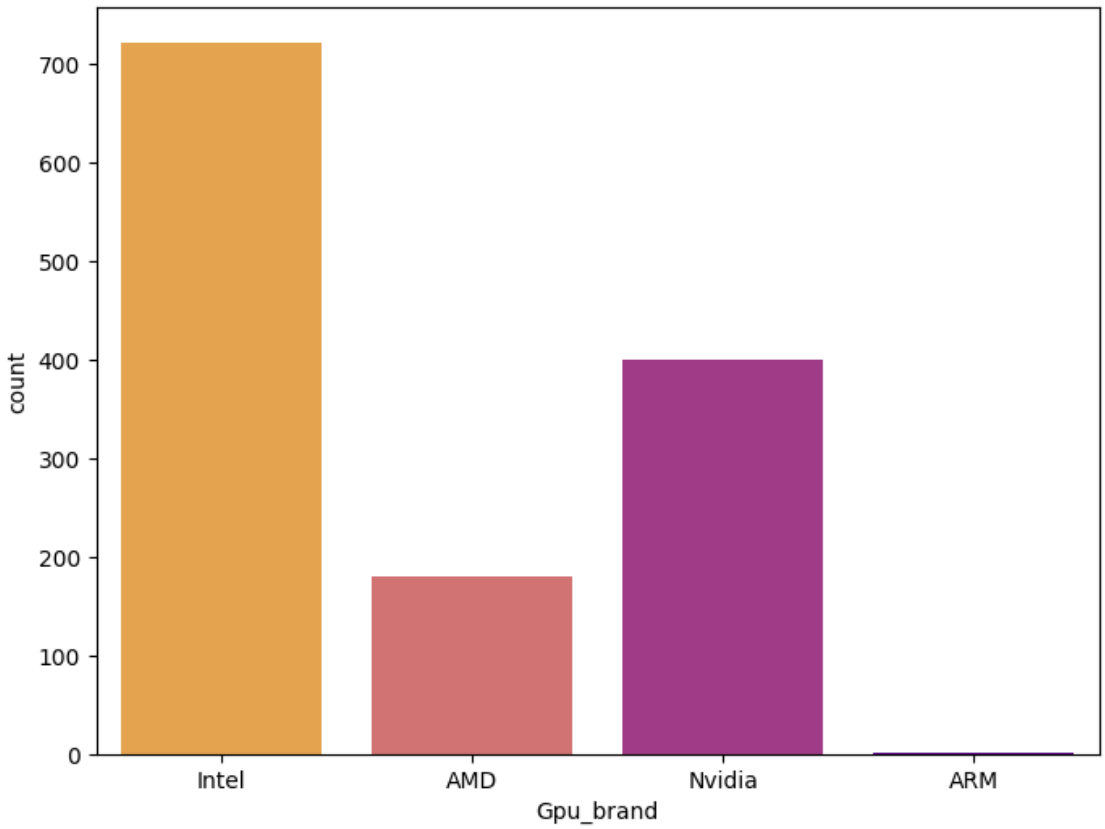
[86]:
```
Gpu
Intel HD Graphics 620      281
Intel HD Graphics 520      185
Intel UHD Graphics 620      68
Nvidia GeForce GTX 1050     66
Nvidia GeForce GTX 1060     48
                          ...
AMD Radeon R5 520            1
AMD Radeon R7                1
Intel HD Graphics 540        1
AMD Radeon 540               1
ARM Mali T860 MP4            1
Name: count, Length: 110, dtype: int64
```

[87]:
```
df['Gpu_brand'] = df['Gpu'].str.split(' ').apply(lambda x:x[0])
```

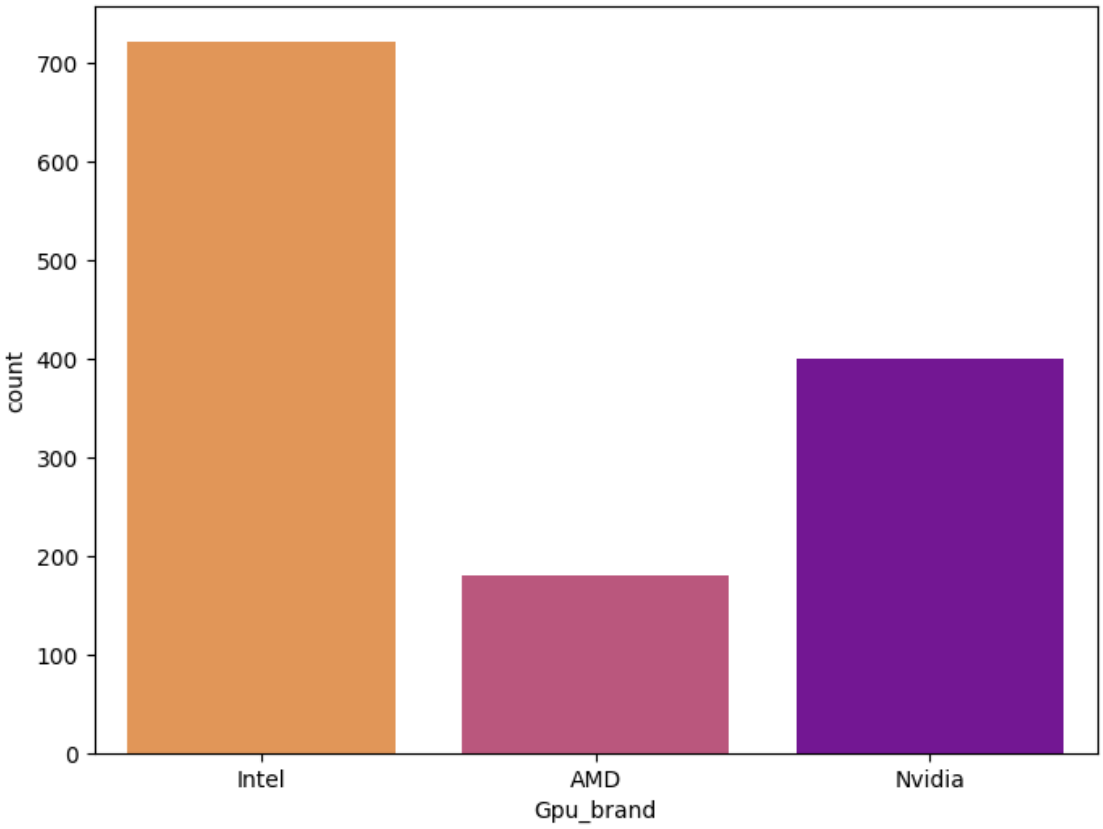
[88]:
```
df['Gpu_brand'].value_counts()
```

[88]:
```
Gpu_brand
Intel     722
Nvidia    400
AMD       180
ARM         1
Name: count, dtype: int64
```

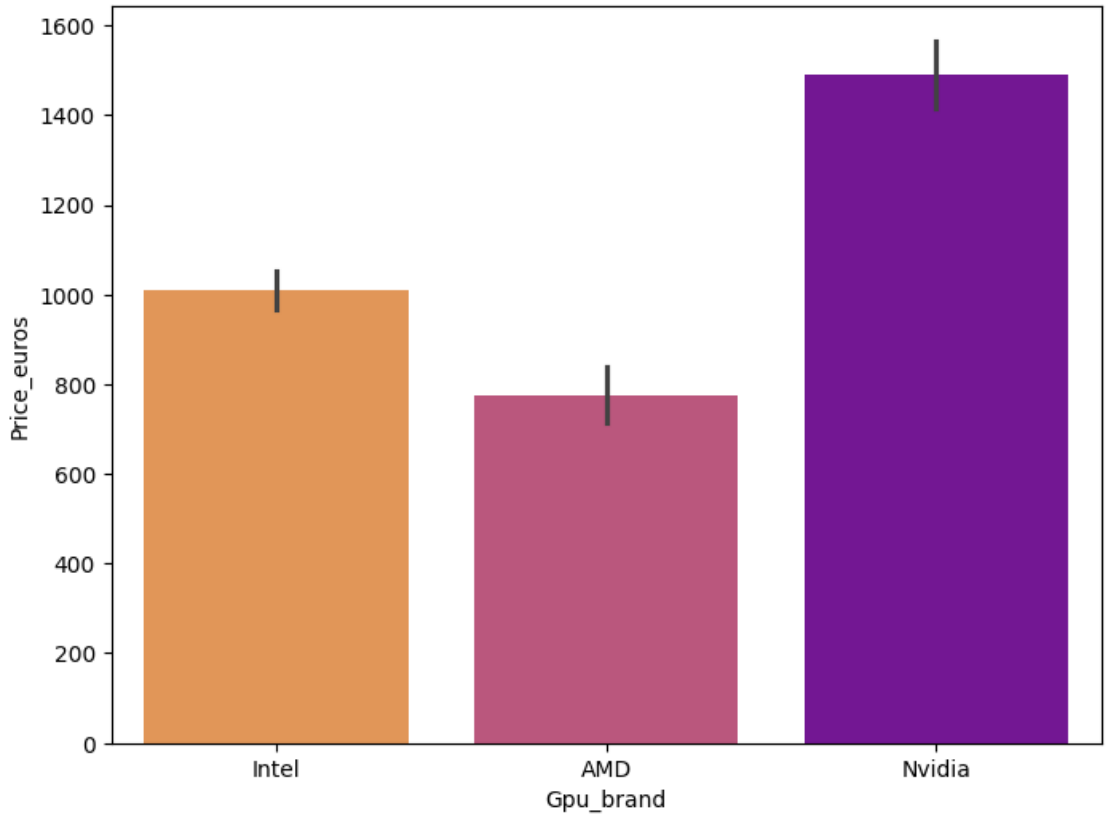
[89]:
```
plt.figure(figsize = (8,6))
sns.countplot(data = df, x = df['Gpu_brand'], palette = 'plasma_r')
plt.show()
```



[90]:
```
# Remove all laptop have gpu brand 'ARM'

df = df[df['Gpu_brand'] != 'ARM']

plt.figure(figsize = (8,6))
sns.countplot(data = df, x = df['Gpu_brand'], palette = 'plasma_r')
plt.show()
```



[91]:
```
plt.figure(figsize = (8,6))
sns.barplot(data = df, x = df['Gpu_brand'], y = df['Price_euros'], palette =␣
↪'plasma_r')
plt.show()
```



[92]:
```
df.drop(columns = ['Gpu'], axis = 1, inplace = True)
```

[93]:
```
df.head()
```

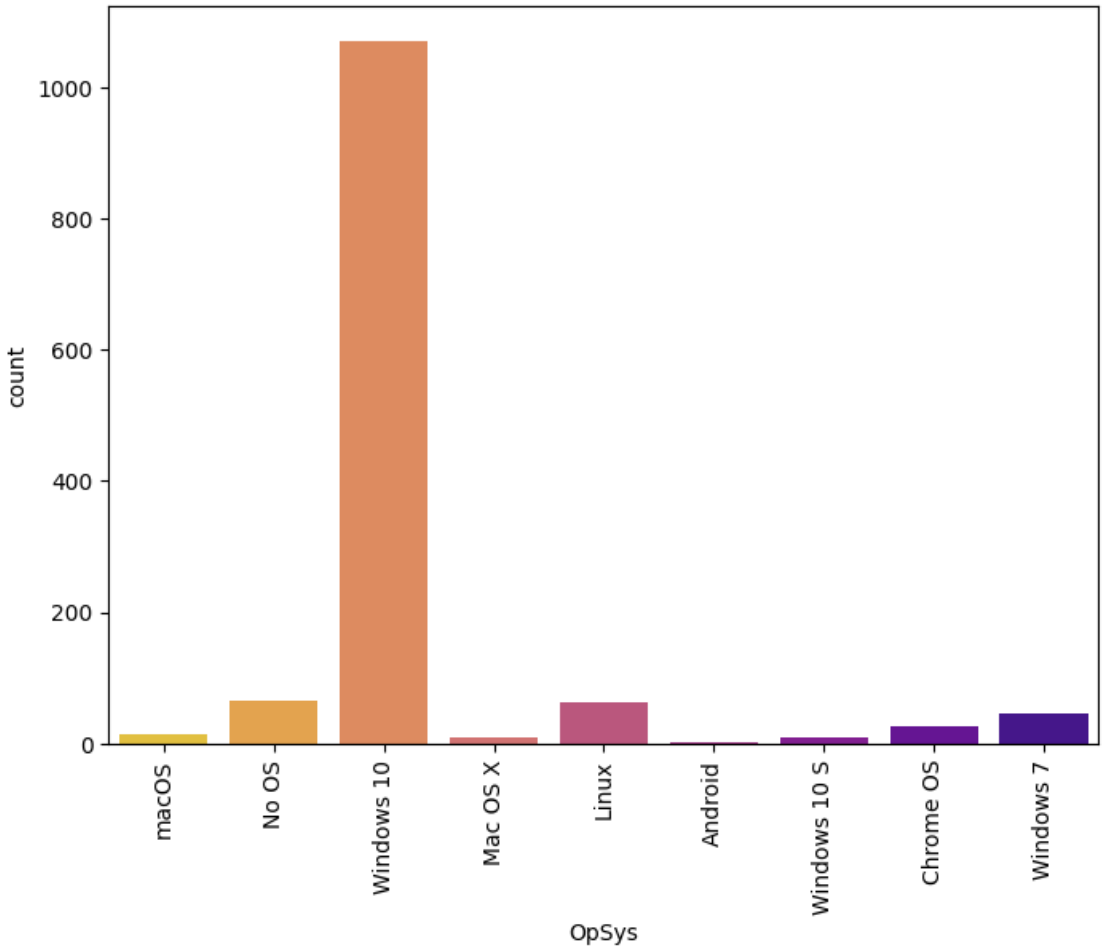
[93]:

	Company	TypeName	Ram_GB	OpSys	Weight_KG	Price_euros	TouchScreen	IPS	PPI	Cpu_name	HDD	SSD	Gpu_brand
0	Apple	Ultrabook	8	macOS	1.37	1339.69	0	1	226.992481	Intel Core i5	0	128	Intel
1	Apple	Ultrabook	8	macOS	1.34	898.94	0	0	127.669173	Intel Core i5	0	0	Intel
2	HP	Notebook	8	No OS	1.86	575.00	0	0	141.217949	Intel Core i5	0	256	Intel
3	Apple	Ultrabook	16	macOS	1.83	2537.45	0	1	220.519481	Intel Core i7	0	512	AMD
4	Apple	Ultrabook	8	macOS	1.37	1803.60	0	1	226.992481	Intel Core i5	0	256	Intel

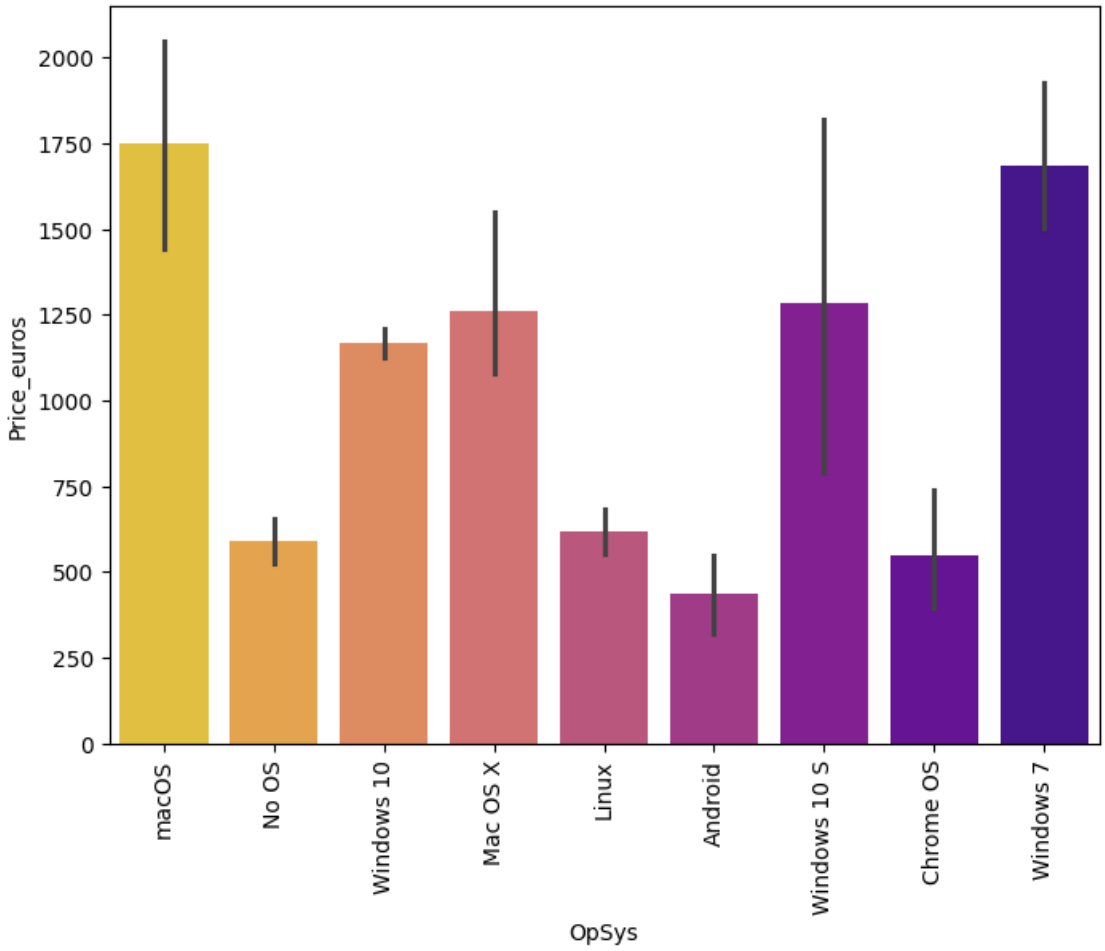
[94]:
```
# Operating System Analysis

df['OpSys'].value_counts()
```

[94]:
```
OpSys
Windows 10      1072
No OS             66
Linux             62
Windows 7         45
Chrome OS         26
macOS             13
Mac OS X           8
Windows 10 S       8
Android            2
Name: count, dtype: int64
```

[95]:
```
# Countplot

plt.figure(figsize = (8,6))
sns.countplot(data = df, x = df['OpSys'], palette = 'plasma_r')
plt.xticks(rotation = 'vertical')
plt.show()
```



[96]:
```
# Variation in price

plt.figure(figsize = (8,6))
sns.barplot(data = df, x = df['OpSys'], y = df['Price_euros'], palette =␣
 ↪'plasma_r')
plt.xticks(rotation = 'vertical')
plt.show()
```

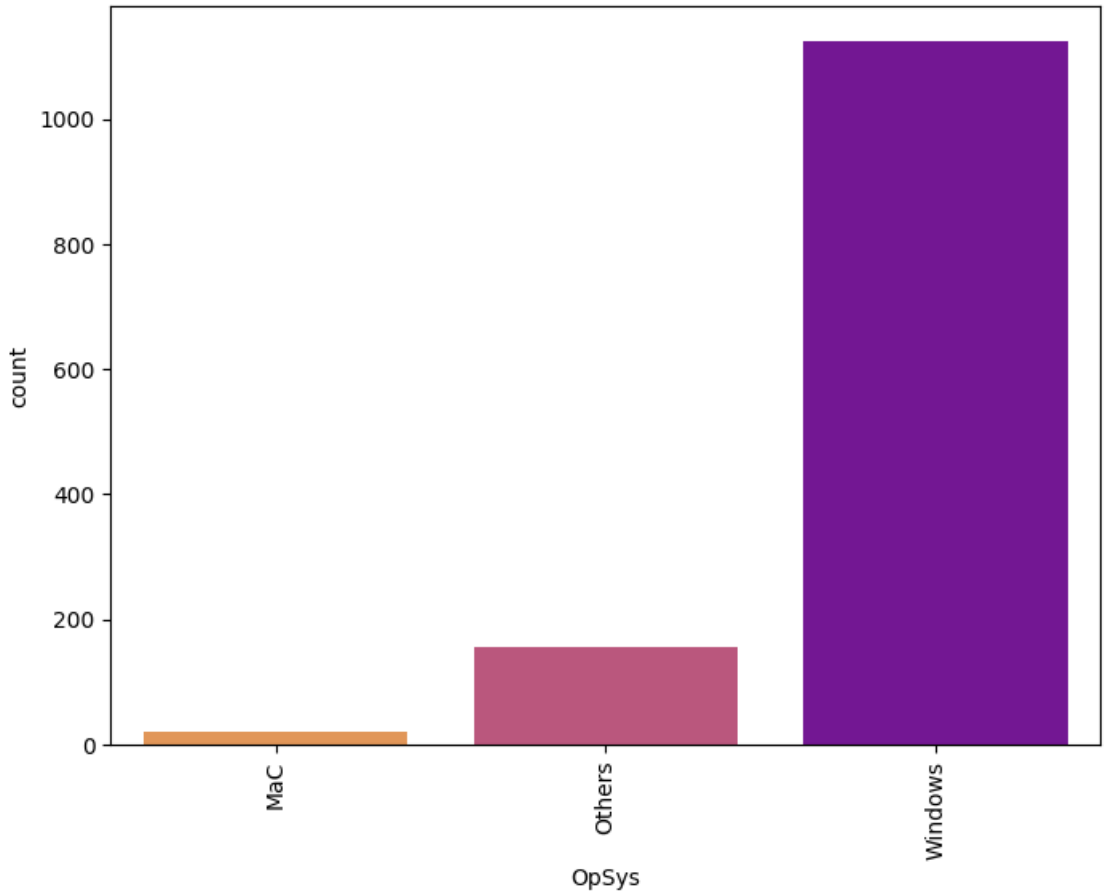


[97]:
```
# Clubing Windows all variations

def clubWindows(text):

    if text == 'Windows 10' or text == 'Windows 10 S' or text == 'Windows 7':
        return 'Windows'
    elif text == 'Mac OS X' or text == 'macOS':
        return 'MaC'
```

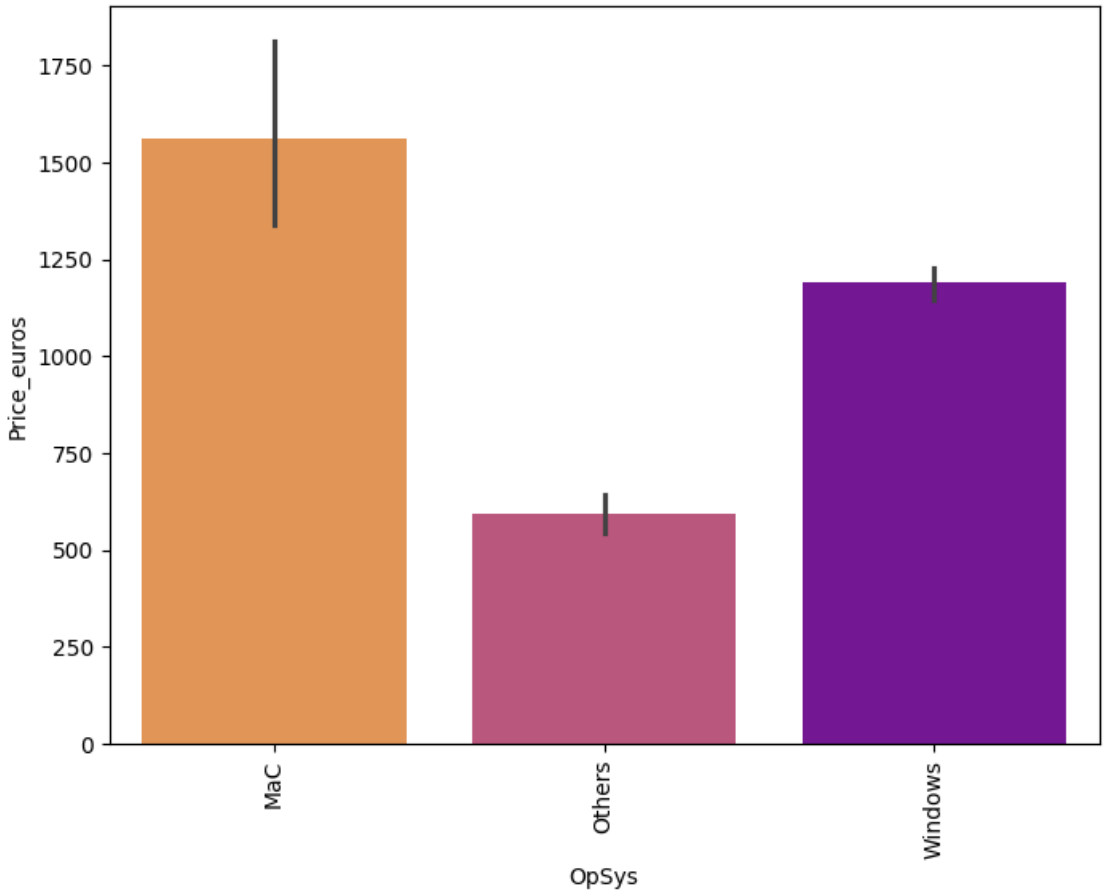
```
    else:
        return 'Others'

df['OpSys'] = df['OpSys'].apply(lambda x:clubWindows(x))
df['OpSys'].value_counts()
```

[97]:
```
OpSys
Windows    1125
Others      156
MaC          21
Name: count, dtype: int64
```

[98]:
```
plt.figure(figsize = (8,6))
sns.countplot(data = df, x = df['OpSys'], palette = 'plasma_r')
plt.xticks(rotation = 'vertical')
plt.show()
```



[99]:
```
# Variation in price

plt.figure(figsize = (8,6))
sns.barplot(data = df, x = df['OpSys'], y = df['Price_euros'], palette =␣
↪'plasma_r')
plt.xticks(rotation = 'vertical')
plt.show()
```

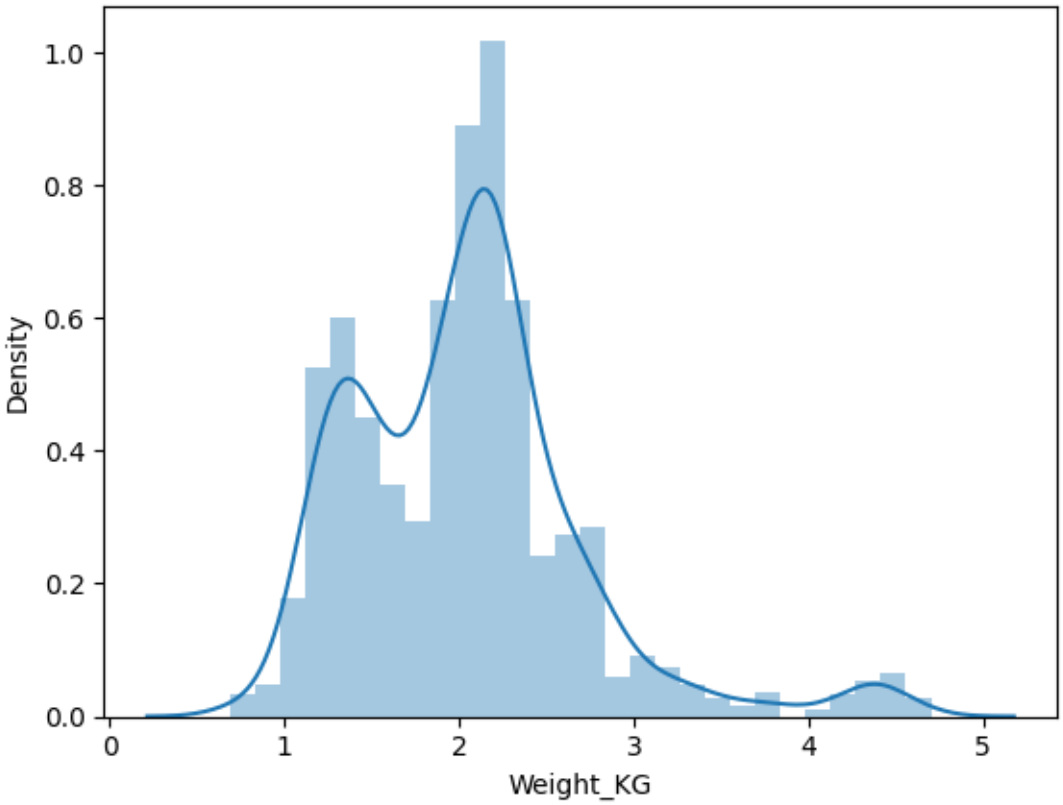


[100]:
```
# Weight Analysis

df['Weight_KG']
```

[100]:
```
0       1.37
1       1.34
2       1.86
3       1.83
4       1.37
       ...
1298    1.80
```

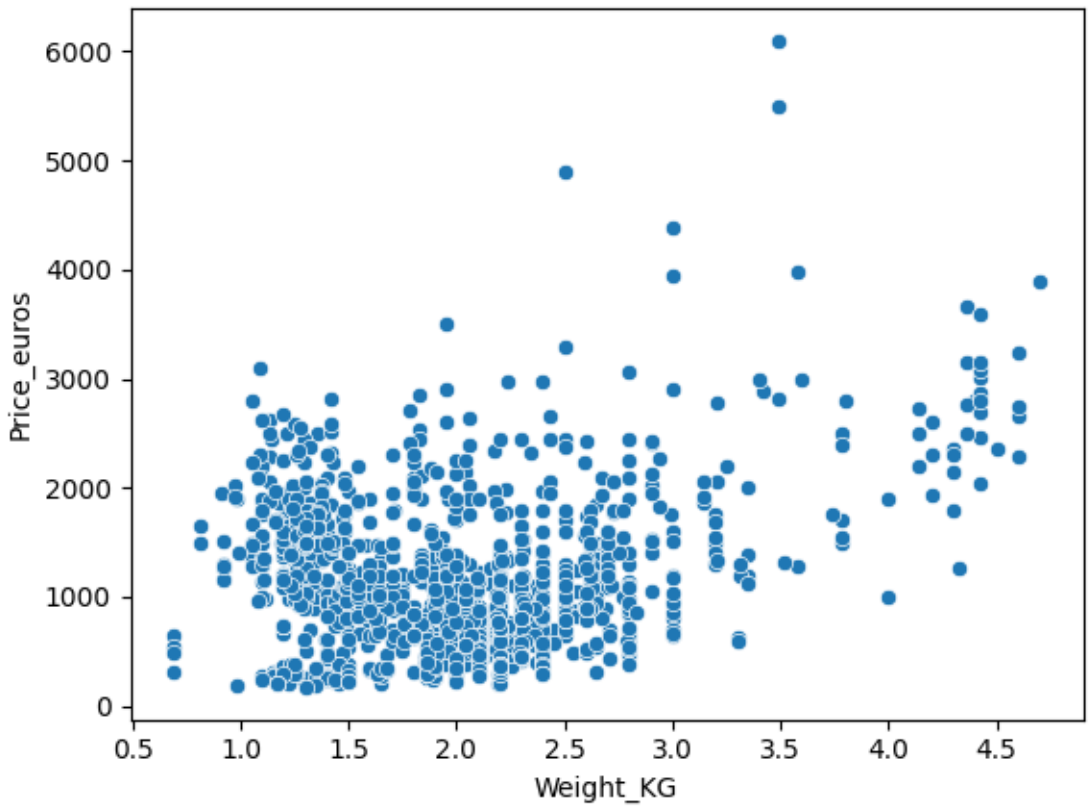
```
1299    1.30
1300    1.50
1301    2.19
1302    2.20
Name: Weight_KG, Length: 1302, dtype: float32
```

[101]:
```
# Distribution plot

sns.distplot(df['Weight_KG'])
plt.show()
```



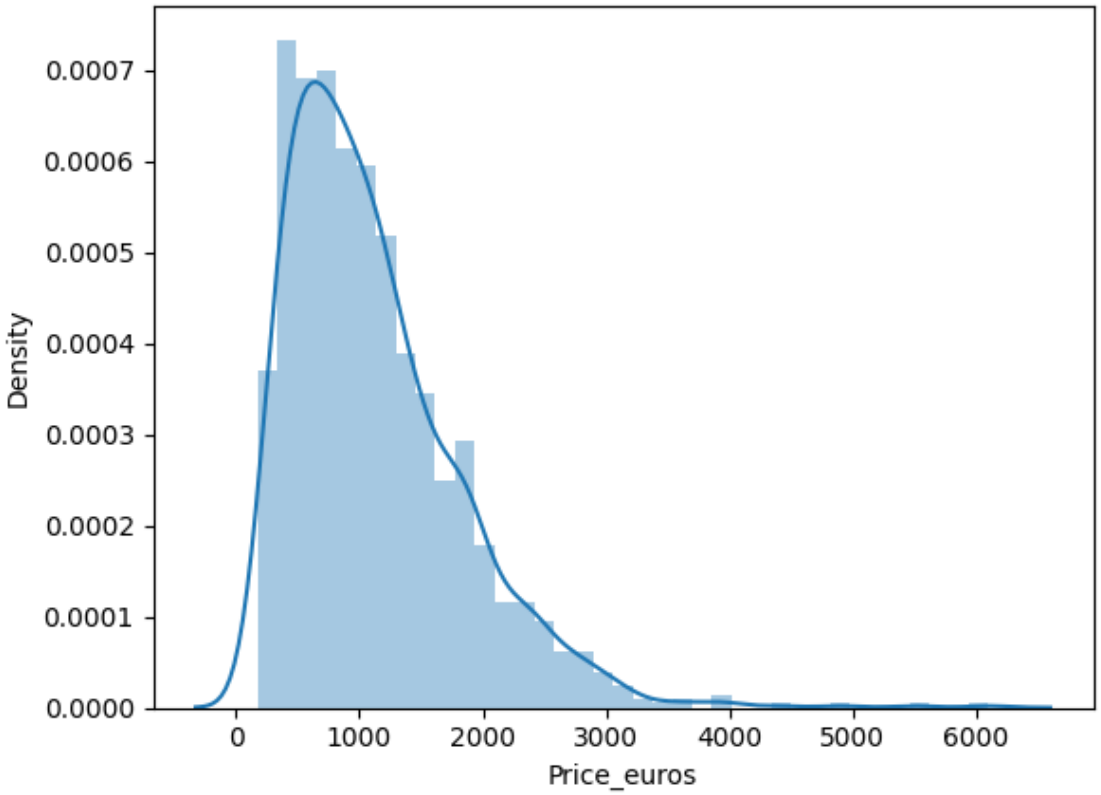
[102]:
```
# Price variations w.r.t Weight

sns.scatterplot(data = df, x = df['Weight_KG'], y = df['Price_euros'])
```

[102]:
```
<Axes: xlabel='Weight_KG', ylabel='Price_euros'>
```



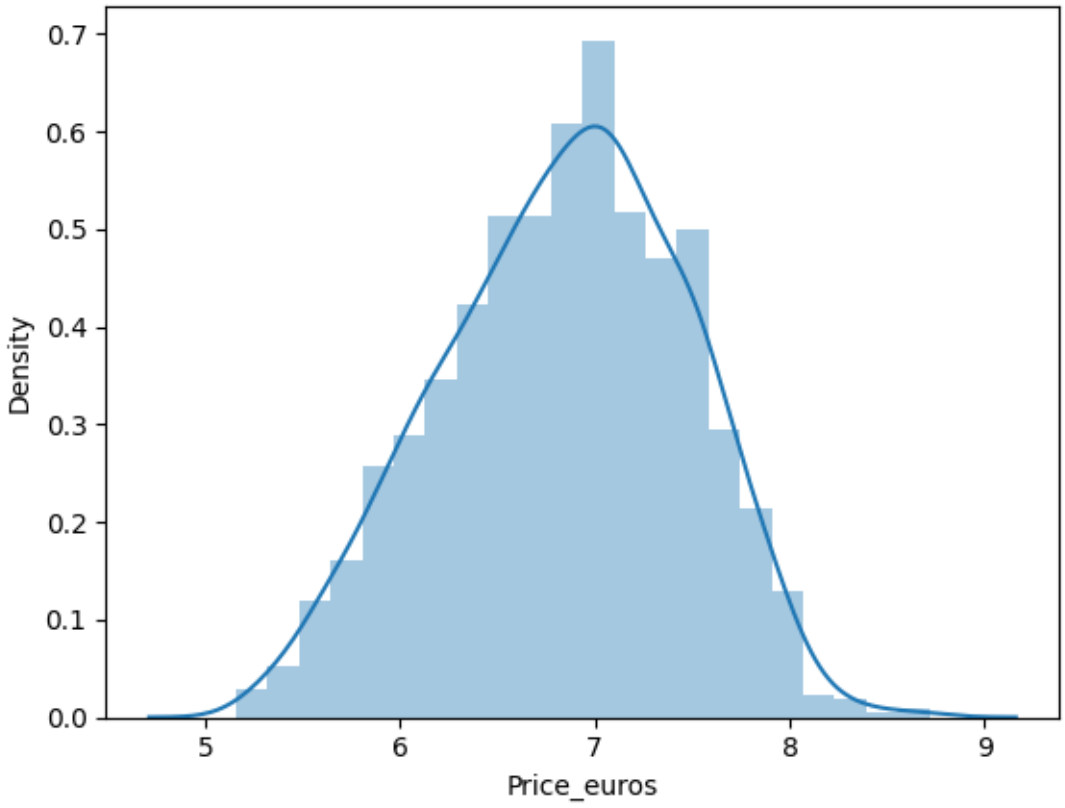
[103]:
```
# Price analysis

sns.distplot(df['Price_euros'])
```

[103]: <Axes: xlabel='Price_euros', ylabel='Density'>



[104]:
```
# apply log to normaize price

sns.distplot(np.log(df['Price_euros']))
```

[104]: <Axes: xlabel='Price_euros', ylabel='Density'>



[105]:
```
df.head()
```

[105]:

	Company	TypeName	Ram_GB	OpSys	Weight_KG	Price_euros	TouchScreen	\
0	Apple	Ultrabook	8	MaC	1.37	1339.69	0	
1	Apple	Ultrabook	8	MaC	1.34	898.94	0	
2	HP	Notebook	8	Others	1.86	575.00	0	
3	Apple	Ultrabook	16	MaC	1.83	2537.45	0	
4	Apple	Ultrabook	8	MaC	1.37	1803.60	0	

	IPS	PPI	Cpu_name	HDD	SSD	Gpu_brand
0	1	226.992481	Intel Core i5	0	128	Intel
1	0	127.669173	Intel Core i5	0	0	Intel
2	0	141.217949	Intel Core i5	0	256	Intel
3	1	220.519481	Intel Core i7	0	512	AMD
4	1	226.992481	Intel Core i5	0	256	Intel

[106]:
```
df_clean = df.copy()
df_clean
```

[106]:

	Company	TypeName	Ram_GB	OpSys	Weight_KG	Price_euros	\
0	Apple	Ultrabook	8	MaC	1.37	1339.69	

```
1      Apple           Ultrabook       8      MaC       1.34       898.94
2         HP            Notebook       8   Others       1.86       575.00
3      Apple           Ultrabook      16      MaC       1.83      2537.45
4      Apple           Ultrabook       8      MaC       1.37      1803.60
…      …                     …     …       …         …          …
1298  Lenovo  2 in 1 Convertible       4  Windows       1.80       638.00
1299  Lenovo  2 in 1 Convertible      16  Windows       1.30      1499.00
1300  Lenovo            Notebook       2  Windows       1.50       229.00
1301      HP            Notebook       6  Windows       2.19       764.00
1302    Asus            Notebook       4  Windows       2.20       369.00

      TouchScreen  IPS         PPI                Cpu_name   HDD  SSD Gpu_brand
0               0    1  226.992481           Intel Core i5     0  128     Intel
1               0    0  127.669173           Intel Core i5     0    0     Intel
2               0    0  141.217949           Intel Core i5     0  256     Intel
3               0    1  220.519481           Intel Core i7     0  512       AMD
4               0    1  226.992481           Intel Core i5     0  256     Intel
…           …   …          …                      …   …  …       …
1298            1    1  157.357143           Intel Core i7     0  128     Intel
1299            1    1  276.090226           Intel Core i7     0  512     Intel
1300            0    0  111.928571  Other Intel Processor     0    0     Intel
1301            0    0  100.448718           Intel Core i7  1000    0       AMD
1302            0    0  100.448718  Other Intel Processor   500    0     Intel

[1302 rows x 13 columns]
```

[107]:
```
numF = [feature for feature in df.columns if df[feature].dtypes != 'object']
catF = [feature for feature in df.columns if df[feature].dtypes == 'object']
```

[108]:
```
dfnum = df[numF]
```

[109]:
```
dfnum
```

[109]:
```
      Ram_GB  Weight_KG  Price_euros  TouchScreen  IPS         PPI   HDD  SSD
0          8       1.37      1339.69            0    1  226.992481     0  128
1          8       1.34       898.94            0    0  127.669173     0    0
2          8       1.86       575.00            0    0  141.217949     0  256
3         16       1.83      2537.45            0    1  220.519481     0  512
4          8       1.37      1803.60            0    1  226.992481     0  256
…      …         …           …           …  …          …    …  …
1298       4       1.80       638.00            1    1  157.357143     0  128
1299      16       1.30      1499.00            1    1  276.090226     0  512
1300       2       1.50       229.00            0    0  111.928571     0    0
1301       6       2.19       764.00            0    0  100.448718  1000    0
1302       4       2.20       369.00            0    0  100.448718   500    0

[1302 rows x 8 columns]
```

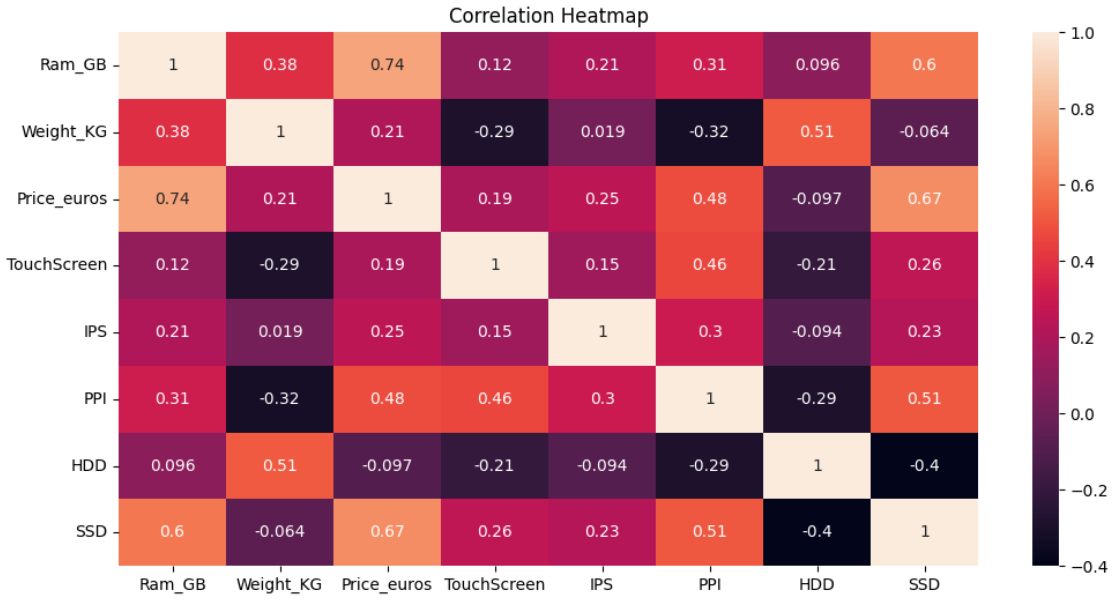
[110]:
```
dfnum.corr()
```

[110]:

	Ram_GB	Weight_KG	Price_euros	TouchScreen	IPS	\
Ram_GB	1.000000	0.383362	0.742905	0.118875	0.207949	
Weight_KG	0.383362	1.000000	0.209867	-0.293004	0.018643	
Price_euros	0.742905	0.209867	1.000000	0.192917	0.253320	
TouchScreen	0.118875	-0.293004	0.192917	1.000000	0.148026	
IPS	0.207949	0.018643	0.253320	0.148026	1.000000	
PPI	0.305712	-0.321842	0.475377	0.458553	0.299145	
HDD	0.095808	0.514147	-0.096891	-0.208766	-0.093588	
SSD	0.603379	-0.063818	0.670660	0.257577	0.225311	

	PPI	HDD	SSD
Ram_GB	0.305712	0.095808	0.603379
Weight_KG	-0.321842	0.514147	-0.063818
Price_euros	0.475377	-0.096891	0.670660
TouchScreen	0.458553	-0.208766	0.257577
IPS	0.299145	-0.093588	0.225311
PPI	1.000000	-0.294678	0.509451
HDD	-0.294678	1.000000	-0.400750
SSD	0.509451	-0.400750	1.000000

[111]:
```
plt.figure(figsize = (12,6))
sns.heatmap(dfnum.corr(), annot = True)
plt.title('Correlation Heatmap')
plt.show()
```



[112]:
```
# Getting my X and y

X = df.drop(['Price_euros'],axis=1)

y = np.log(df['Price_euros'])
```

[113]:
```
X.head()
```

[113]:
```
   Company   TypeName  Ram_GB   OpSys  Weight_KG  TouchScreen  IPS         PPI  \
0   Apple  Ultrabook       8     MaC       1.37            0    1  226.992481
1   Apple  Ultrabook       8     MaC       1.34            0    0  127.669173
2      HP   Notebook       8  Others       1.86            0    0  141.217949
3   Apple  Ultrabook      16     MaC       1.83            0    1  220.519481
4   Apple  Ultrabook       8     MaC       1.37            0    1  226.992481

         Cpu_name  HDD  SSD Gpu_brand
0  Intel Core i5    0  128     Intel
1  Intel Core i5    0    0     Intel
2  Intel Core i5    0  256     Intel
3  Intel Core i7    0  512       AMD
4  Intel Core i5    0  256     Intel
```

[114]:
```
y.head()
```

[114]:
```
0    7.200194
1    6.801216
2    6.354370
3    7.838915
4    7.497540
Name: Price_euros, dtype: float64
```

[115]:
```
# Saving dataframe as csv

X.to_csv('traintest.csv', index=False)
```

[116]:
```
# Importing library for model building

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn import metrics
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import RandomizedSearchCV
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn import tree
```

[117]:
```
# Spliting dataset into train and test

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.15,␣
 ↪random_state = 2)

X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

[117]:
```
((1106, 12), (196, 12), (1106,), (196,))
```

[118]:
```
mapper = {i:value for i,value in enumerate(X_train.columns)}

mapper
```

[118]:
```
{0: 'Company',
 1: 'TypeName',
 2: 'Ram_GB',
 3: 'OpSys',
 4: 'Weight_KG',
 5: 'TouchScreen',
 6: 'IPS',
 7: 'PPI',
 8: 'Cpu_name',
 9: 'HDD',
 10: 'SSD',
 11: 'Gpu_brand'}
```

[119]:
```
# columnTransformer will use to apply different trasformation to different␣
 ↪columns

step1 = ColumnTransformer(transformers = [('col_tnf',␣
 ↪OneHotEncoder(sparse_output = False, drop =␣
 ↪'first'),[0,1,3,8,11])],remainder = 'passthrough')
```

[120]:
```
step2 = LinearRegression()
```

[121]:
```
pipe = Pipeline([
    ('step1',step1),
    ('step2',step2)
])
```

[122]:
```
pipe.fit(X_train,y_train)
```

[122]:
```
Pipeline(steps=[('step1',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('col_tnf',
                                                  OneHotEncoder(drop='first',
sparse_output=False),
```

```
                                                    [0, 1, 3, 8, 11])])),
                ('step2', LinearRegression())])
```

[123]:
```
coefficients = pipe.named_steps['step2'].coef_
intercept = pipe.named_steps['step2'].intercept_

print('coefficients\n')
print(coefficients)
print('\n')
print('intercept\n')
print(intercept)
```

```
coefficients

[ 3.07044239e-01  1.08391539e-01 -3.25734233e-01  1.57800887e-01
 -2.05597991e-03  4.77848564e-01  1.88664443e-01  5.08057577e-02
  4.58680518e-01  1.05908468e-01  2.13758527e-01 -2.89265960e-01
  4.14394493e-01  2.60026925e-01  3.28367085e-01  3.11663372e-01
 -5.09370788e-01  3.65534878e-01 -1.40944948e-02 -2.04166654e-01
 -2.23107418e-01  2.69571022e-02  5.00558037e-01 -2.68981046e-01
 -3.80631938e-02  7.89146951e-02  3.89749471e-01  4.50304052e-01
 -1.75203681e-01  1.04728619e-01  9.46067892e-02  2.45454639e-02
  8.45816029e-02 -1.06161703e-01  3.94970916e-02  2.37946025e-03
  8.26267261e-06  6.23784408e-04]


intercept

5.656016528938852
```

[124]:
```
y_pred_LR = pipe.predict(X_test)
y_pred_LR
```

[124]:
```
array([6.73995697, 5.71233467, 6.08215907, 6.3886321 , 6.95520093,
       7.42932259, 6.28334069, 6.43364678, 7.16910102, 5.61653262,
       6.95116007, 7.05635187, 7.76322801, 6.91325292, 8.00184611,
       6.8600843 , 6.82209117, 5.62670867, 7.41147649, 7.51143965,
       7.69573076, 7.13293284, 6.84386414, 7.30967163, 6.52985406,
       7.79698669, 7.07352198, 6.15219287, 6.91631631, 6.64733872,
       7.11536536, 6.04723074, 6.61236793, 7.08096101, 7.13458068,
       6.81189073, 6.95879587, 6.98100906, 7.23797964, 7.45458039,
       7.71332488, 7.29154081, 5.77710651, 6.0242735 , 6.22395211,
       6.03488453, 7.03124121, 7.9488529 , 7.55048895, 6.8652697 ,
       7.54592851, 7.50789952, 5.77523693, 5.55097061, 6.76299228,
       6.80795999, 6.09242144, 7.08896744, 6.23279742, 6.49844655,
       7.02380118, 7.12612253, 5.90511632, 6.49970388, 6.80849634,
       6.57663716, 6.05659937, 7.48469014, 6.77039895, 5.94738386,
```

```
       6.99893067, 6.77216167, 6.64511266, 6.4740318 , 6.65022965,
       7.1315841 , 7.33258165, 7.93965987, 7.28645088, 6.57555475,
       6.64201473, 6.95788519, 7.10805991, 6.29597537, 6.93391618,
       6.66161244, 7.17145573, 7.25163957, 6.20359874, 6.93963812,
       6.7993016 , 6.38569908, 6.45248364, 7.76030707, 6.33253407,
       5.63777578, 6.47318599, 6.2161444 , 5.96181711, 7.64126926,
       6.61964769, 7.19463362, 5.93641051, 7.72195841, 6.84442808,
       7.17837227, 7.15140896, 6.64442146, 7.09798274, 6.0095782 ,
       6.79118379, 6.82833943, 6.82429609, 7.37458788, 6.32537699,
       6.73995697, 5.64608716, 7.81080141, 6.61359731, 6.56746786,
       6.43820683, 7.72457375, 6.97340016, 6.450287  , 6.72236896,
       7.25821668, 6.57791461, 6.00236191, 7.66034015, 7.36294137,
       6.44033568, 7.29316878, 6.33430951, 7.32271946, 7.09837976,
       6.59840454, 6.9862482 , 7.17837227, 7.39930168, 7.37458788,
       6.01930535, 5.95025333, 6.84572336, 6.45829487, 6.59186184,
       7.39128065, 6.60696044, 5.96048727, 7.53604195, 6.43658454,
       7.50375579, 7.22303506, 6.61156536, 7.23148437, 6.89990495,
       6.57832879, 7.16111088, 7.95661448, 7.31721014, 7.46153777,
       5.87257224, 6.20088267, 6.74438091, 6.67397982, 6.88721413,
       7.03782518, 7.44000603, 6.38452598, 7.52410336, 6.93364681,
       6.48481395, 7.08197851, 6.14719787, 6.92507367, 6.80619558,
       7.40177332, 8.09238981, 7.49904556, 7.20937461, 6.33895438,
       6.92358781, 7.39274728, 7.11911544, 6.73369491, 6.16107376,
       6.70389517, 6.4102108 , 7.3823518 , 6.14216967, 6.04572348,
       6.1994674 , 7.18719873, 5.79724299, 6.95791092, 7.16601269,
       7.70810073])
```

[125]:
```
y_test
```

[125]:
```
248     6.514713
555     5.411646
1251    5.988961
547     6.136322
885     7.494986
          …
1129    7.029088
172     5.733341
322     6.997596
996     7.207119
470     7.600402
Name: Price_euros, Length: 196, dtype: float64
```

[126]:
```
print(pipe.get_params)
```

```
<bound method Pipeline.get_params of Pipeline(steps=[('step1',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('col_tnf',
```

```
                                                    OneHotEncoder(drop='first',
sparse_output=False),
                                                    [0, 1, 3, 8, 11])])),
                ('step2', LinearRegression())])>
```

[127]:
```
print("MSE = ", metrics.mean_squared_error(y_test,y_pred_LR))

print('MAE = ', metrics.mean_absolute_error(y_test, y_pred_LR))

print('R2 score = ', metrics.r2_score(y_test,y_pred_LR))
```

```
MSE =  0.07370782180866566
MAE =  0.21017986543194037
R2 score =  0.8073255624974497
```

[128]:
```
# Calculate R-squared scores
train_r2 = metrics.r2_score(y_train, pipe.predict(X_train))
test_r2 = metrics.r2_score(y_test, y_pred_LR)

# Print the results
print(f'Training R-squared: {train_r2}')
print(f'Test R-squared: {test_r2}')
```

```
Training R-squared: 0.8339738735512752
Test R-squared: 0.8073255624974497
```

- Avobe results indicates overfitting

[129]:
```
# Ridge Regression

# columnTransformer will will use to apply different trasformation to different␣
 ↪columns

step1 = ColumnTransformer(transformers = [('col_tnf',␣
 ↪OneHotEncoder(sparse_output = False, drop = 'first'),
                                          [0,1,3,8,11])],remainder =␣
 ↪'passthrough')

step2 = Ridge(alpha = 10)

pipe = Pipeline([
    ('step1',step1),
    ('step2',step2)
])

pipe.fit(X_train,y_train)
```

[129]:
```
Pipeline(steps=[('step1',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('col_tnf',
                                                  OneHotEncoder(drop='first',
sparse_output=False),
                                                  [0, 1, 3, 8, 11])])),
                ('step2', Ridge(alpha=10))])
```

[130]:
```
coefficients = pipe.named_steps['step2'].coef_
intercept = pipe.named_steps['step2'].intercept_

print('Coefficient:\n')
print(coefficients)
print('\n')
print('Intercepts:\n')
print(intercept)
```

```
Coefficient:

[ 1.69733745e-01  1.98210582e-02 -8.37202171e-02  7.47256926e-02
 -1.40168506e-02  6.96048351e-02  1.11330330e-01 -7.78921388e-03
  7.66410317e-02  2.51623450e-02  1.03149102e-01 -1.22453287e-01
  7.84201936e-02  5.19229390e-02  8.03133689e-02  1.92211716e-01
 -1.56455173e-01  5.48782376e-02 -1.54777357e-03 -1.27951836e-01
 -2.10912298e-01  7.63613000e-02  3.52842436e-01 -1.84524416e-01
  1.47906710e-02  1.21754345e-03  3.09373040e-01  3.61942490e-01
 -2.66685810e-01  9.81274656e-02  1.11153726e-01  2.67976464e-02
  7.82061121e-02 -7.20840779e-02  3.06536736e-02  2.36437820e-03
  9.44074783e-06  6.51096074e-04]


Intercepts:

5.732313969455914
```

[131]:
```
y_pred_ridge = pipe.predict(X_test)
y_pred_ridge
```

[131]:
```
array([6.70574839, 5.69396191, 6.14239576, 6.38820723, 6.95945987,
       7.46268575, 6.25590498, 6.48079724, 7.13491049, 5.68668183,
       6.96158834, 7.06132826, 7.60188185, 6.91189412, 7.86471114,
       6.88744743, 6.80543585, 5.69547924, 7.40134951, 7.52462349,
       7.608144  , 7.18180584, 6.82588116, 7.34724214, 6.52837486,
       7.81197548, 7.11171856, 6.15758902, 6.89979013, 6.62361046,
       7.08934348, 6.09855569, 6.72044316, 7.07420443, 7.11885984,
       6.79487259, 6.95538088, 6.96316828, 7.23742474, 7.43892476,
       7.55574025, 7.30917784, 5.77010251, 5.9863784 , 6.273756  ,
```

```
6.04242568, 7.10720699, 7.76429215, 7.37988317, 6.85100884,
7.52338735, 7.49050448, 5.86214391, 5.61794194, 6.73789452,
6.79000938, 6.14316752, 7.11829669, 6.19483698, 6.49108852,
7.1219062 , 7.11103924, 5.92004768, 6.49031586, 6.79135566,
6.54989627, 6.0614677 , 7.47322627, 6.7579197 , 5.89799737,
6.99827831, 6.74662243, 6.65586093, 6.44566455, 6.70722238,
7.15559241, 7.32711766, 7.96894562, 7.147322  , 6.55057279,
6.61223173, 6.96106409, 7.12192265, 6.26860339, 6.98343877,
6.64293315, 7.1917561 , 7.258143  , 6.26227947, 6.91268775,
6.76379981, 6.36797337, 6.55538271, 7.7966022 , 6.39921506,
5.70542331, 6.44488249, 6.19982017, 5.91897989, 7.65987499,
6.58671224, 7.17438618, 5.94291434, 7.54978929, 6.82232746,
7.14120329, 7.1745809 , 6.42449081, 7.00694509, 5.96382286,
6.7722782 , 6.85274623, 6.80226776, 7.37905854, 6.29786248,
6.70574839, 5.85474672, 7.81708936, 6.58846194, 6.58845797,
6.450744  , 7.76798585, 6.96063866, 6.41999153, 6.70010835,
7.22454359, 6.54300995, 5.94883135, 7.66801995, 7.38441606,
6.46378957, 7.27446331, 6.33196394, 7.35239073, 7.08133707,
6.56797078, 6.96644628, 7.14120329, 7.40056686, 7.37905854,
5.98424827, 5.9098179 , 6.82615502, 6.4632639 , 6.56397336,
7.39090643, 6.57498132, 5.91022757, 7.44146341, 6.41777809,
7.49232924, 7.24198002, 6.58407753, 7.25877572, 6.88196456,
6.55146039, 7.17297159, 8.00359601, 7.31378253, 7.46350499,
5.82834228, 6.25783166, 6.73571625, 6.63943184, 6.88148133,
7.02876762, 7.42724986, 6.35717667, 7.41576033, 6.92388179,
6.42001154, 7.08101149, 6.12745674, 6.98951497, 6.77816646,
7.40913199, 7.9441461 , 7.52474948, 7.00054966, 6.40466016,
6.90386228, 7.20232368, 7.13119871, 6.7386055 , 6.1436133 ,
6.78788019, 6.41474017, 7.3491842 , 6.12397879, 6.05411122,
6.2575591 , 7.1783027 , 5.88350277, 6.92560914, 7.21293984,
7.70896971])
```

[132]:
```
print("MSE = ", metrics.mean_squared_error(y_test,y_pred_ridge))

print('MAE = ', metrics.mean_absolute_error(y_test, y_pred_ridge))

print('R2 score = ', metrics.r2_score(y_test,y_pred_ridge))
```

```
MSE =  0.07164005649770154
MAE =  0.20926927121196615
R2 score =  0.8127307624938829
```

[133]:
```
# Calculate R-squared scores
train_r2 = metrics.r2_score(y_train, pipe.predict(X_train))
test_r2 = metrics.r2_score(y_test, y_pred_ridge)

# Print the results
```

```
print(f'Training R-squared: {train_r2}')
print(f'Test R-squared: {test_r2}')
```

```
Training R-squared: 0.8243896600540359
Test R-squared: 0.8127307624938829
```

[134]:
```
# Lasso Regression

# columnTransformer will will use to apply different trasformation to different␣
 ↪columns

step1 = ColumnTransformer(transformers = [('col_tnf',
                                           OneHotEncoder(sparse_output = False,␣
 ↪drop = 'first'),
                                           [0,1,3,8,11])],
                                              remainder = 'passthrough')

step2 = Lasso(alpha = 0.0001)

pipe = Pipeline([
    ('step1',step1),
    ('step2',step2)
])

pipe.fit(X_train,y_train)
```

[134]:
```
Pipeline(steps=[('step1',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('col_tnf',
                                                  OneHotEncoder(drop='first',
sparse_output=False),
                                                  [0, 1, 3, 8, 11])])),
                ('step2', Lasso(alpha=0.0001))])
```

[135]:
```
coefficients = pipe.named_steps['step2'].coef_
intercept = pipe.named_steps['step2'].intercept_

print('Coefficient:\n')
print(coefficients)
print('\n')
print('Intercepts:\n')
print(intercept)
```

```
Coefficient:

[ 3.24008743e-01  9.27491206e-02 -2.95997332e-01  1.41179186e-01
 -0.00000000e+00  4.09805496e-01  1.73998472e-01  0.00000000e+00
```

```
 3.96791742e-01  9.13622877e-02  1.93638099e-01 -2.78407235e-01
 3.58926302e-01  2.19557010e-01  2.90336444e-01  2.93615703e-01
-4.90373526e-01  3.11143871e-01 -0.00000000e+00 -1.89598076e-01
-2.13495955e-01  3.88826686e-02  5.07784799e-01 -2.27562116e-01
 0.00000000e+00  7.66615024e-02  3.89574592e-01  4.50104809e-01
-1.79880100e-01  1.01583041e-01  9.17665776e-02  2.47275783e-02
 8.17876197e-02 -9.62199518e-02  3.77977288e-02  2.36570549e-03
 8.75013808e-06  6.27919521e-04]


Intercepts:

5.630356562390023
```

[136]:
```
y_pred_lasso = pipe.predict(X_test)
y_pred_lasso
```

[136]:
```
array([6.7368167 , 5.70824326, 6.08390152, 6.38940233, 6.9554919 ,
       7.43193624, 6.27951506, 6.445226  , 7.16682335, 5.62804796,
       6.95205817, 7.05429797, 7.75848925, 6.91113332, 8.00088818,
       6.86406999, 6.82152628, 5.63752331, 7.41230262, 7.51539486,
       7.69056515, 7.13903171, 6.8400209 , 7.31088805, 6.53051827,
       7.79650077, 7.07501658, 6.15066096, 6.9141578 , 6.64442168,
       7.11386133, 6.0484465 , 6.62828703, 7.07611593, 7.13506031,
       6.80847562, 6.95613452, 6.9798312 , 7.23820226, 7.45308281,
       7.71023456, 7.29521866, 5.77467629, 6.01815829, 6.23313597,
       6.03458395, 7.04661261, 7.94395031, 7.54422917, 6.86515878,
       7.54141343, 7.50554052, 5.78679418, 5.56264982, 6.76234666,
       6.80874745, 6.09157923, 7.0925806 , 6.22698345, 6.49293095,
       7.04021807, 7.12688155, 5.90439175, 6.49474187, 6.80955223,
       6.57505915, 6.05575418, 7.4827953 , 6.76765454, 5.94018044,
       6.99701894, 6.76814528, 6.6482561 , 6.47255399, 6.65339563,
       7.13647984, 7.33344989, 7.93964245, 7.26740263, 6.57233135,
       6.63902153, 6.95843109, 7.10800595, 6.29149505, 6.94025438,
       6.65721384, 7.17144369, 7.25557623, 6.2043536 , 6.93941879,
       6.7963015 , 6.38222661, 6.46811101, 7.75966511, 6.33418246,
       5.64820919, 6.47173612, 6.20956172, 5.95515114, 7.63985953,
       6.61667875, 7.19312952, 5.93645386, 7.715213  , 6.84260124,
       7.17488849, 7.15259617, 6.61039934, 7.09384931, 6.00059297,
       6.78951429, 6.83850764, 6.82250624, 7.37673011, 6.31914355,
       6.7368167 , 5.66592337, 7.80924434, 6.61482647, 6.5681098 ,
       6.44063588, 7.72783051, 6.97233125, 6.44582849, 6.72128494,
       7.25526219, 6.57331916, 5.9933424 , 7.65741707, 7.36491351,
       6.44229271, 7.2883324 , 6.33348889, 7.32340322, 7.09853579,
       6.59651753, 6.98568425, 7.17488849, 7.39622208, 7.37673011,
       6.01412997, 5.94458139, 6.84119065, 6.45846013, 6.58978092,
       7.39337669, 6.6044106 , 5.95113781, 7.52981776, 6.43324016,
```

```
       7.49872264, 7.22213875, 6.60957799, 7.23140305, 6.897242  ,
       6.5766949 , 7.16324497, 7.9623591 , 7.31824256, 7.46232722,
       5.86685246, 6.20151975, 6.74217898, 6.66923497, 6.88451096,
       7.03800557, 7.43975029, 6.37976429, 7.50587776, 6.93389107,
       6.47739094, 7.07749448, 6.1401061 , 6.93454068, 6.80261331,
       7.40080505, 8.08768493, 7.50184518, 7.17985403, 6.34081122,
       6.92014254, 7.3546315 , 7.12246107, 6.73080291, 6.16056302,
       6.72553078, 6.41289686, 7.37690445, 6.13014422, 6.04307402,
       6.19997853, 7.18567831, 5.80151101, 6.9557015 , 7.17770894,
       7.70614875])
```

[137]:
```
print("MSE = ", metrics.mean_squared_error(y_test,y_pred_lasso))

print('MAE = ', metrics.mean_absolute_error(y_test, y_pred_lasso))

print('R2 score = ', metrics.r2_score(y_test,y_pred_lasso))
```

```
MSE =  0.07333907159439433
MAE =  0.20991914616513468
R2 score =  0.8082894865203044
```

[138]:
```
# Calculate R-squared scores
train_r2 = metrics.r2_score(y_train, pipe.predict(X_train))
test_r2 = metrics.r2_score(y_test, y_pred_lasso)

# Print the results
print(f'Training R-squared: {train_r2}')
print(f'Test R-squared: {test_r2}')
```

```
Training R-squared: 0.8338295502344776
Test R-squared: 0.8082894865203044
```

[139]:
```
# Decision Tree Regression

# columnTransformer will will use to apply different trasformation to different␣
 ↪columns

step1 = ColumnTransformer(transformers = [('col_tnf',
                                           OneHotEncoder(sparse_output = False,␣
 ↪drop = 'first'),
                                           [0,1,3,8,11])],
                                              remainder = 'passthrough')

step2 = DecisionTreeRegressor(max_depth=9,splitter='random')

pipe = Pipeline([
    ('step1',step1),
```

```
    ('step2',step2)
])

pipe.fit(X_train,y_train)
```

[139]:
```
Pipeline(steps=[('step1',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('col_tnf',
                                                  OneHotEncoder(drop='first',
sparse_output=False),
                                                  [0, 1, 3, 8, 11])])),
                ('step2',
                 DecisionTreeRegressor(max_depth=9, splitter='random'))])
```

[140]:
```
y_pred_DT = pipe.predict(X_test)

print('R2_sore: ', metrics.r2_score(y_test,y_pred_DT))
print('MAE: ', metrics.mean_absolute_error(y_test,y_pred_DT))
print('MSE: ',metrics.mean_squared_error(y_test,y_pred_DT))
```

```
R2_sore:  0.7877429487304191
MAE:  0.21324798645801574
MSE:  0.08119917263235295
```

[141]:
```
# Calculate R-squared scores
train_r2 = metrics.r2_score(y_train, pipe.predict(X_train))
test_r2 = metrics.r2_score(y_test, y_pred_DT)

# Print the results
print(f'Training R-squared: {train_r2}')
print(f'Test R-squared: {test_r2}')
```

```
Training R-squared: 0.9161776318199706
Test R-squared: 0.7877429487304191
```

- Its a clear case of overfitting

[142]:
```
# Random Forest Regression

# columnTransformer will will use to apply different trasformation to different␣
 ↪columns

step1 = ColumnTransformer(transformers = [('col_tnf',
                                           OneHotEncoder(sparse_output = False,␣
 ↪drop = 'first'),
                                           [0,1,3,8,11])],
                          remainder = 'passthrough')
```

```
step2 =␣
 ↪RandomForestRegressor(n_estimators=100,random_state=3,max_depth=15,max_samples=0.
 ↪5,max_features=0.75)

pipe = Pipeline([
    ('step1',step1),
    ('step2',step2)
])

pipe.fit(X_train, y_train)

y_pred_RF = pipe.predict(X_test)

print('R2_sore: ', metrics.r2_score(y_test,y_pred_RF))
print('MAE: ', metrics.mean_absolute_error(y_test,y_pred_RF))
print('MSE: ',metrics.mean_squared_error(y_test,y_pred_RF))

# Calculate R-squared scores
train_r2 = metrics.r2_score(y_train, pipe.predict(X_train))
test_r2 = metrics.r2_score(y_test, y_pred_RF)

# Print the results
print(f'Training R-squared: {train_r2}')
print(f'Test R-squared: {test_r2}')
```

```
R2_sore:  0.8848024216171648
MAE:  0.16024231797483873
MSE:  0.044068962599771035
Training R-squared: 0.9521322934303903
Test R-squared: 0.8848024216171648
```

- Random Forest Tree has given so far better result so will do some hyperparameter tunning and try to improve the performance of the model.

[143]:
```
# Hyperparameter Tunning for Random Forest Regressor

X.head()
```

[143]:

	Company	TypeName	Ram_GB	OpSys	Weight_KG	TouchScreen	IPS	PPI	\
0	Apple	Ultrabook	8	MaC	1.37	0	1	226.992481	
1	Apple	Ultrabook	8	MaC	1.34	0	0	127.669173	
2	HP	Notebook	8	Others	1.86	0	0	141.217949	
3	Apple	Ultrabook	16	MaC	1.83	0	1	220.519481	
4	Apple	Ultrabook	8	MaC	1.37	0	1	226.992481	

	Cpu_name	HDD	SSD	Gpu_brand
0	Intel Core i5	0	128	Intel
1	Intel Core i5	0	0	Intel

```
2  Intel Core i5    0  256     Intel
3  Intel Core i7    0  512       AMD
4  Intel Core i5    0  256     Intel
```

[144]:
```
X.shape
```

[144]:
```
(1302, 12)
```

[145]:
```
X_new = X.copy()
```

[159]:
```
X_new = pd.get_dummies(X_new, drop_first = True)
```

[160]:
```
X_new
```

[160]:
```
      Ram_GB  Weight_KG  TouchScreen  IPS         PPI   HDD  SSD  \
0          8       1.37            0    1  226.992481     0  128
1          8       1.34            0    0  127.669173     0    0
2          8       1.86            0    0  141.217949     0  256
3         16       1.83            0    1  220.519481     0  512
4          8       1.37            0    1  226.992481     0  256
…        …         …          …   …          …    …  …
1298       4       1.80            1    1  157.357143     0  128
1299      16       1.30            1    1  276.090226     0  512
1300       2       1.50            0    0  111.928571     0    0
1301       6       2.19            0    0  100.448718  1000    0
1302       4       2.20            0    0  100.448718   500    0

      Company_Apple  Company_Asus  Company_Chuwi  …  TypeName_Ultrabook  \
0              True         False          False  …                True
1              True         False          False  …                True
2             False         False          False  …               False
3              True         False          False  …                True
4              True         False          False  …                True
…             …           …            …  …                …
1298          False         False          False  …               False
1299          False         False          False  …               False
1300          False         False          False  …               False
1301          False         False          False  …               False
1302          False          True          False  …               False

      TypeName_Workstation  OpSys_Others  OpSys_Windows  \
0                    False         False          False
1                    False         False          False
2                    False          True          False
3                    False         False          False
4                    False         False          False
…                    …           …            …
```

```
1298                False         False           True
1299                False         False           True
1300                False         False           True
1301                False         False           True
1302                False         False           True

      Cpu_name_Intel Core i3  Cpu_name_Intel Core i5  Cpu_name_Intel Core i7  \
0                      False                    True                   False
1                      False                    True                   False
2                      False                    True                   False
3                      False                   False                    True
4                      False                    True                   False
…                      …                        …                      …
1298                   False                   False                    True
1299                   False                   False                    True
1300                   False                   False                   False
1301                   False                   False                    True
1302                   False                   False                   False

      Cpu_name_Other Intel Processor  Gpu_brand_Intel  Gpu_brand_Nvidia
0                              False             True             False
1                              False             True             False
2                              False             True             False
3                              False            False             False
4                              False             True             False
…                              …                 …                …
1298                           False             True             False
1299                           False             True             False
1300                            True             True             False
1301                           False            False             False
1302                            True             True             False

[1302 rows x 38 columns]
```

[161]:
```
X_new.columns
```

[161]:
```
Index(['Ram_GB', 'Weight_KG', 'TouchScreen', 'IPS', 'PPI', 'HDD', 'SSD',
       'Company_Apple', 'Company_Asus', 'Company_Chuwi', 'Company_Dell',
       'Company_Fujitsu', 'Company_Google', 'Company_HP', 'Company_Huawei',
       'Company_LG', 'Company_Lenovo', 'Company_MSI', 'Company_Mediacom',
       'Company_Microsoft', 'Company_Razer', 'Company_Samsung',
       'Company_Toshiba', 'Company_Vero', 'Company_Xiaomi', 'TypeName_Gaming',
       'TypeName_Netbook', 'TypeName_Notebook', 'TypeName_Ultrabook',
       'TypeName_Workstation', 'OpSys_Others', 'OpSys_Windows',
       'Cpu_name_Intel Core i3', 'Cpu_name_Intel Core i5',
       'Cpu_name_Intel Core i7', 'Cpu_name_Other Intel Processor',
       'Gpu_brand_Intel', 'Gpu_brand_Nvidia'],
```

```
      dtype='object')
```

[162]:
```
indexList = [0,1,3,8,11]
tranformList = []

for k,v in mapper.items():
    if k in indexList:
        tranformList.append(v)

tranformList
```

[162]:
```
['Company', 'TypeName', 'OpSys', 'Cpu_name', 'Gpu_brand']
```

[163]:
```
train_new = pd.get_dummies(X.copy(), columns = tranformList, drop_first = True)
train_new.head()
```

[163]:

	Ram_GB	Weight_KG	TouchScreen	IPS	PPI	HDD	SSD	Company_Apple	\
0	8	1.37	0	1	226.992481	0	128	True	
1	8	1.34	0	0	127.669173	0	0	True	
2	8	1.86	0	0	141.217949	0	256	False	
3	16	1.83	0	1	220.519481	0	512	True	
4	8	1.37	0	1	226.992481	0	256	True	

	Company_Asus	Company_Chuwi	…	TypeName_Ultrabook	TypeName_Workstation	\
0	False	False	…	True	False	
1	False	False	…	True	False	
2	False	False	…	False	False	
3	False	False	…	True	False	
4	False	False	…	True	False	

	OpSys_Others	OpSys_Windows	Cpu_name_Intel Core i3	\
0	False	False	False	
1	False	False	False	
2	True	False	False	
3	False	False	False	
4	False	False	False	

	Cpu_name_Intel Core i5	Cpu_name_Intel Core i7	\
0	True	False	
1	True	False	
2	True	False	
3	False	True	
4	True	False	

	Cpu_name_Other Intel Processor	Gpu_brand_Intel	Gpu_brand_Nvidia
0	False	True	False
1	False	True	False

```
2                          False            True            False
3                          False           False            False
4                          False            True            False

[5 rows x 38 columns]
```

[164]:
```
train_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1302 entries, 0 to 1302
Data columns (total 38 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Ram_GB                          1302 non-null   int32
 1   Weight_KG                       1302 non-null   float32
 2   TouchScreen                     1302 non-null   int64
 3   IPS                             1302 non-null   int64
 4   PPI                             1302 non-null   float64
 5   HDD                             1302 non-null   int64
 6   SSD                             1302 non-null   int64
 7   Company_Apple                   1302 non-null   bool
 8   Company_Asus                    1302 non-null   bool
 9   Company_Chuwi                   1302 non-null   bool
 10  Company_Dell                    1302 non-null   bool
 11  Company_Fujitsu                 1302 non-null   bool
 12  Company_Google                  1302 non-null   bool
 13  Company_HP                      1302 non-null   bool
 14  Company_Huawei                  1302 non-null   bool
 15  Company_LG                      1302 non-null   bool
 16  Company_Lenovo                  1302 non-null   bool
 17  Company_MSI                     1302 non-null   bool
 18  Company_Mediacom                1302 non-null   bool
 19  Company_Microsoft               1302 non-null   bool
 20  Company_Razer                   1302 non-null   bool
 21  Company_Samsung                 1302 non-null   bool
 22  Company_Toshiba                 1302 non-null   bool
 23  Company_Vero                    1302 non-null   bool
 24  Company_Xiaomi                  1302 non-null   bool
 25  TypeName_Gaming                 1302 non-null   bool
 26  TypeName_Netbook                1302 non-null   bool
 27  TypeName_Notebook               1302 non-null   bool
 28  TypeName_Ultrabook              1302 non-null   bool
 29  TypeName_Workstation            1302 non-null   bool
 30  OpSys_Others                    1302 non-null   bool
 31  OpSys_Windows                   1302 non-null   bool
 32  Cpu_name_Intel Core i3          1302 non-null   bool
 33  Cpu_name_Intel Core i5          1302 non-null   bool
```

```
 34  Cpu_name_Intel Core i7          1302 non-null   bool
 35  Cpu_name_Other Intel Processor  1302 non-null   bool
 36  Gpu_brand_Intel                 1302 non-null   bool
 37  Gpu_brand_Nvidia                1302 non-null   bool
dtypes: bool(31), float32(1), float64(1), int32(1), int64(4)
memory usage: 110.6 KB
```

[165]:
```
# Converting all bool to numerical

train_new['Company_Apple'] = train_new['Company_Apple'].replace({True: 1, False:
 ↪ 0})
train_new['Company_Asus'] = train_new['Company_Asus'].replace({True: 1, False:␣
 ↪0})
train_new['Company_Chuwi'] = train_new['Company_Chuwi'].replace({True: 1, False:
 ↪ 0})
train_new['Company_Dell'] = train_new['Company_Dell'].replace({True: 1, False:␣
 ↪0})
train_new['Company_Fujitsu'] = train_new['Company_Fujitsu'].replace({True: 1,␣
 ↪False: 0})
train_new['Company_Google'] = train_new['Company_Google'].replace({True: 1,␣
 ↪False: 0})
train_new['Company_HP'] = train_new['Company_HP'].replace({True: 1, False: 0})
train_new['Company_Huawei'] = train_new['Company_Huawei'].replace({True: 1,␣
 ↪False: 0})
train_new['Company_LG'] = train_new['Company_LG'].replace({True: 1, False: 0})
train_new['Company_Lenovo'] = train_new['Company_Lenovo'].replace({True: 1,␣
 ↪False: 0})
train_new['Company_MSI'] = train_new['Company_MSI'].replace({True: 1, False: 0})
train_new['Company_Mediacom'] = train_new['Company_Mediacom'].replace({True: 1,␣
 ↪False: 0})
train_new['Company_Microsoft'] = train_new['Company_Microsoft'].replace({True:␣
 ↪1, False: 0})
train_new['Company_Razer'] = train_new['Company_Razer'].replace({True: 1, False:
 ↪ 0})
train_new['Company_Samsung'] = train_new['Company_Samsung'].replace({True: 1,␣
 ↪False: 0})
train_new['Company_Toshiba'] = train_new['Company_Toshiba'].replace({True: 1,␣
 ↪False: 0})
train_new['Company_Vero'] = train_new['Company_Vero'].replace({True: 1, False:␣
 ↪0})
train_new['Company_Xiaomi'] = train_new['Company_Xiaomi'].replace({True: 1,␣
 ↪False: 0})
train_new['TypeName_Gaming'] = train_new['TypeName_Gaming'].replace({True: 1,␣
 ↪False: 0})
train_new['TypeName_Netbook'] = train_new['TypeName_Netbook'].replace({True: 1,␣
 ↪False: 0})
```

```
train_new['TypeName_Notebook'] = train_new['TypeName_Notebook'].replace({True:␣
 ↪1, False: 0})
train_new['TypeName_Ultrabook'] = train_new['TypeName_Ultrabook'].replace({True:
 ↪ 1, False: 0})
train_new['TypeName_Workstation'] = train_new['TypeName_Workstation'].
 ↪replace({True: 1, False: 0})
train_new['OpSys_Others'] = train_new['OpSys_Others'].replace({True: 1, False:␣
 ↪0})
train_new['OpSys_Windows'] = train_new['OpSys_Windows'].replace({True: 1, False:
 ↪ 0})
train_new['Cpu_name_Intel Core i3'] = train_new['Cpu_name_Intel Core i3'].
 ↪replace({True: 1, False: 0})
train_new['Cpu_name_Intel Core i5'] = train_new['Cpu_name_Intel Core i5'].
 ↪replace({True: 1, False: 0})
train_new['Cpu_name_Intel Core i7'] = train_new['Cpu_name_Intel Core i7'].
 ↪replace({True: 1, False: 0})
train_new['Cpu_name_Other Intel Processor'] = train_new['Cpu_name_Other Intel␣
 ↪Processor'].replace({True: 1, False: 0})
train_new['Gpu_brand_Nvidia'] = train_new['Gpu_brand_Nvidia'].replace({True: 1,␣
 ↪False: 0})
train_new['Gpu_brand_Intel'] = train_new['Gpu_brand_Intel'].replace({True: 1,␣
 ↪False: 0})
```

[166]:
```
train_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1302 entries, 0 to 1302
Data columns (total 38 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Ram_GB                          1302 non-null   int32
 1   Weight_KG                       1302 non-null   float32
 2   TouchScreen                     1302 non-null   int64
 3   IPS                             1302 non-null   int64
 4   PPI                             1302 non-null   float64
 5   HDD                             1302 non-null   int64
 6   SSD                             1302 non-null   int64
 7   Company_Apple                   1302 non-null   int64
 8   Company_Asus                    1302 non-null   int64
 9   Company_Chuwi                   1302 non-null   int64
 10  Company_Dell                    1302 non-null   int64
 11  Company_Fujitsu                 1302 non-null   int64
 12  Company_Google                  1302 non-null   int64
 13  Company_HP                      1302 non-null   int64
 14  Company_Huawei                  1302 non-null   int64
 15  Company_LG                      1302 non-null   int64
 16  Company_Lenovo                  1302 non-null   int64
```

```
 17  Company_MSI                     1302 non-null   int64
 18  Company_Mediacom                1302 non-null   int64
 19  Company_Microsoft               1302 non-null   int64
 20  Company_Razer                   1302 non-null   int64
 21  Company_Samsung                 1302 non-null   int64
 22  Company_Toshiba                 1302 non-null   int64
 23  Company_Vero                    1302 non-null   int64
 24  Company_Xiaomi                  1302 non-null   int64
 25  TypeName_Gaming                 1302 non-null   int64
 26  TypeName_Netbook                1302 non-null   int64
 27  TypeName_Notebook               1302 non-null   int64
 28  TypeName_Ultrabook              1302 non-null   int64
 29  TypeName_Workstation            1302 non-null   int64
 30  OpSys_Others                    1302 non-null   int64
 31  OpSys_Windows                   1302 non-null   int64
 32  Cpu_name_Intel Core i3          1302 non-null   int64
 33  Cpu_name_Intel Core i5          1302 non-null   int64
 34  Cpu_name_Intel Core i7          1302 non-null   int64
 35  Cpu_name_Other Intel Processor  1302 non-null   int64
 36  Gpu_brand_Intel                 1302 non-null   int64
 37  Gpu_brand_Nvidia                1302 non-null   int64
dtypes: float32(1), float64(1), int32(1), int64(35)
memory usage: 386.5 KB
```

[167]:
```
X_train_new,X_test_new, y_train, y_test = train_test_split(X_new, y, test_size
 ↪= 0.15, random_state = 2)

X_train_new.shape, X_test_new.shape, y_train.shape, y_test.shape
```

[167]:
```
((1106, 38), (196, 38), (1106,), (196,))
```

[168]:
```
DTReg  = DecisionTreeRegressor(random_state=0)
DTReg.fit(X_train_new,y_train)

plt.figure(figsize=(16,9))
tree.plot_tree(DTReg,filled=True,feature_names=X_train_new.columns)
```

[168]:
```
[Text(0.48811201891965883, 0.9782608695652174, 'Ram_GB <= 7.0\nsquared_error =
0.39\nsamples = 1106\nvalue = 6.846'),
 Text(0.2178686502372735, 0.9347826086956522, 'Cpu_name_Other Intel Processor <=
0.5\nsquared_error = 0.192\nsamples = 362\nvalue = 6.234'),
 Text(0.1450868412756355, 0.8913043478260869, 'Cpu_name_Intel Core i5 <=
0.5\nsquared_error = 0.13\nsamples = 256\nvalue = 6.411'),
 Text(0.09647221576956262, 0.8478260869565217, 'Cpu_name_Intel Core i7 <=
0.5\nsquared_error = 0.08\nsamples = 154\nvalue = 6.257'),
 Text(0.03936375854516493, 0.8043478260869565, 'Weight_KG <=
1.645\nsquared_error = 0.07\nsamples = 135\nvalue = 6.208'),
```

```
 Text(0.0074334638614189956, 0.7608695652173914, 'TypeName_Ultrabook <=
0.5\nsquared_error = 0.064\nsamples = 10\nvalue = 6.611'),
 Text(0.005309617043870711, 0.717391304347826, 'Weight_KG <=
1.395\nsquared_error = 0.02\nsamples = 8\nvalue = 6.502'),
 Text(0.0042476936350965685, 0.6739130434782609, 'squared_error = 0.0\nsamples =
1\nvalue = 6.232'),
 Text(0.006371540452644853, 0.6739130434782609, 'Company_Lenovo <=
0.5\nsquared_error = 0.011\nsamples = 7\nvalue = 6.541'),
 Text(0.005309617043870711, 0.6304347826086957, 'Weight_KG <=
1.615\nsquared_error = 0.003\nsamples = 6\nvalue = 6.579'),
 Text(0.0031857702263224266, 0.5869565217391305, 'HDD <= 250.0\nsquared_error =
0.002\nsamples = 3\nvalue = 6.621'),
 Text(0.0021238468175482843, 0.5434782608695652, 'PPI <= 161.498\nsquared_error
= 0.0\nsamples = 2\nvalue = 6.589'),
 Text(0.0010619234087741421, 0.5, 'squared_error = 0.0\nsamples = 1\nvalue =
6.601'),
 Text(0.0031857702263224266, 0.5, 'squared_error = 0.0\nsamples = 1\nvalue =
6.578'),
 Text(0.0042476936350965685, 0.5434782608695652, 'squared_error = -0.0\nsamples
= 1\nvalue = 6.685'),
 Text(0.0074334638614189956, 0.5869565217391305, 'Gpu_brand_Nvidia <=
0.5\nsquared_error = 0.0\nsamples = 3\nvalue = 6.538'),
 Text(0.006371540452644853, 0.5434782608695652, 'PPI <= 106.189\nsquared_error =
0.0\nsamples = 2\nvalue = 6.532'),
 Text(0.005309617043870711, 0.5, 'squared_error = 0.0\nsamples = 1\nvalue =
6.535'),
 Text(0.0074334638614189956, 0.5, 'squared_error = 0.0\nsamples = 1\nvalue =
6.528'),
 Text(0.008495387270193137, 0.5434782608695652, 'squared_error = 0.0\nsamples =
1\nvalue = 6.55'),
 Text(0.0074334638614189956, 0.6304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 6.308'),
 Text(0.00955731067896728, 0.717391304347826, 'SSD <= 64.0\nsquared_error =
0.0\nsamples = 2\nvalue = 7.048'),
 Text(0.008495387270193137, 0.6739130434782609, 'squared_error = 0.0\nsamples =
1\nvalue = 7.035'),
 Text(0.010619234087741422, 0.6739130434782609, 'squared_error = -0.0\nsamples =
1\nvalue = 7.062'),
 Text(0.07129405322891086, 0.7608695652173914, 'PPI <= 114.874\nsquared_error =
0.056\nsamples = 125\nvalue = 6.175'),
 Text(0.03448762195526647, 0.717391304347826, 'Cpu_name_Intel Core i3 <=
0.5\nsquared_error = 0.057\nsamples = 63\nvalue = 6.084'),
 Text(0.018318178801353953, 0.6739130434782609, 'Weight_KG <=
1.665\nsquared_error = 0.049\nsamples = 18\nvalue = 5.878'),
 Text(0.01725625539257981, 0.6304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 5.293'),
 Text(0.019380102210128095, 0.6304347826086957, 'Company_HP <=
```

```
0.5\nsquared_error = 0.03\nsamples = 17\nvalue = 5.913'),
 Text(0.01539788942722506, 0.5869565217391305, 'Company_Lenovo <=
0.5\nsquared_error = 0.022\nsamples = 13\nvalue = 5.966'),
 Text(0.010619234087741422, 0.5434782608695652, 'Weight_KG <=
1.84\nsquared_error = 0.007\nsamples = 9\nvalue = 6.013'),
 Text(0.00955731067896728, 0.5, 'squared_error = 0.0\nsamples = 1\nvalue =
6.151'),
 Text(0.011681157496515564, 0.5, 'Weight_KG <= 2.015\nsquared_error =
0.005\nsamples = 8\nvalue = 5.995'),
 Text(0.010619234087741422, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 5.855'),
 Text(0.012743080905289706, 0.45652173913043476, 'HDD <= 250.0\nsquared_error =
0.002\nsamples = 7\nvalue = 6.015'),
 Text(0.010619234087741422, 0.41304347826086957, 'SSD <= 192.0\nsquared_error =
0.001\nsamples = 2\nvalue = 6.083'),
 Text(0.00955731067896728, 0.3695652173913043, 'squared_error = 0.0\nsamples =
1\nvalue = 6.054'),
 Text(0.011681157496515564, 0.3695652173913043, 'squared_error = -0.0\nsamples =
1\nvalue = 6.111'),
 Text(0.014866927722837991, 0.41304347826086957, 'Weight_KG <=
2.25\nsquared_error = 0.0\nsamples = 5\nvalue = 5.988'),
 Text(0.013805004314063849, 0.3695652173913043, 'Ram_GB <= 5.0\nsquared_error =
0.0\nsamples = 4\nvalue = 5.996'),
 Text(0.012743080905289706, 0.32608695652173914, 'Weight_KG <=
2.065\nsquared_error = 0.0\nsamples = 3\nvalue = 5.99'),
 Text(0.011681157496515564, 0.2826086956521739, 'squared_error = 0.0\nsamples =
1\nvalue = 5.999'),
 Text(0.013805004314063849, 0.2826086956521739, 'HDD <= 750.0\nsquared_error =
0.0\nsamples = 2\nvalue = 5.985'),
 Text(0.012743080905289706, 0.2391304347826087, 'squared_error = 0.0\nsamples =
1\nvalue = 5.991'),
 Text(0.014866927722837991, 0.2391304347826087, 'squared_error = 0.0\nsamples =
1\nvalue = 5.979'),
 Text(0.014866927722837991, 0.32608695652173914, 'squared_error = -0.0\nsamples
= 1\nvalue = 6.014'),
 Text(0.01592885113161213, 0.3695652173913043, 'squared_error = 0.0\nsamples =
1\nvalue = 5.958'),
 Text(0.0201765447667087, 0.5434782608695652, 'PPI <= 103.288\nsquared_error =
0.041\nsamples = 4\nvalue = 5.861'),
 Text(0.01911462135793456, 0.5, 'SSD <= 64.0\nsquared_error = 0.005\nsamples =
3\nvalue = 5.751'),
 Text(0.018052697949160416, 0.45652173913043476, 'Weight_KG <=
2.195\nsquared_error = 0.0\nsamples = 2\nvalue = 5.699'),
 Text(0.016990774540386274, 0.41304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 5.697'),
 Text(0.01911462135793456, 0.41304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 5.7'),
```

```
 Text(0.0201765447667087, 0.45652173913043476, 'squared_error = -0.0\nsamples =
1\nvalue = 5.855'),
 Text(0.021238468175482843, 0.5, 'squared_error = -0.0\nsamples = 1\nvalue =
6.192'),
 Text(0.023362314993031128, 0.5869565217391305, 'OpSys_Windows <=
0.5\nsquared_error = 0.017\nsamples = 4\nvalue = 5.739'),
 Text(0.022300391584256986, 0.5434782608695652, 'squared_error = 0.0\nsamples =
1\nvalue = 5.553'),
 Text(0.02442423840180527, 0.5434782608695652, 'HDD <= 250.0\nsquared_error =
0.008\nsamples = 3\nvalue = 5.801'),
 Text(0.023362314993031128, 0.5, 'squared_error = 0.0\nsamples = 1\nvalue =
5.911'),
 Text(0.025486161810579413, 0.5, 'Weight_KG <= 1.98\nsquared_error =
0.003\nsamples = 2\nvalue = 5.746'),
 Text(0.02442423840180527, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 5.694'),
 Text(0.026548085219353555, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 5.799'),
 Text(0.050657065109179, 0.6739130434782609, 'OpSys_Windows <=
0.5\nsquared_error = 0.036\nsamples = 45\nvalue = 6.166'),
 Text(0.036636357602707906, 0.6304347826086957, 'Company_Dell <=
0.5\nsquared_error = 0.012\nsamples = 12\nvalue = 5.976'),
 Text(0.03291962567199841, 0.5869565217391305, 'HDD <= 750.0\nsquared_error =
0.008\nsamples = 8\nvalue = 5.934'),
 Text(0.03079577885445012, 0.5434782608695652, 'Weight_KG <= 2.25\nsquared_error
= 0.008\nsamples = 3\nvalue = 6.007'),
 Text(0.029733855445675982, 0.5, 'HDD <= 250.0\nsquared_error = 0.002\nsamples =
2\nvalue = 6.065'),
 Text(0.02867193203690184, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 6.021'),
 Text(0.03079577885445012, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 6.109'),
 Text(0.03185770226322426, 0.5, 'squared_error = 0.0\nsamples = 1\nvalue =
5.891'),
 Text(0.035043472489546694, 0.5434782608695652, 'Weight_KG <= 2.0\nsquared_error
= 0.002\nsamples = 5\nvalue = 5.89'),
 Text(0.03398154908077255, 0.5, 'Gpu_brand_Intel <= 0.5\nsquared_error =
0.001\nsamples = 3\nvalue = 5.86'),
 Text(0.03291962567199841, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 5.908'),
 Text(0.035043472489546694, 0.45652173913043476, 'Company_HP <=
0.5\nsquared_error = 0.0\nsamples = 2\nvalue = 5.836'),
 Text(0.03398154908077255, 0.41304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 5.826'),
 Text(0.03610539589832083, 0.41304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 5.846'),
 Text(0.03610539589832083, 0.5, 'squared_error = 0.001\nsamples = 2\nvalue =
```

```
5.936'),
 Text(0.0403530895334174, 0.5869565217391305, 'Weight_KG <= 2.215\nsquared_error
= 0.011\nsamples = 4\nvalue = 6.061'),
 Text(0.03929116612464326, 0.5434782608695652, 'Gpu_brand_Intel <=
0.5\nsquared_error = 0.0\nsamples = 3\nvalue = 6.12'),
 Text(0.03822924271586912, 0.5, 'squared_error = 0.0\nsamples = 2\nvalue =
6.132'),
 Text(0.0403530895334174, 0.5, 'squared_error = -0.0\nsamples = 1\nvalue =
6.096'),
 Text(0.04141501294219154, 0.5434782608695652, 'squared_error = 0.0\nsamples =
1\nvalue = 5.883'),
 Text(0.0646777726156501, 0.6304347826086957, 'Company_HP <= 0.5\nsquared_error
= 0.027\nsamples = 33\nvalue = 6.235'),
 Text(0.05502090661711024, 0.5869565217391305, 'Gpu_brand_Intel <=
0.5\nsquared_error = 0.022\nsamples = 29\nvalue = 6.212'),
 Text(0.045131744872901045, 0.5434782608695652, 'HDD <= 750.0\nsquared_error =
0.013\nsamples = 7\nvalue = 6.342'),
 Text(0.04247693635096569, 0.5, 'HDD <= 250.0\nsquared_error = 0.001\nsamples =
2\nvalue = 6.182'),
 Text(0.04141501294219154, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 6.213'),
 Text(0.043538859759739826, 0.45652173913043476, 'squared_error = -0.0\nsamples
= 1\nvalue = 6.151'),
 Text(0.047786553394836395, 0.5, 'PPI <= 103.288\nsquared_error = 0.004\nsamples
= 5\nvalue = 6.407'),
 Text(0.04566270657728811, 0.45652173913043476, 'Weight_KG <=
2.19\nsquared_error = 0.003\nsamples = 3\nvalue = 6.383'),
 Text(0.04460078316851397, 0.41304347826086957, 'squared_error = 0.0\nsamples =
2\nvalue = 6.423'),
 Text(0.046724629986062256, 0.41304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 6.302'),
 Text(0.04991040021238468, 0.45652173913043476, 'Gpu_brand_Nvidia <=
0.5\nsquared_error = 0.002\nsamples = 2\nvalue = 6.442'),
 Text(0.04884847680361054, 0.41304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 6.491'),
 Text(0.050972323621158826, 0.41304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 6.394'),
 Text(0.06491006836131943, 0.5434782608695652, 'Weight_KG <=
2.295\nsquared_error = 0.017\nsamples = 22\nvalue = 6.17'),
 Text(0.059733191743545494, 0.5, 'Company_Dell <= 0.5\nsquared_error =
0.017\nsamples = 14\nvalue = 6.221'),
 Text(0.05575097896064246, 0.45652173913043476, 'Company_Toshiba <=
0.5\nsquared_error = 0.018\nsamples = 8\nvalue = 6.168'),
 Text(0.05309617043870711, 0.41304347826086957, 'HDD <= 250.0\nsquared_error =
0.002\nsamples = 5\nvalue = 6.108'),
 Text(0.050972323621158826, 0.3695652173913043, 'Company_Lenovo <=
0.5\nsquared_error = 0.001\nsamples = 2\nvalue = 6.084'),
```

```
 Text(0.04991040021238468, 0.32608695652173914, 'squared_error = 0.0\nsamples =
1\nvalue = 6.107'),
 Text(0.052034247029932965, 0.32608695652173914, 'squared_error = 0.0\nsamples =
1\nvalue = 6.061'),
 Text(0.055220017256255395, 0.3695652173913043, 'Company_Lenovo <=
0.5\nsquared_error = 0.003\nsamples = 3\nvalue = 6.124'),
 Text(0.05415809384748125, 0.32608695652173914, 'Weight_KG <=
2.05\nsquared_error = 0.0\nsamples = 2\nvalue = 6.089'),
 Text(0.05309617043870711, 0.2826086956521739, 'squared_error = 0.0\nsamples =
1\nvalue = 6.075'),
 Text(0.055220017256255395, 0.2826086956521739, 'squared_error = 0.0\nsamples =
1\nvalue = 6.103'),
 Text(0.056281940665029534, 0.32608695652173914, 'squared_error = -0.0\nsamples
= 1\nvalue = 6.194'),
 Text(0.05840578748257782, 0.41304347826086957, 'Weight_KG <=
2.05\nsquared_error = 0.03\nsamples = 3\nvalue = 6.267'),
 Text(0.05734386407380368, 0.3695652173913043, 'squared_error = 0.025\nsamples =
2\nvalue = 6.349'),
 Text(0.059467710891351964, 0.3695652173913043, 'squared_error = -0.0\nsamples =
1\nvalue = 6.103'),
 Text(0.06371540452644853, 0.45652173913043476, 'Weight_KG <=
2.235\nsquared_error = 0.006\nsamples = 6\nvalue = 6.291'),
 Text(0.06265348111767438, 0.41304347826086957, 'HDD <= 750.0\nsquared_error =
0.005\nsamples = 5\nvalue = 6.311'),
 Text(0.06159155770890024, 0.3695652173913043, 'Weight_KG <= 1.91\nsquared_error
= 0.006\nsamples = 3\nvalue = 6.285'),
 Text(0.0605296343001261, 0.32608695652173914, 'squared_error = 0.0\nsamples =
1\nvalue = 6.372'),
 Text(0.06265348111767438, 0.32608695652173914, 'SSD <= 64.0\nsquared_error =
0.004\nsamples = 2\nvalue = 6.241'),
 Text(0.06159155770890024, 0.2826086956521739, 'squared_error = 0.0\nsamples =
1\nvalue = 6.178'),
 Text(0.06371540452644853, 0.2826086956521739, 'squared_error = 0.0\nsamples =
1\nvalue = 6.304'),
 Text(0.06371540452644853, 0.3695652173913043, 'squared_error = 0.001\nsamples =
2\nvalue = 6.35'),
 Text(0.06477732793522267, 0.41304347826086957, 'squared_error = -0.0\nsamples =
1\nvalue = 6.194'),
 Text(0.07008694497909339, 0.5, 'Weight_KG <= 2.6\nsquared_error =
0.005\nsamples = 8\nvalue = 6.081'),
 Text(0.06902502157031924, 0.45652173913043476, 'Ram_GB <= 5.0\nsquared_error =
0.004\nsamples = 7\nvalue = 6.065'),
 Text(0.06690117475277095, 0.41304347826086957, 'HDD <= 750.0\nsquared_error =
0.003\nsamples = 4\nvalue = 6.029'),
 Text(0.06583925134399682, 0.3695652173913043, 'squared_error = 0.002\nsamples =
2\nvalue = 6.064'),
 Text(0.0679630981615451, 0.3695652173913043, 'Company_Dell <=
```

```
0.5\nsquared_error = 0.002\nsamples = 2\nvalue = 5.994'),
 Text(0.06690117475277095, 0.32608695652173914, 'squared_error = 0.0\nsamples =
1\nvalue = 5.951'),
 Text(0.06902502157031924, 0.32608695652173914, 'squared_error = -0.0\nsamples =
1\nvalue = 6.037'),
 Text(0.07114886838786752, 0.41304347826086957, 'TouchScreen <=
0.5\nsquared_error = 0.0\nsamples = 3\nvalue = 6.114'),
 Text(0.07008694497909339, 0.3695652173913043, 'squared_error = 0.0\nsamples =
2\nvalue = 6.129'),
 Text(0.07221079179664167, 0.3695652173913043, 'squared_error = 0.0\nsamples =
1\nvalue = 6.084'),
 Text(0.07114886838786752, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 6.192'),
 Text(0.07433463861418996, 0.5869565217391305, 'Weight_KG <=
1.965\nsquared_error = 0.035\nsamples = 4\nvalue = 6.406'),
 Text(0.07327271520541581, 0.5434782608695652, 'squared_error = 0.0\nsamples =
1\nvalue = 6.084'),
 Text(0.07539656202296409, 0.5434782608695652, 'Weight_KG <=
2.205\nsquared_error = 0.001\nsamples = 3\nvalue = 6.513'),
 Text(0.07433463861418996, 0.5, 'Weight_KG <= 2.085\nsquared_error =
0.0\nsamples = 2\nvalue = 6.491'),
 Text(0.07327271520541581, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 6.477'),
 Text(0.07539656202296409, 0.45652173913043476, 'squared_error = -0.0\nsamples =
1\nvalue = 6.505'),
 Text(0.07645848543173824, 0.5, 'squared_error = -0.0\nsamples = 1\nvalue =
6.558'),
 Text(0.10810048450255526, 0.717391304347826, 'IPS <= 0.5\nsquared_error =
0.038\nsamples = 62\nvalue = 6.268'),
 Text(0.08584987057808456, 0.6739130434782609, 'Weight_KG <= 1.75\nsquared_error
= 0.034\nsamples = 50\nvalue = 6.225'),
 Text(0.08017521736244773, 0.6304347826086957, 'SSD <= 64.0\nsquared_error =
0.011\nsamples = 2\nvalue = 6.64'),
 Text(0.07911329395367359, 0.5869565217391305, 'squared_error = 0.0\nsamples =
1\nvalue = 6.744'),
 Text(0.08123714077122188, 0.5869565217391305, 'squared_error = -0.0\nsamples =
1\nvalue = 6.535'),
 Text(0.09152452379372138, 0.6304347826086957, 'Weight_KG <=
1.885\nsquared_error = 0.028\nsamples = 48\nvalue = 6.208'),
 Text(0.08336098758877016, 0.5869565217391305, 'HDD <= 750.0\nsquared_error =
0.008\nsamples = 6\nvalue = 5.999'),
 Text(0.0807061790668348, 0.5434782608695652, 'SSD <= 64.0\nsquared_error =
0.004\nsamples = 4\nvalue = 5.954'),
 Text(0.07858233224928653, 0.5, 'OpSys_Windows <= 0.5\nsquared_error =
0.005\nsamples = 2\nvalue = 5.914'),
 Text(0.07752040884051238, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 5.844'),
```

```
 Text(0.07964425565806066, 0.45652173913043476, 'squared_error = -0.0\nsamples =
1\nvalue = 5.984'),
 Text(0.08283002588438308, 0.5, 'Company_HP <= 0.5\nsquared_error = 0.0\nsamples
= 2\nvalue = 5.994'),
 Text(0.08176810247560895, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 6.0'),
 Text(0.08389194929315723, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 5.988'),
 Text(0.08601579611070552, 0.5434782608695652, 'Weight_KG <=
1.855\nsquared_error = 0.004\nsamples = 2\nvalue = 6.089'),
 Text(0.08495387270193137, 0.5, 'squared_error = 0.0\nsamples = 1\nvalue =
6.028'),
 Text(0.08707771951947965, 0.5, 'squared_error = 0.0\nsamples = 1\nvalue =
6.151'),
 Text(0.0996880599986726, 0.5869565217391305, 'PPI <= 152.112\nsquared_error =
0.023\nsamples = 42\nvalue = 6.238'),
 Text(0.09862613658989845, 0.5434782608695652, 'OpSys_Windows <=
0.5\nsquared_error = 0.02\nsamples = 41\nvalue = 6.228'),
 Text(0.08920156633702794, 0.5, 'Company_Asus <= 0.5\nsquared_error =
0.008\nsamples = 8\nvalue = 6.116'),
 Text(0.08707771951947965, 0.45652173913043476, 'PPI <= 134.279\nsquared_error =
0.005\nsamples = 6\nvalue = 6.155'),
 Text(0.08601579611070552, 0.41304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 6.271'),
 Text(0.0881396429282538, 0.41304347826086957, 'Weight_KG <=
2.225\nsquared_error = 0.002\nsamples = 5\nvalue = 6.132'),
 Text(0.08707771951947965, 0.3695652173913043, 'Gpu_brand_Intel <=
0.5\nsquared_error = 0.001\nsamples = 4\nvalue = 6.151'),
 Text(0.08495387270193137, 0.32608695652173914, 'HDD <= 500.0\nsquared_error =
0.0\nsamples = 2\nvalue = 6.178'),
 Text(0.08389194929315723, 0.2826086956521739, 'squared_error = 0.0\nsamples =
1\nvalue = 6.184'),
 Text(0.08601579611070552, 0.2826086956521739, 'squared_error = 0.0\nsamples =
1\nvalue = 6.172'),
 Text(0.08920156633702794, 0.32608695652173914, 'SSD <= 128.0\nsquared_error =
0.001\nsamples = 2\nvalue = 6.123'),
 Text(0.0881396429282538, 0.2826086956521739, 'squared_error = 0.0\nsamples =
1\nvalue = 6.151'),
 Text(0.09026348974580209, 0.2826086956521739, 'squared_error = 0.0\nsamples =
1\nvalue = 6.096'),
 Text(0.08920156633702794, 0.3695652173913043, 'squared_error = -0.0\nsamples =
1\nvalue = 6.059'),
 Text(0.09132541315457622, 0.45652173913043476, 'SSD <= 128.0\nsquared_error =
0.001\nsamples = 2\nvalue = 5.996'),
 Text(0.09026348974580209, 0.41304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 5.964'),
 Text(0.09238733656335037, 0.41304347826086957, 'squared_error = -0.0\nsamples =
```

```
1\nvalue = 6.028'),
 Text(0.10805070684276896, 0.5, 'Weight_KG <= 2.065\nsquared_error =
0.019\nsamples = 33\nvalue = 6.255'),
 Text(0.09796243445941462, 0.45652173913043476, 'Weight_KG <=
1.955\nsquared_error = 0.013\nsamples = 11\nvalue = 6.307'),
 Text(0.09451118338089866, 0.41304347826086957, 'Gpu_brand_Intel <=
0.5\nsquared_error = 0.003\nsamples = 3\nvalue = 6.163'),
 Text(0.09344925997212451, 0.3695652173913043, 'TypeName_Ultrabook <=
0.5\nsquared_error = 0.0\nsamples = 2\nvalue = 6.202'),
 Text(0.09238733656335037, 0.32608695652173914, 'squared_error = 0.0\nsamples =
1\nvalue = 6.192'),
 Text(0.09451118338089866, 0.32608695652173914, 'squared_error = -0.0\nsamples =
1\nvalue = 6.213'),
 Text(0.09557310678967279, 0.3695652173913043, 'squared_error = 0.0\nsamples =
1\nvalue = 6.084'),
 Text(0.10141368553793058, 0.41304347826086957, 'Gpu_brand_Intel <=
0.5\nsquared_error = 0.006\nsamples = 8\nvalue = 6.361'),
 Text(0.09875887701599523, 0.3695652173913043, 'Company_Asus <=
0.5\nsquared_error = 0.0\nsamples = 6\nvalue = 6.319'),
 Text(0.09663503019844694, 0.32608695652173914, 'Cpu_name_Intel Core i3 <=
0.5\nsquared_error = 0.0\nsamples = 4\nvalue = 6.305'),
 Text(0.09557310678967279, 0.2826086956521739, 'squared_error = 0.0\nsamples =
3\nvalue = 6.31'),
 Text(0.09769695360722108, 0.2826086956521739, 'squared_error = 0.0\nsamples =
1\nvalue = 6.29'),
 Text(0.1008827238335435, 0.32608695652173914, 'PPI <= 134.279\nsquared_error =
0.0\nsamples = 2\nvalue = 6.346'),
 Text(0.09982080042476936, 0.2826086956521739, 'squared_error = 0.0\nsamples =
1\nvalue = 6.335'),
 Text(0.10194464724231765, 0.2826086956521739, 'squared_error = 0.0\nsamples =
1\nvalue = 6.358'),
 Text(0.10406849405986593, 0.3695652173913043, 'Weight_KG <=
2.045\nsquared_error = 0.002\nsamples = 2\nvalue = 6.488'),
 Text(0.10300657065109178, 0.32608695652173914, 'squared_error = 0.0\nsamples =
1\nvalue = 6.529'),
 Text(0.10513041746864007, 0.32608695652173914, 'squared_error = 0.0\nsamples =
1\nvalue = 6.446'),
 Text(0.11813897922612332, 0.45652173913043476, 'SSD <= 64.0\nsquared_error =
0.019\nsamples = 22\nvalue = 6.229'),
 Text(0.11309484303444614, 0.41304347826086957, 'Gpu_brand_Nvidia <=
0.5\nsquared_error = 0.011\nsamples = 13\nvalue = 6.178'),
 Text(0.10937811110373664, 0.3695652173913043, 'Ram_GB <= 5.0\nsquared_error =
0.008\nsamples = 9\nvalue = 6.136'),
 Text(0.10725426428618835, 0.32608695652173914, 'TypeName_Notebook <=
0.5\nsquared_error = 0.006\nsamples = 7\nvalue = 6.106'),
 Text(0.10619234087741422, 0.2826086956521739, 'squared_error = 0.0\nsamples =
1\nvalue = 6.232'),
```

```
 Text(0.1083161876949625, 0.2826086956521739, 'Cpu_name_Intel Core i3 <=
0.5\nsquared_error = 0.004\nsamples = 6\nvalue = 6.085'),
 Text(0.10619234087741422, 0.2391304347826087, 'Weight_KG <=
2.405\nsquared_error = 0.002\nsamples = 2\nvalue = 6.035'),
 Text(0.10513041746864007, 0.1956521739130435, 'squared_error = 0.0\nsamples =
1\nvalue = 5.986'),
 Text(0.10725426428618835, 0.1956521739130435, 'squared_error = 0.0\nsamples =
1\nvalue = 6.084'),
 Text(0.11044003451251079, 0.2391304347826087, 'Company_Lenovo <=
0.5\nsquared_error = 0.004\nsamples = 4\nvalue = 6.11'),
 Text(0.10937811110373664, 0.1956521739130435, 'Company_HP <= 0.5\nsquared_error
= 0.001\nsamples = 3\nvalue = 6.142'),
 Text(0.1083161876949625, 0.15217391304347827, 'squared_error = 0.0\nsamples =
1\nvalue = 6.107'),
 Text(0.11044003451251079, 0.15217391304347827, 'Gpu_brand_Intel <=
0.5\nsquared_error = 0.0\nsamples = 2\nvalue = 6.16'),
 Text(0.10937811110373664, 0.10869565217391304, 'squared_error = 0.0\nsamples =
1\nvalue = 6.151'),
 Text(0.11150195792128492, 0.10869565217391304, 'squared_error = -0.0\nsamples =
1\nvalue = 6.17'),
 Text(0.11150195792128492, 0.1956521739130435, 'squared_error = -0.0\nsamples =
1\nvalue = 6.014'),
 Text(0.11150195792128492, 0.32608695652173914, 'Company_HP <=
0.5\nsquared_error = 0.001\nsamples = 2\nvalue = 6.242'),
 Text(0.11044003451251079, 0.2826086956521739, 'squared_error = 0.0\nsamples =
1\nvalue = 6.213'),
 Text(0.11256388133005907, 0.2826086956521739, 'squared_error = 0.0\nsamples =
1\nvalue = 6.271'),
 Text(0.11681157496515564, 0.3695652173913043, 'Weight_KG <=
2.215\nsquared_error = 0.006\nsamples = 4\nvalue = 6.27'),
 Text(0.11574965155638149, 0.32608695652173914, 'Company_Lenovo <=
0.5\nsquared_error = 0.008\nsamples = 3\nvalue = 6.261'),
 Text(0.11468772814760736, 0.2826086956521739, 'squared_error = 0.0\nsamples =
1\nvalue = 6.273'),
 Text(0.11681157496515564, 0.2826086956521739, 'squared_error = 0.011\nsamples =
2\nvalue = 6.255'),
 Text(0.11787349837392978, 0.32608695652173914, 'squared_error = 0.0\nsamples =
1\nvalue = 6.299'),
 Text(0.12318311541780048, 0.41304347826086957, 'Gpu_brand_Intel <=
0.5\nsquared_error = 0.022\nsamples = 9\nvalue = 6.302'),
 Text(0.12212119200902635, 0.3695652173913043, 'Ram_GB <= 5.0\nsquared_error =
0.015\nsamples = 8\nvalue = 6.27'),
 Text(0.11999734519147806, 0.32608695652173914, 'Company_Lenovo <=
0.5\nsquared_error = 0.018\nsamples = 5\nvalue = 6.223'),
 Text(0.11893542178270393, 0.2826086956521739, 'Cpu_name_Intel Core i3 <=
0.5\nsquared_error = 0.005\nsamples = 4\nvalue = 6.282'),
 Text(0.11787349837392978, 0.2391304347826087, 'squared_error = 0.0\nsamples =
```

```
1\nvalue = 6.354'),
 Text(0.11999734519147806, 0.2391304347826087, 'Gpu_brand_Nvidia <=
0.5\nsquared_error = 0.004\nsamples = 3\nvalue = 6.258'),
 Text(0.11893542178270393, 0.1956521739130435, 'SSD <= 192.0\nsquared_error =
0.0\nsamples = 2\nvalue = 6.213'),
 Text(0.11787349837392978, 0.15217391304347827, 'squared_error = 0.0\nsamples =
1\nvalue = 6.213'),
 Text(0.11999734519147806, 0.15217391304347827, 'squared_error = 0.0\nsamples =
1\nvalue = 6.212'),
 Text(0.1210592686002522, 0.1956521739130435, 'squared_error = 0.0\nsamples =
1\nvalue = 6.349'),
 Text(0.1210592686002522, 0.2826086956521739, 'squared_error = -0.0\nsamples =
1\nvalue = 5.989'),
 Text(0.12424503882657463, 0.32608695652173914, 'Gpu_brand_Nvidia <=
0.5\nsquared_error = 0.001\nsamples = 3\nvalue = 6.348'),
 Text(0.12318311541780048, 0.2826086956521739, 'squared_error = 0.002\nsamples =
2\nvalue = 6.35'),
 Text(0.12530696223534876, 0.2826086956521739, 'squared_error = 0.0\nsamples =
1\nvalue = 6.344'),
 Text(0.12424503882657463, 0.3695652173913043, 'squared_error = -0.0\nsamples =
1\nvalue = 6.559'),
 Text(0.10074998340744674, 0.5434782608695652, 'squared_error = -0.0\nsamples =
1\nvalue = 6.645'),
 Text(0.13035109842702594, 0.6739130434782609, 'Ram_GB <= 5.0\nsquared_error =
0.015\nsamples = 12\nvalue = 6.448'),
 Text(0.1255724430875423, 0.6304347826086957, 'Company_Dell <=
0.5\nsquared_error = 0.008\nsamples = 7\nvalue = 6.509'),
 Text(0.12451051967876817, 0.5869565217391305, 'Gpu_brand_Intel <=
0.5\nsquared_error = 0.003\nsamples = 6\nvalue = 6.48'),
 Text(0.12238667286121989, 0.5434782608695652, 'HDD <= 500.0\nsquared_error =
0.001\nsamples = 3\nvalue = 6.529'),
 Text(0.12132474945244574, 0.5, 'Gpu_brand_Nvidia <= 0.5\nsquared_error =
0.0\nsamples = 2\nvalue = 6.552'),
 Text(0.1202628260436716, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 6.55'),
 Text(0.12238667286121989, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 6.554'),
 Text(0.12344859626999402, 0.5, 'squared_error = -0.0\nsamples = 1\nvalue =
6.485'),
 Text(0.12663436649631646, 0.5434782608695652, 'Weight_KG <= 1.72\nsquared_error
= 0.001\nsamples = 3\nvalue = 6.432'),
 Text(0.1255724430875423, 0.5, 'squared_error = 0.0\nsamples = 1\nvalue =
6.395'),
 Text(0.1276962899050906, 0.5, 'TypeName_Notebook <= 0.5\nsquared_error =
0.0\nsamples = 2\nvalue = 6.45'),
 Text(0.12663436649631646, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 6.444'),
```

```
 Text(0.12875821331386475, 0.45652173913043476, 'squared_error = -0.0\nsamples =
1\nvalue = 6.455'),
 Text(0.12663436649631646, 0.5869565217391305, 'squared_error = -0.0\nsamples =
1\nvalue = 6.68'),
 Text(0.1351297537665096, 0.6304347826086957, 'TouchScreen <= 0.5\nsquared_error
= 0.013\nsamples = 5\nvalue = 6.364'),
 Text(0.13406783035773545, 0.5869565217391305, 'Weight_KG <= 2.68\nsquared_error
= 0.005\nsamples = 4\nvalue = 6.412'),
 Text(0.1330059069489613, 0.5434782608695652, 'SSD <= 192.0\nsquared_error =
0.004\nsamples = 3\nvalue = 6.388'),
 Text(0.13194398354018716, 0.5, 'Weight_KG <= 2.28\nsquared_error = 0.0\nsamples
= 2\nvalue = 6.344'),
 Text(0.130882060131413, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 6.344'),
 Text(0.1330059069489613, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 6.344'),
 Text(0.13406783035773545, 0.5, 'squared_error = -0.0\nsamples = 1\nvalue =
6.475'),
 Text(0.1351297537665096, 0.5434782608695652, 'squared_error = -0.0\nsamples =
1\nvalue = 6.485'),
 Text(0.13619167717528374, 0.5869565217391305, 'squared_error = 0.0\nsamples =
1\nvalue = 6.172'),
 Text(0.1535806729939603, 0.8043478260869565, 'Weight_KG <= 2.26\nsquared_error
= 0.015\nsamples = 19\nvalue = 6.609'),
 Text(0.14760735381960577, 0.7608695652173914, 'SSD <= 192.0\nsquared_error =
0.012\nsamples = 16\nvalue = 6.578'),
 Text(0.14203225592354152, 0.717391304347826, 'Weight_KG <= 2.145\nsquared_error
= 0.01\nsamples = 12\nvalue = 6.542'),
 Text(0.138315523992832, 0.6739130434782609, 'PPI <= 120.833\nsquared_error =
0.008\nsamples = 6\nvalue = 6.488'),
 Text(0.13725360058405786, 0.6304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 6.361'),
 Text(0.13937744740160615, 0.6304347826086957, 'IPS <= 0.5\nsquared_error =
0.006\nsamples = 5\nvalue = 6.514'),
 Text(0.138315523992832, 0.5869565217391305, 'squared_error = 0.0\nsamples =
1\nvalue = 6.658'),
 Text(0.1404393708103803, 0.5869565217391305, 'HDD <= 250.0\nsquared_error =
0.001\nsamples = 4\nvalue = 6.478'),
 Text(0.13937744740160615, 0.5434782608695652, 'squared_error = 0.0\nsamples =
3\nvalue = 6.458'),
 Text(0.14150129421915444, 0.5434782608695652, 'squared_error = -0.0\nsamples =
1\nvalue = 6.535'),
 Text(0.145748987854251, 0.6739130434782609, 'HDD <= 750.0\nsquared_error =
0.005\nsamples = 6\nvalue = 6.597'),
 Text(0.14362514103670274, 0.6304347826086957, 'Company_Asus <=
0.5\nsquared_error = 0.003\nsamples = 3\nvalue = 6.54'),
 Text(0.1425632176279286, 0.5869565217391305, 'squared_error = 0.0\nsamples =
```

```
1\nvalue = 6.46'),
 Text(0.14468706444547688, 0.5869565217391305, 'squared_error = 0.0\nsamples =
2\nvalue = 6.58'),
 Text(0.1478728346717993, 0.6304347826086957, 'Company_HP <= 0.5\nsquared_error
= 0.0\nsamples = 3\nvalue = 6.653'),
 Text(0.14681091126302515, 0.5869565217391305, 'squared_error = 0.0\nsamples =
1\nvalue = 6.683'),
 Text(0.14893475808057344, 0.5869565217391305, 'squared_error = 0.0\nsamples =
2\nvalue = 6.639'),
 Text(0.15318245171567002, 0.717391304347826, 'TypeName_Notebook <=
0.5\nsquared_error = 0.003\nsamples = 4\nvalue = 6.685'),
 Text(0.15105860489812173, 0.6739130434782609, 'TypeName_Ultrabook <=
0.5\nsquared_error = 0.0\nsamples = 2\nvalue = 6.732'),
 Text(0.14999668148934758, 0.6304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 6.742'),
 Text(0.15212052830689587, 0.6304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 6.723'),
 Text(0.15530629853321828, 0.6739130434782609, 'Company_Dell <=
0.5\nsquared_error = 0.001\nsamples = 2\nvalue = 6.638'),
 Text(0.15424437512444414, 0.6304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 6.671'),
 Text(0.15636822194199243, 0.6304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 6.605'),
 Text(0.15955399216831487, 0.7608695652173914, 'Ram_GB <= 5.0\nsquared_error =
0.004\nsamples = 3\nvalue = 6.772'),
 Text(0.15849206875954072, 0.717391304347826, 'PPI <= 120.833\nsquared_error =
0.0\nsamples = 2\nvalue = 6.73'),
 Text(0.15743014535076658, 0.6739130434782609, 'squared_error = 0.0\nsamples =
1\nvalue = 6.745'),
 Text(0.15955399216831487, 0.6739130434782609, 'squared_error = -0.0\nsamples =
1\nvalue = 6.715'),
 Text(0.160615915577089, 0.717391304347826, 'squared_error = -0.0\nsamples =
1\nvalue = 6.855'),
 Text(0.19370146678170838, 0.8478260869565217, 'Weight_KG <=
1.485\nsquared_error = 0.114\nsamples = 102\nvalue = 6.643'),
 Text(0.1733589964823787, 0.8043478260869565, 'IPS <= 0.5\nsquared_error =
0.06\nsamples = 17\nvalue = 7.072'),
 Text(0.168049379438508, 0.7608695652173914, 'Company_HP <= 0.5\nsquared_error =
0.039\nsamples = 14\nvalue = 7.139'),
 Text(0.16380168580341142, 0.717391304347826, 'Company_Toshiba <=
0.5\nsquared_error = 0.018\nsamples = 6\nvalue = 6.995'),
 Text(0.16167783898586313, 0.6739130434782609, 'PPI <= 150.363\nsquared_error =
0.002\nsamples = 2\nvalue = 6.823'),
 Text(0.160615915577089, 0.6304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 6.866'),
 Text(0.16273976239463728, 0.6304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 6.781'),
```

```
 Text(0.16592553262095971, 0.6739130434782609, 'Weight_KG <=
1.225\nsquared_error = 0.003\nsamples = 4\nvalue = 7.081'),
 Text(0.16486360921218557, 0.6304347826086957, 'Weight_KG <=
1.125\nsquared_error = 0.001\nsamples = 3\nvalue = 7.11'),
 Text(0.16380168580341142, 0.5869565217391305, 'squared_error = 0.0\nsamples =
1\nvalue = 7.159'),
 Text(0.16592553262095971, 0.5869565217391305, 'squared_error = 0.0\nsamples =
2\nvalue = 7.086'),
 Text(0.16698745602973386, 0.6304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 6.994'),
 Text(0.17229707307360456, 0.717391304347826, 'PPI <= 161.498\nsquared_error =
0.028\nsamples = 8\nvalue = 7.248'),
 Text(0.17017322625605627, 0.6739130434782609, 'PPI <= 118.644\nsquared_error =
0.027\nsamples = 5\nvalue = 7.168'),
 Text(0.16911130284728215, 0.6304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 7.003'),
 Text(0.17123514966483042, 0.6304347826086957, 'TypeName_Notebook <=
0.5\nsquared_error = 0.026\nsamples = 4\nvalue = 7.209'),
 Text(0.17017322625605627, 0.5869565217391305, 'PPI <= 141.359\nsquared_error =
0.023\nsamples = 3\nvalue = 7.155'),
 Text(0.16911130284728215, 0.5434782608695652, 'squared_error = 0.013\nsamples =
2\nvalue = 7.239'),
 Text(0.17123514966483042, 0.5434782608695652, 'squared_error = 0.0\nsamples =
1\nvalue = 6.987'),
 Text(0.17229707307360456, 0.5869565217391305, 'squared_error = -0.0\nsamples =
1\nvalue = 7.371'),
 Text(0.17442091989115285, 0.6739130434782609, 'HDD <= 250.0\nsquared_error =
0.002\nsamples = 3\nvalue = 7.381'),
 Text(0.1733589964823787, 0.6304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 7.438'),
 Text(0.175482843299927, 0.6304347826086957, 'squared_error = 0.0\nsamples =
2\nvalue = 7.352'),
 Text(0.1786686135262494, 0.7608695652173914, 'Company_Toshiba <=
0.5\nsquared_error = 0.036\nsamples = 3\nvalue = 6.755'),
 Text(0.17760669011747526, 0.717391304347826, 'TypeName_Ultrabook <=
0.5\nsquared_error = 0.006\nsamples = 2\nvalue = 6.628'),
 Text(0.17654476670870115, 0.6739130434782609, 'squared_error = 0.0\nsamples =
1\nvalue = 6.55'),
 Text(0.1786686135262494, 0.6739130434782609, 'squared_error = 0.0\nsamples =
1\nvalue = 6.707'),
 Text(0.17973053693502355, 0.717391304347826, 'squared_error = 0.0\nsamples =
1\nvalue = 7.008'),
 Text(0.21404393708103803, 0.8043478260869565, 'OpSys_Windows <=
0.5\nsquared_error = 0.081\nsamples = 85\nvalue = 6.558'),
 Text(0.1874294816486361, 0.7608695652173914, 'Company_Dell <=
0.5\nsquared_error = 0.034\nsamples = 15\nvalue = 6.234'),
 Text(0.182916307161346, 0.717391304347826, 'Gpu_brand_Intel <=
```

```
0.5\nsquared_error = 0.011\nsamples = 11\nvalue = 6.159'),
 Text(0.1807924603437977, 0.6739130434782609, 'Weight_KG <= 2.325\nsquared_error
= 0.004\nsamples = 3\nvalue = 6.275'),
 Text(0.17973053693502355, 0.6304347826086957, 'Company_Asus <=
0.5\nsquared_error = 0.0\nsamples = 2\nvalue = 6.232'),
 Text(0.1786686135262494, 0.5869565217391305, 'squared_error = 0.0\nsamples =
1\nvalue = 6.213'),
 Text(0.1807924603437977, 0.5869565217391305, 'squared_error = -0.0\nsamples =
1\nvalue = 6.252'),
 Text(0.18185438375257185, 0.6304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 6.361'),
 Text(0.18504015397889428, 0.6739130434782609, 'Weight_KG <= 1.88\nsquared_error
= 0.007\nsamples = 8\nvalue = 6.115'),
 Text(0.18397823057012014, 0.6304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 5.976'),
 Text(0.1861020773876684, 0.6304347826086957, 'Weight_KG <= 2.03\nsquared_error
= 0.005\nsamples = 7\nvalue = 6.135'),
 Text(0.18344726886573307, 0.5869565217391305, 'HDD <= 750.0\nsquared_error =
0.0\nsamples = 2\nvalue = 6.203'),
 Text(0.18238534545695892, 0.5434782608695652, 'squared_error = 0.0\nsamples =
1\nvalue = 6.213'),
 Text(0.1845091922745072, 0.5434782608695652, 'squared_error = 0.0\nsamples =
1\nvalue = 6.194'),
 Text(0.18875688590960377, 0.5869565217391305, 'PPI <= 120.833\nsquared_error =
0.004\nsamples = 5\nvalue = 6.108'),
 Text(0.18663303909205547, 0.5434782608695652, 'Company_Lenovo <=
0.5\nsquared_error = 0.006\nsamples = 3\nvalue = 6.082'),
 Text(0.18557111568328136, 0.5, 'Company_HP <= 0.5\nsquared_error =
0.001\nsamples = 2\nvalue = 6.129'),
 Text(0.1845091922745072, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 6.091'),
 Text(0.18663303909205547, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 6.167'),
 Text(0.18769496250082962, 0.5, 'squared_error = 0.0\nsamples = 1\nvalue =
5.986'),
 Text(0.19088073272715206, 0.5434782608695652, 'HDD <= 500.0\nsquared_error =
0.0\nsamples = 2\nvalue = 6.147'),
 Text(0.1898188093183779, 0.5, 'squared_error = 0.0\nsamples = 1\nvalue =
6.148'),
 Text(0.1919426561359262, 0.5, 'squared_error = -0.0\nsamples = 1\nvalue =
6.146'),
 Text(0.1919426561359262, 0.717391304347826, 'Weight_KG <= 2.055\nsquared_error
= 0.038\nsamples = 4\nvalue = 6.441'),
 Text(0.19088073272715206, 0.6739130434782609, 'squared_error = 0.0\nsamples =
1\nvalue = 6.767'),
 Text(0.19300457954470035, 0.6739130434782609, 'HDD <= 500.0\nsquared_error =
0.003\nsamples = 3\nvalue = 6.332'),
```

```
 Text(0.1919426561359262, 0.6304347826086957, 'Weight_KG <= 2.24\nsquared_error
= 0.001\nsamples = 2\nvalue = 6.369'),
 Text(0.19088073272715206, 0.5869565217391305, 'squared_error = 0.0\nsamples =
1\nvalue = 6.344'),
 Text(0.19300457954470035, 0.5869565217391305, 'squared_error = -0.0\nsamples =
1\nvalue = 6.395'),
 Text(0.19406650295347447, 0.6304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 6.257'),
 Text(0.24065839251343996, 0.7608695652173914, 'HDD <= 750.0\nsquared_error =
0.064\nsamples = 70\nvalue = 6.627'),
 Text(0.22340213712086016, 0.717391304347826, 'Company_HP <= 0.5\nsquared_error
= 0.064\nsamples = 52\nvalue = 6.688'),
 Text(0.2099953540850866, 0.6739130434782609, 'Weight_KG <= 1.885\nsquared_error
= 0.047\nsamples = 32\nvalue = 6.611'),
 Text(0.1993761199973452, 0.6304347826086957, 'TypeName_Ultrabook <=
0.5\nsquared_error = 0.024\nsamples = 11\nvalue = 6.746'),
 Text(0.19831419658857105, 0.5869565217391305, 'PPI <= 114.874\nsquared_error =
0.019\nsamples = 10\nvalue = 6.72'),
 Text(0.1951284263622486, 0.5434782608695652, 'Weight_KG <= 1.755\nsquared_error
= 0.012\nsamples = 5\nvalue = 6.811'),
 Text(0.19406650295347447, 0.5, 'HDD <= 250.0\nsquared_error = 0.004\nsamples =
4\nvalue = 6.858'),
 Text(0.1919426561359262, 0.45652173913043476, 'Company_Toshiba <=
0.5\nsquared_error = 0.0\nsamples = 2\nvalue = 6.919'),
 Text(0.19088073272715206, 0.41304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 6.91'),
 Text(0.19300457954470035, 0.41304347826086957, 'squared_error = -0.0\nsamples =
1\nvalue = 6.928'),
 Text(0.19619034977102276, 0.45652173913043476, 'Company_Toshiba <=
0.5\nsquared_error = 0.0\nsamples = 2\nvalue = 6.797'),
 Text(0.1951284263622486, 0.41304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 6.774'),
 Text(0.1972522731797969, 0.41304347826086957, 'squared_error = -0.0\nsamples =
1\nvalue = 6.819'),
 Text(0.19619034977102276, 0.5, 'squared_error = 0.0\nsamples = 1\nvalue =
6.627'),
 Text(0.2014999668148935, 0.5434782608695652, 'Weight_KG <= 1.75\nsquared_error
= 0.009\nsamples = 5\nvalue = 6.629'),
 Text(0.1993761199973452, 0.5, 'Weight_KG <= 1.65\nsquared_error =
0.001\nsamples = 2\nvalue = 6.517'),
 Text(0.19831419658857105, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 6.485'),
 Text(0.20043804340611934, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 6.55'),
 Text(0.20362381363244175, 0.5, 'SSD <= 64.0\nsquared_error = 0.001\nsamples =
3\nvalue = 6.704'),
 Text(0.2025618902236676, 0.45652173913043476, 'squared_error = 0.0\nsamples =
```

```
1\nvalue = 6.666'),
 Text(0.2046857370412159, 0.45652173913043476, 'TouchScreen <=
0.5\nsquared_error = 0.0\nsamples = 2\nvalue = 6.723'),
 Text(0.20362381363244175, 0.41304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 6.72'),
 Text(0.20574766044999004, 0.41304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 6.725'),
 Text(0.20043804340611934, 0.5869565217391305, 'squared_error = 0.0\nsamples =
1\nvalue = 6.999'),
 Text(0.22061458817282803, 0.6304347826086957, 'Weight_KG <= 2.34\nsquared_error
= 0.045\nsamples = 21\nvalue = 6.541'),
 Text(0.21636689453773147, 0.5869565217391305, 'SSD <= 64.0\nsquared_error =
0.04\nsamples = 18\nvalue = 6.509'),
 Text(0.2121192009026349, 0.5434782608695652, 'Weight_KG <= 2.06\nsquared_error
= 0.043\nsamples = 9\nvalue = 6.58'),
 Text(0.2099953540850866, 0.5, 'Weight_KG <= 1.985\nsquared_error =
0.015\nsamples = 4\nvalue = 6.768'),
 Text(0.20893343067631248, 0.45652173913043476, 'Weight_KG <=
1.94\nsquared_error = 0.012\nsamples = 3\nvalue = 6.725'),
 Text(0.20787150726753834, 0.41304347826086957, 'Weight_KG <=
1.915\nsquared_error = 0.006\nsamples = 2\nvalue = 6.791'),
 Text(0.2068095838587642, 0.3695652173913043, 'squared_error = 0.0\nsamples =
1\nvalue = 6.715'),
 Text(0.20893343067631248, 0.3695652173913043, 'squared_error = 0.0\nsamples =
1\nvalue = 6.866'),
 Text(0.2099953540850866, 0.41304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 6.593'),
 Text(0.21105727749386075, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 6.898'),
 Text(0.21424304772018318, 0.5, 'Company_Toshiba <= 0.5\nsquared_error =
0.015\nsamples = 5\nvalue = 6.43'),
 Text(0.21318112431140904, 0.45652173913043476, 'Company_Lenovo <=
0.5\nsquared_error = 0.003\nsamples = 3\nvalue = 6.364'),
 Text(0.2121192009026349, 0.41304347826086957, 'Gpu_brand_Intel <=
0.5\nsquared_error = 0.001\nsamples = 2\nvalue = 6.4'),
 Text(0.21105727749386075, 0.3695652173913043, 'squared_error = 0.0\nsamples =
1\nvalue = 6.372'),
 Text(0.21318112431140904, 0.3695652173913043, 'squared_error = -0.0\nsamples =
1\nvalue = 6.428'),
 Text(0.21424304772018318, 0.41304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 6.292'),
 Text(0.21530497112895733, 0.45652173913043476, 'squared_error = 0.017\nsamples
= 2\nvalue = 6.53'),
 Text(0.22061458817282803, 0.5434782608695652, 'SSD <= 192.0\nsquared_error =
0.027\nsamples = 9\nvalue = 6.438'),
 Text(0.21849074135527974, 0.5, 'Weight_KG <= 2.0\nsquared_error = 0.0\nsamples
= 2\nvalue = 6.206'),
```

```
 Text(0.21742881794650562, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 6.211'),
 Text(0.21955266476405388, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 6.201'),
 Text(0.22273843499037632, 0.5, 'TouchScreen <= 0.5\nsquared_error =
0.014\nsamples = 7\nvalue = 6.505'),
 Text(0.22167651158160218, 0.45652173913043476, 'Weight_KG <=
2.15\nsquared_error = 0.004\nsamples = 6\nvalue = 6.461'),
 Text(0.2190217030596668, 0.41304347826086957, 'Weight_KG <= 2.05\nsquared_error
= 0.0\nsamples = 2\nvalue = 6.384'),
 Text(0.2179597796508927, 0.3695652173913043, 'squared_error = 0.0\nsamples =
1\nvalue = 6.374'),
 Text(0.22008362646844096, 0.3695652173913043, 'squared_error = 0.0\nsamples =
1\nvalue = 6.395'),
 Text(0.22433132010353754, 0.41304347826086957, 'Weight_KG <=
2.265\nsquared_error = 0.001\nsamples = 4\nvalue = 6.5'),
 Text(0.22220747328598925, 0.3695652173913043, 'Gpu_brand_Intel <=
0.5\nsquared_error = 0.0\nsamples = 2\nvalue = 6.532'),
 Text(0.2211455498772151, 0.32608695652173914, 'squared_error = 0.0\nsamples =
1\nvalue = 6.538'),
 Text(0.2232693966947634, 0.32608695652173914, 'squared_error = -0.0\nsamples =
1\nvalue = 6.525'),
 Text(0.2264551669210858, 0.3695652173913043, 'PPI <= 120.833\nsquared_error =
0.0\nsamples = 2\nvalue = 6.468'),
 Text(0.2253932435123117, 0.32608695652173914, 'squared_error = 0.0\nsamples =
1\nvalue = 6.475'),
 Text(0.22751709032985995, 0.32608695652173914, 'squared_error = 0.0\nsamples =
1\nvalue = 6.46'),
 Text(0.22380035839915047, 0.45652173913043476, 'squared_error = -0.0\nsamples =
1\nvalue = 6.765'),
 Text(0.2248622818079246, 0.5869565217391305, 'SSD <= 128.0\nsquared_error =
0.03\nsamples = 3\nvalue = 6.732'),
 Text(0.22380035839915047, 0.5434782608695652, 'squared_error = 0.0\nsamples =
1\nvalue = 6.961'),
 Text(0.22592420521669876, 0.5434782608695652, 'Weight_KG <= 2.7\nsquared_error
= 0.005\nsamples = 2\nvalue = 6.617'),
 Text(0.2248622818079246, 0.5, 'squared_error = 0.0\nsamples = 1\nvalue =
6.544'),
 Text(0.22698612862547288, 0.5, 'squared_error = 0.0\nsamples = 1\nvalue =
6.691'),
 Text(0.2368089201566337, 0.6739130434782609, 'Gpu_brand_Intel <=
0.5\nsquared_error = 0.067\nsamples = 20\nvalue = 6.811'),
 Text(0.23415411163469835, 0.6304347826086957, 'IPS <= 0.5\nsquared_error =
0.001\nsamples = 3\nvalue = 6.446'),
 Text(0.2330921882259242, 0.5869565217391305, 'HDD <= 250.0\nsquared_error =
0.0\nsamples = 2\nvalue = 6.423'),
 Text(0.23203026481715006, 0.5434782608695652, 'squared_error = 0.0\nsamples =
```

```
1\nvalue = 6.428'),
 Text(0.23415411163469835, 0.5434782608695652, 'squared_error = 0.0\nsamples =
1\nvalue = 6.418'),
 Text(0.2352160350434725, 0.5869565217391305, 'squared_error = -0.0\nsamples =
1\nvalue = 6.491'),
 Text(0.23946372867856905, 0.6304347826086957, 'IPS <= 0.5\nsquared_error =
0.051\nsamples = 17\nvalue = 6.876'),
 Text(0.23733988186102078, 0.5869565217391305, 'Weight_KG <=
2.355\nsquared_error = 0.04\nsamples = 14\nvalue = 6.938'),
 Text(0.23627795845224664, 0.5434782608695652, 'TypeName_Notebook <=
0.5\nsquared_error = 0.027\nsamples = 13\nvalue = 6.904'),
 Text(0.2352160350434725, 0.5, 'squared_error = 0.0\nsamples = 1\nvalue =
6.529'),
 Text(0.23733988186102078, 0.5, 'PPI <= 149.288\nsquared_error = 0.017\nsamples
= 12\nvalue = 6.935'),
 Text(0.2322957456693436, 0.45652173913043476, 'PPI <= 106.189\nsquared_error =
0.015\nsamples = 7\nvalue = 6.877'),
 Text(0.22964093714740824, 0.41304347826086957, 'HDD <= 250.0\nsquared_error =
0.01\nsamples = 2\nvalue = 6.988'),
 Text(0.2285790137386341, 0.3695652173913043, 'squared_error = 0.0\nsamples =
1\nvalue = 6.887'),
 Text(0.2307028605561824, 0.3695652173913043, 'squared_error = -0.0\nsamples =
1\nvalue = 7.089'),
 Text(0.23495055419127894, 0.41304347826086957, 'Weight_KG <=
1.795\nsquared_error = 0.01\nsamples = 5\nvalue = 6.833'),
 Text(0.23282670737373068, 0.3695652173913043, 'HDD <= 250.0\nsquared_error =
0.012\nsamples = 2\nvalue = 6.769'),
 Text(0.23176478396495653, 0.32608695652173914, 'squared_error = 0.0\nsamples =
1\nvalue = 6.88'),
 Text(0.23388863078250482, 0.32608695652173914, 'squared_error = 0.0\nsamples =
1\nvalue = 6.658'),
 Text(0.23707440100882723, 0.3695652173913043, 'PPI <= 126.573\nsquared_error =
0.004\nsamples = 3\nvalue = 6.875'),
 Text(0.2360124776000531, 0.32608695652173914, 'squared_error = 0.0\nsamples =
1\nvalue = 6.966'),
 Text(0.23813632441760138, 0.32608695652173914, 'Weight_KG <=
2.19\nsquared_error = 0.0\nsamples = 2\nvalue = 6.83'),
 Text(0.23707440100882723, 0.2826086956521739, 'squared_error = 0.0\nsamples =
1\nvalue = 6.813'),
 Text(0.23919824782637553, 0.2826086956521739, 'squared_error = 0.0\nsamples =
1\nvalue = 6.846'),
 Text(0.24238401805269794, 0.45652173913043476, 'Weight_KG <=
1.745\nsquared_error = 0.008\nsamples = 5\nvalue = 7.017'),
 Text(0.24026017123514967, 0.41304347826086957, 'HDD <= 250.0\nsquared_error =
0.002\nsamples = 3\nvalue = 6.95'),
 Text(0.23919824782637553, 0.3695652173913043, 'squared_error = 0.0\nsamples =
1\nvalue = 7.003'),
```

```
 Text(0.24132209464392382, 0.3695652173913043, 'squared_error = 0.0\nsamples =
2\nvalue = 6.923'),
 Text(0.24450786487024623, 0.41304347826086957, 'SSD <= 192.0\nsquared_error =
0.001\nsamples = 2\nvalue = 7.119'),
 Text(0.24344594146147208, 0.3695652173913043, 'squared_error = 0.0\nsamples =
1\nvalue = 7.143'),
 Text(0.24556978827902037, 0.3695652173913043, 'squared_error = -0.0\nsamples =
1\nvalue = 7.094'),
 Text(0.2384018052697949, 0.5434782608695652, 'squared_error = 0.0\nsamples =
1\nvalue = 7.377'),
 Text(0.24158757549611734, 0.5869565217391305, 'PPI <= 153.429\nsquared_error =
0.0\nsamples = 3\nvalue = 6.584'),
 Text(0.2405256520873432, 0.5434782608695652, 'squared_error = 0.0\nsamples =
2\nvalue = 6.582'),
 Text(0.2426494989048915, 0.5434782608695652, 'squared_error = -0.0\nsamples =
1\nvalue = 6.588'),
 Text(0.25791464790601976, 0.717391304347826, 'SSD <= 64.0\nsquared_error =
0.021\nsamples = 18\nvalue = 6.45'),
 Text(0.25379969469701996, 0.6739130434782609, 'Gpu_brand_Intel <=
0.5\nsquared_error = 0.014\nsamples = 16\nvalue = 6.42'),
 Text(0.2487555585053428, 0.6304347826086957, 'Gpu_brand_Nvidia <=
0.5\nsquared_error = 0.007\nsamples = 8\nvalue = 6.48'),
 Text(0.24663171168779452, 0.5869565217391305, 'PPI <= 134.279\nsquared_error =
0.003\nsamples = 3\nvalue = 6.578'),
 Text(0.24556978827902037, 0.5434782608695652, 'Company_Dell <=
0.5\nsquared_error = 0.0\nsamples = 2\nvalue = 6.54'),
 Text(0.24450786487024623, 0.5, 'squared_error = 0.0\nsamples = 1\nvalue =
6.55'),
 Text(0.24663171168779452, 0.5, 'squared_error = -0.0\nsamples = 1\nvalue =
6.529'),
 Text(0.24769363509656867, 0.5434782608695652, 'squared_error = 0.0\nsamples =
1\nvalue = 6.654'),
 Text(0.2508794053228911, 0.5869565217391305, 'Ram_GB <= 5.0\nsquared_error =
0.001\nsamples = 5\nvalue = 6.421'),
 Text(0.24981748191411696, 0.5434782608695652, 'Weight_KG <=
2.595\nsquared_error = 0.0\nsamples = 4\nvalue = 6.408'),
 Text(0.2487555585053428, 0.5, 'Weight_KG <= 2.315\nsquared_error = 0.0\nsamples
= 3\nvalue = 6.418'),
 Text(0.24769363509656867, 0.45652173913043476, 'IPS <= 0.5\nsquared_error =
0.0\nsamples = 2\nvalue = 6.421'),
 Text(0.24663171168779452, 0.41304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 6.423'),
 Text(0.2487555585053428, 0.41304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 6.418'),
 Text(0.24981748191411696, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 6.412'),
 Text(0.2508794053228911, 0.5, 'squared_error = -0.0\nsamples = 1\nvalue =
```

```
6.378'),
 Text(0.25194132873166525, 0.5434782608695652, 'squared_error = 0.0\nsamples =
1\nvalue = 6.475'),
 Text(0.25884383088869717, 0.6304347826086957, 'Company_HP <= 0.5\nsquared_error
= 0.012\nsamples = 8\nvalue = 6.359'),
 Text(0.2561890223667618, 0.5869565217391305, 'Weight_KG <= 2.205\nsquared_error
= 0.008\nsamples = 6\nvalue = 6.404'),
 Text(0.25406517554921354, 0.5434782608695652, 'Company_Dell <=
0.5\nsquared_error = 0.003\nsamples = 3\nvalue = 6.485'),
 Text(0.2530032521404394, 0.5, 'Company_Lenovo <= 0.5\nsquared_error =
0.0\nsamples = 2\nvalue = 6.446'),
 Text(0.25194132873166525, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 6.46'),
 Text(0.25406517554921354, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 6.432'),
 Text(0.25512709895798763, 0.5, 'squared_error = -0.0\nsamples = 1\nvalue =
6.563'),
 Text(0.25831286918431007, 0.5434782608695652, 'Weight_KG <= 2.24\nsquared_error
= 0.0\nsamples = 3\nvalue = 6.323'),
 Text(0.2572509457755359, 0.5, 'squared_error = 0.0\nsamples = 1\nvalue =
6.308'),
 Text(0.2593747925930842, 0.5, 'Ram_GB <= 5.0\nsquared_error = 0.0\nsamples =
2\nvalue = 6.331'),
 Text(0.25831286918431007, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 6.326'),
 Text(0.26043671600185836, 0.45652173913043476, 'squared_error = -0.0\nsamples =
1\nvalue = 6.335'),
 Text(0.2614986394106325, 0.5869565217391305, 'Weight_KG <= 1.95\nsquared_error
= 0.001\nsamples = 2\nvalue = 6.224'),
 Text(0.26043671600185836, 0.5434782608695652, 'squared_error = 0.0\nsamples =
1\nvalue = 6.192'),
 Text(0.26256056281940665, 0.5434782608695652, 'squared_error = 0.0\nsamples =
1\nvalue = 6.257'),
 Text(0.2620296011150196, 0.6739130434782609, 'Company_Dell <=
0.5\nsquared_error = 0.014\nsamples = 2\nvalue = 6.697'),
 Text(0.26096767770624546, 0.6304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 6.578'),
 Text(0.2630915245237937, 0.6304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 6.816'),
 Text(0.2906504591989115, 0.8913043478260869, 'Ram_GB <= 3.0\nsquared_error =
0.084\nsamples = 106\nvalue = 5.806'),
 Text(0.26999402668082567, 0.8478260869565217, 'Weight_KG <=
2.195\nsquared_error = 0.045\nsamples = 16\nvalue = 5.502'),
 Text(0.2689321032720515, 0.8043478260869565, 'Weight_KG <= 1.16\nsquared_error
= 0.021\nsamples = 14\nvalue = 5.44'),
 Text(0.26627729475011613, 0.7608695652173914, 'TypeName_Netbook <=
0.5\nsquared_error = 0.008\nsamples = 3\nvalue = 5.612'),
```

```
 Text(0.265215371341342, 0.717391304347826, 'TouchScreen <= 0.5\nsquared_error =
0.003\nsamples = 2\nvalue = 5.668'),
 Text(0.26415344793256784, 0.6739130434782609, 'squared_error = 0.0\nsamples =
1\nvalue = 5.719'),
 Text(0.26627729475011613, 0.6739130434782609, 'squared_error = 0.0\nsamples =
1\nvalue = 5.617'),
 Text(0.2673392181588903, 0.717391304347826, 'squared_error = 0.0\nsamples =
1\nvalue = 5.501'),
 Text(0.27158691179398686, 0.7608695652173914, 'SSD <= 8.0\nsquared_error =
0.014\nsamples = 11\nvalue = 5.393'),
 Text(0.26946306497643857, 0.717391304347826, 'Weight_KG <= 1.285\nsquared_error
= 0.009\nsamples = 9\nvalue = 5.43'),
 Text(0.2684011415676644, 0.6739130434782609, 'squared_error = 0.0\nsamples =
3\nvalue = 5.342'),
 Text(0.2705249883852127, 0.6739130434782609, 'Company_HP <= 0.5\nsquared_error
= 0.008\nsamples = 6\nvalue = 5.474'),
 Text(0.26946306497643857, 0.6304347826086957, 'Company_Vero <=
0.5\nsquared_error = 0.003\nsamples = 5\nvalue = 5.442'),
 Text(0.2684011415676644, 0.5869565217391305, 'Weight_KG <= 1.46\nsquared_error
= 0.001\nsamples = 4\nvalue = 5.465'),
 Text(0.2673392181588903, 0.5434782608695652, 'Weight_KG <= 1.41\nsquared_error
= 0.0\nsamples = 2\nvalue = 5.497'),
 Text(0.26627729475011613, 0.5, 'squared_error = 0.0\nsamples = 1\nvalue =
5.476'),
 Text(0.2684011415676644, 0.5, 'squared_error = 0.0\nsamples = 1\nvalue =
5.517'),
 Text(0.26946306497643857, 0.5434782608695652, 'squared_error = -0.0\nsamples =
2\nvalue = 5.434'),
 Text(0.2705249883852127, 0.5869565217391305, 'squared_error = -0.0\nsamples =
1\nvalue = 5.351'),
 Text(0.27158691179398686, 0.6304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 5.631'),
 Text(0.27371075861153515, 0.717391304347826, 'PPI <= 117.767\nsquared_error =
0.005\nsamples = 2\nvalue = 5.226'),
 Text(0.272648835202761, 0.6739130434782609, 'squared_error = 0.0\nsamples =
1\nvalue = 5.293'),
 Text(0.2747726820203093, 0.6739130434782609, 'squared_error = 0.0\nsamples =
1\nvalue = 5.159'),
 Text(0.27105595008959976, 0.8043478260869565, 'squared_error = 0.0\nsamples =
2\nvalue = 5.938'),
 Text(0.31130689171699744, 0.8478260869565217, 'SSD <= 80.0\nsquared_error =
0.071\nsamples = 90\nvalue = 5.86'),
 Text(0.30091736078847814, 0.8043478260869565, 'PPI <= 170.94\nsquared_error =
0.057\nsamples = 80\nvalue = 5.822'),
 Text(0.29394330324550344, 0.7608695652173914, 'Company_Vero <=
0.5\nsquared_error = 0.041\nsamples = 72\nvalue = 5.788'),
 Text(0.2868976903165859, 0.717391304347826, 'Gpu_brand_Intel <=
```

```
0.5\nsquared_error = 0.035\nsamples = 69\nvalue = 5.806'),
 Text(0.2825545895002323, 0.6739130434782609, 'PPI <= 98.353\nsquared_error =
0.003\nsamples = 3\nvalue = 6.19'),
 Text(0.2814926660914582, 0.6304347826086957, 'squared_error = 0.0\nsamples =
2\nvalue = 6.149'),
 Text(0.2836165129090064, 0.6304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 6.273'),
 Text(0.29124079113293955, 0.6739130434782609, 'TouchScreen <=
0.5\nsquared_error = 0.03\nsamples = 66\nvalue = 5.788'),
 Text(0.2857403597265547, 0.6304347826086957, 'Company_Chuwi <=
0.5\nsquared_error = 0.024\nsamples = 63\nvalue = 5.774'),
 Text(0.28467843631778056, 0.5869565217391305, 'Weight_KG <= 1.95\nsquared_error
= 0.022\nsamples = 61\nvalue = 5.782'),
 Text(0.27686334373133337, 0.5434782608695652, 'HDD <= 16.0\nsquared_error =
0.028\nsamples = 32\nvalue = 5.744'),
 Text(0.2705249883852127, 0.5, 'SSD <= 24.0\nsquared_error = 0.028\nsamples =
27\nvalue = 5.769'),
 Text(0.26660914581535805, 0.45652173913043476, 'Weight_KG <=
1.445\nsquared_error = 0.028\nsamples = 25\nvalue = 5.755'),
 Text(0.2619632309019712, 0.41304347826086957, 'Company_Lenovo <=
0.5\nsquared_error = 0.032\nsamples = 14\nvalue = 5.714'),
 Text(0.2598393840844229, 0.3695652173913043, 'Company_Mediacom <=
0.5\nsquared_error = 0.036\nsamples = 10\nvalue = 5.761'),
 Text(0.2587774606756488, 0.32608695652173914, 'TypeName_Notebook <=
0.5\nsquared_error = 0.034\nsamples = 9\nvalue = 5.786'),
 Text(0.25771553726687463, 0.2826086956521739, 'Company_HP <= 0.5\nsquared_error
= 0.015\nsamples = 8\nvalue = 5.736'),
 Text(0.2566536138581005, 0.2391304347826087, 'IPS <= 0.5\nsquared_error =
0.01\nsamples = 7\nvalue = 5.705'),
 Text(0.25399880533616515, 0.1956521739130435, 'Company_Asus <=
0.5\nsquared_error = 0.007\nsamples = 5\nvalue = 5.674'),
 Text(0.25187495851861685, 0.15217391304347827, 'Company_Dell <=
0.5\nsquared_error = 0.002\nsamples = 3\nvalue = 5.625'),
 Text(0.2508130351098427, 0.10869565217391304, 'squared_error = 0.0\nsamples =
2\nvalue = 5.595'),
 Text(0.252936881927391, 0.10869565217391304, 'squared_error = 0.0\nsamples =
1\nvalue = 5.687'),
 Text(0.25612265215371344, 0.15217391304347827, 'OpSys_Windows <=
0.5\nsquared_error = 0.007\nsamples = 2\nvalue = 5.746'),
 Text(0.2550607287449393, 0.10869565217391304, 'squared_error = 0.0\nsamples =
1\nvalue = 5.663'),
 Text(0.2571845755624876, 0.10869565217391304, 'squared_error = 0.0\nsamples =
1\nvalue = 5.829'),
 Text(0.2593084223800358, 0.1956521739130435, 'Weight_KG <= 1.3\nsquared_error =
0.008\nsamples = 2\nvalue = 5.783'),
 Text(0.2582464989712617, 0.15217391304347827, 'squared_error = 0.0\nsamples =
1\nvalue = 5.694'),
```

```
 Text(0.26037034578880996, 0.15217391304347827, 'squared_error = 0.0\nsamples =
1\nvalue = 5.872'),
 Text(0.2587774606756488, 0.2391304347826087, 'squared_error = 0.0\nsamples =
1\nvalue = 5.953'),
 Text(0.2598393840844229, 0.2826086956521739, 'squared_error = -0.0\nsamples =
1\nvalue = 6.184'),
 Text(0.26090130749319707, 0.32608695652173914, 'squared_error = -0.0\nsamples =
1\nvalue = 5.541'),
 Text(0.2640870777195195, 0.3695652173913043, 'SSD <= 8.0\nsquared_error =
0.003\nsamples = 4\nvalue = 5.597'),
 Text(0.26302515431074536, 0.32608695652173914, 'Weight_KG <=
1.435\nsquared_error = 0.004\nsamples = 3\nvalue = 5.602'),
 Text(0.2619632309019712, 0.2826086956521739, 'squared_error = 0.0\nsamples =
1\nvalue = 5.613'),
 Text(0.2640870777195195, 0.2826086956521739, 'squared_error = 0.006\nsamples =
2\nvalue = 5.597'),
 Text(0.26514900112829365, 0.32608695652173914, 'squared_error = -0.0\nsamples =
1\nvalue = 5.58'),
 Text(0.27125506072874495, 0.41304347826086957, 'Weight_KG <=
1.475\nsquared_error = 0.019\nsamples = 11\nvalue = 5.807'),
 Text(0.26833477135461603, 0.3695652173913043, 'Company_Lenovo <=
0.5\nsquared_error = 0.01\nsamples = 2\nvalue = 6.029'),
 Text(0.2672728479458419, 0.32608695652173914, 'squared_error = 0.0\nsamples =
1\nvalue = 5.927'),
 Text(0.2693966947633902, 0.32608695652173914, 'squared_error = 0.0\nsamples =
1\nvalue = 6.131'),
 Text(0.2741753501028738, 0.3695652173913043, 'Weight_KG <= 1.55\nsquared_error
= 0.007\nsamples = 9\nvalue = 5.758'),
 Text(0.27152054158093847, 0.32608695652173914, 'OpSys_Others <=
0.5\nsquared_error = 0.003\nsamples = 4\nvalue = 5.691'),
 Text(0.2704586181721643, 0.2826086956521739, 'PPI <= 134.643\nsquared_error =
0.001\nsamples = 3\nvalue = 5.66'),
 Text(0.2693966947633902, 0.2391304347826087, 'squared_error = 0.0\nsamples =
2\nvalue = 5.642'),
 Text(0.27152054158093847, 0.2391304347826087, 'squared_error = -0.0\nsamples =
1\nvalue = 5.697'),
 Text(0.2725824649897126, 0.2826086956521739, 'squared_error = -0.0\nsamples =
1\nvalue = 5.784'),
 Text(0.2768301586248092, 0.32608695652173914, 'Weight_KG <= 1.64\nsquared_error
= 0.004\nsamples = 5\nvalue = 5.812'),
 Text(0.2747063118072609, 0.2826086956521739, 'Company_Asus <=
0.5\nsquared_error = 0.004\nsamples = 3\nvalue = 5.784'),
 Text(0.27364438839848676, 0.2391304347826087, 'PPI <= 134.643\nsquared_error =
0.001\nsamples = 2\nvalue = 5.826'),
 Text(0.2725824649897126, 0.1956521739130435, 'squared_error = 0.0\nsamples =
1\nvalue = 5.855'),
 Text(0.2747063118072609, 0.1956521739130435, 'squared_error = 0.0\nsamples =
```

```
1\nvalue = 5.796'),
 Text(0.27576823521603505, 0.2391304347826087, 'squared_error = 0.0\nsamples =
1\nvalue = 5.7'),
 Text(0.2789540054423575, 0.2826086956521739, 'Company_Asus <=
0.5\nsquared_error = 0.0\nsamples = 2\nvalue = 5.854'),
 Text(0.27789208203358334, 0.2391304347826087, 'squared_error = 0.0\nsamples =
1\nvalue = 5.855'),
 Text(0.28001592885113163, 0.2391304347826087, 'squared_error = 0.0\nsamples =
1\nvalue = 5.852'),
 Text(0.2744408309550674, 0.45652173913043476, 'PPI <= 161.498\nsquared_error =
0.001\nsamples = 2\nvalue = 5.937'),
 Text(0.27337890754629324, 0.41304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 5.964'),
 Text(0.27550275436384153, 0.41304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 5.911'),
 Text(0.283201699077454, 0.5, 'Company_Dell <= 0.5\nsquared_error =
0.006\nsamples = 5\nvalue = 5.613'),
 Text(0.28213977566867987, 0.45652173913043476, 'Weight_KG <=
1.88\nsquared_error = 0.003\nsamples = 4\nvalue = 5.583'),
 Text(0.2810778522599058, 0.41304347826086957, 'HDD <= 750.0\nsquared_error =
0.002\nsamples = 3\nvalue = 5.601'),
 Text(0.28001592885113163, 0.3695652173913043, 'HDD <= 266.0\nsquared_error =
0.0\nsamples = 2\nvalue = 5.568'),
 Text(0.2789540054423575, 0.32608695652173914, 'squared_error = 0.0\nsamples =
1\nvalue = 5.58'),
 Text(0.2810778522599058, 0.32608695652173914, 'squared_error = 0.0\nsamples =
1\nvalue = 5.557'),
 Text(0.28213977566867987, 0.3695652173913043, 'squared_error = 0.0\nsamples =
1\nvalue = 5.666'),
 Text(0.283201699077454, 0.41304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 5.531'),
 Text(0.28426362248622816, 0.45652173913043476, 'squared_error = -0.0\nsamples =
1\nvalue = 5.733'),
 Text(0.2924935289042278, 0.5434782608695652, 'OpSys_Windows <=
0.5\nsquared_error = 0.012\nsamples = 29\nvalue = 5.824'),
 Text(0.2874493927125506, 0.5, 'Weight_KG <= 2.3\nsquared_error = 0.017\nsamples
= 11\nvalue = 5.777'),
 Text(0.28638746930377645, 0.45652173913043476, 'PPI <= 120.833\nsquared_error =
0.018\nsamples = 9\nvalue = 5.802'),
 Text(0.2853255458950023, 0.41304347826086957, 'Company_Lenovo <=
0.5\nsquared_error = 0.019\nsamples = 8\nvalue = 5.792'),
 Text(0.28426362248622816, 0.3695652173913043, 'Weight_KG <= 2.05\nsquared_error
= 0.024\nsamples = 6\nvalue = 5.779'),
 Text(0.283201699077454, 0.32608695652173914, 'squared_error = 0.035\nsamples =
2\nvalue = 5.803'),
 Text(0.2853255458950023, 0.32608695652173914, 'Weight_KG <= 2.15\nsquared_error
= 0.018\nsamples = 4\nvalue = 5.767'),
```

```
 Text(0.28426362248622816, 0.2826086956521739, 'squared_error = 0.0\nsamples =
1\nvalue = 5.697'),
 Text(0.28638746930377645, 0.2826086956521739, 'squared_error = 0.021\nsamples =
3\nvalue = 5.791'),
 Text(0.28638746930377645, 0.3695652173913043, 'squared_error = 0.003\nsamples =
2\nvalue = 5.83'),
 Text(0.2874493927125506, 0.41304347826086957, 'squared_error = -0.0\nsamples =
1\nvalue = 5.883'),
 Text(0.28851131612132475, 0.45652173913043476, 'squared_error = -0.0\nsamples =
2\nvalue = 5.666'),
 Text(0.29753766509590496, 0.5, 'Weight_KG <= 2.15\nsquared_error =
0.007\nsamples = 18\nvalue = 5.853'),
 Text(0.2938209331651955, 0.45652173913043476, 'HDD <= 750.0\nsquared_error =
0.004\nsamples = 9\nvalue = 5.806'),
 Text(0.2916970863476472, 0.41304347826086957, 'Company_HP <= 0.5\nsquared_error
= 0.003\nsamples = 5\nvalue = 5.767'),
 Text(0.29063516293887304, 0.3695652173913043, 'PPI <= 106.189\nsquared_error =
0.002\nsamples = 4\nvalue = 5.747'),
 Text(0.2895732395300989, 0.32608695652173914, 'Company_Asus <=
0.5\nsquared_error = 0.001\nsamples = 3\nvalue = 5.762'),
 Text(0.28851131612132475, 0.2826086956521739, 'Company_Lenovo <=
0.5\nsquared_error = 0.001\nsamples = 2\nvalue = 5.785'),
 Text(0.2874493927125506, 0.2391304347826087, 'squared_error = 0.0\nsamples =
1\nvalue = 5.808'),
 Text(0.2895732395300989, 0.2391304347826087, 'squared_error = 0.0\nsamples =
1\nvalue = 5.762'),
 Text(0.29063516293887304, 0.2826086956521739, 'squared_error = -0.0\nsamples =
1\nvalue = 5.717'),
 Text(0.2916970863476472, 0.32608695652173914, 'squared_error = 0.0\nsamples =
1\nvalue = 5.7'),
 Text(0.29275900975642133, 0.3695652173913043, 'squared_error = -0.0\nsamples =
1\nvalue = 5.849'),
 Text(0.29594477998274377, 0.41304347826086957, 'Weight_KG <=
2.05\nsquared_error = 0.001\nsamples = 4\nvalue = 5.854'),
 Text(0.2948828565739696, 0.3695652173913043, 'PPI <= 120.833\nsquared_error =
0.0\nsamples = 2\nvalue = 5.833'),
 Text(0.2938209331651955, 0.32608695652173914, 'squared_error = 0.0\nsamples =
1\nvalue = 5.826'),
 Text(0.29594477998274377, 0.32608695652173914, 'squared_error = 0.0\nsamples =
1\nvalue = 5.841'),
 Text(0.2970067033915179, 0.3695652173913043, 'squared_error = 0.0\nsamples =
2\nvalue = 5.874'),
 Text(0.30125439702661444, 0.45652173913043476, 'HDD <= 750.0\nsquared_error =
0.006\nsamples = 9\nvalue = 5.9'),
 Text(0.3001924736178403, 0.41304347826086957, 'PPI <= 120.833\nsquared_error =
0.003\nsamples = 8\nvalue = 5.922'),
 Text(0.29913055020906615, 0.3695652173913043, 'Company_Lenovo <=
```

```
0.5\nsquared_error = 0.002\nsamples = 7\nvalue = 5.933'),
 Text(0.298068626800292, 0.32608695652173914, 'Weight_KG <= 2.3\nsquared_error =
0.0\nsamples = 6\nvalue = 5.916'),
 Text(0.2970067033915179, 0.2826086956521739, 'squared_error = -0.0\nsamples =
5\nvalue = 5.911'),
 Text(0.29913055020906615, 0.2826086956521739, 'squared_error = 0.0\nsamples =
1\nvalue = 5.94'),
 Text(0.3001924736178403, 0.32608695652173914, 'squared_error = 0.0\nsamples =
1\nvalue = 6.038'),
 Text(0.30125439702661444, 0.3695652173913043, 'squared_error = -0.0\nsamples =
1\nvalue = 5.841'),
 Text(0.3023163204353886, 0.41304347826086957, 'squared_error = -0.0\nsamples =
1\nvalue = 5.724'),
 Text(0.28680228313532885, 0.5869565217391305, 'squared_error = 0.0\nsamples =
2\nvalue = 5.509'),
 Text(0.29674122253932433, 0.6304347826086957, 'Company_HP <= 0.5\nsquared_error
= 0.053\nsamples = 3\nvalue = 6.095'),
 Text(0.2956792991305502, 0.5869565217391305, 'Weight_KG <= 1.375\nsquared_error
= 0.0\nsamples = 2\nvalue = 5.932'),
 Text(0.29461737572177604, 0.5434782608695652, 'squared_error = 0.0\nsamples =
1\nvalue = 5.938'),
 Text(0.29674122253932433, 0.5434782608695652, 'squared_error = 0.0\nsamples =
1\nvalue = 5.927'),
 Text(0.2978031459480985, 0.5869565217391305, 'squared_error = -0.0\nsamples =
1\nvalue = 6.422'),
 Text(0.3009889161744209, 0.717391304347826, 'IPS <= 0.5\nsquared_error =
0.016\nsamples = 3\nvalue = 5.384'),
 Text(0.29992699276564677, 0.6739130434782609, 'PPI <= 161.498\nsquared_error =
0.0\nsamples = 2\nvalue = 5.295'),
 Text(0.2988650693568726, 0.6304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 5.313'),
 Text(0.3009889161744209, 0.6304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 5.278'),
 Text(0.30205083958319506, 0.6739130434782609, 'squared_error = 0.0\nsamples =
1\nvalue = 5.561'),
 Text(0.30789141833145284, 0.7608695652173914, 'IPS <= 0.5\nsquared_error =
0.092\nsamples = 8\nvalue = 6.125'),
 Text(0.3052366098095175, 0.717391304347826, 'Weight_KG <= 1.245\nsquared_error
= 0.002\nsamples = 2\nvalue = 6.464'),
 Text(0.30417468640074335, 0.6739130434782609, 'squared_error = 0.0\nsamples =
1\nvalue = 6.506'),
 Text(0.30629853321829165, 0.6739130434782609, 'squared_error = -0.0\nsamples =
1\nvalue = 6.422'),
 Text(0.3105462268533882, 0.717391304347826, 'PPI <= 207.036\nsquared_error =
0.071\nsamples = 6\nvalue = 6.012'),
 Text(0.30842238003583994, 0.6739130434782609, 'Weight_KG <=
1.205\nsquared_error = 0.006\nsamples = 2\nvalue = 5.778'),
```

```
 Text(0.3073604566270658, 0.6304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 5.7'),
 Text(0.3094843034446141, 0.6304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 5.855'),
 Text(0.31267007367093647, 0.6739130434782609, 'OpSys_Others <=
0.5\nsquared_error = 0.063\nsamples = 4\nvalue = 6.129'),
 Text(0.3116081502621623, 0.6304347826086957, 'TouchScreen <= 0.5\nsquared_error
= 0.025\nsamples = 3\nvalue = 6.25'),
 Text(0.3105462268533882, 0.5869565217391305, 'squared_error = 0.0\nsamples =
1\nvalue = 6.107'),
 Text(0.31267007367093647, 0.5869565217391305, 'squared_error = 0.022\nsamples =
2\nvalue = 6.321'),
 Text(0.3137319970797106, 0.6304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 5.765'),
 Text(0.3216964226455167, 0.8043478260869565, 'Weight_KG <= 1.805\nsquared_error
= 0.083\nsamples = 10\nvalue = 6.161'),
 Text(0.3179796907148072, 0.7608695652173914, 'Weight_KG <= 1.425\nsquared_error
= 0.048\nsamples = 6\nvalue = 6.339'),
 Text(0.3158558438972589, 0.717391304347826, 'Company_Lenovo <=
0.5\nsquared_error = 0.002\nsamples = 2\nvalue = 6.119'),
 Text(0.31479392048848476, 0.6739130434782609, 'squared_error = 0.0\nsamples =
1\nvalue = 6.075'),
 Text(0.31691776730603305, 0.6739130434782609, 'squared_error = -0.0\nsamples =
1\nvalue = 6.163'),
 Text(0.3201035375323555, 0.717391304347826, 'PPI <= 123.507\nsquared_error =
0.035\nsamples = 4\nvalue = 6.448'),
 Text(0.31904161412358134, 0.6739130434782609, 'squared_error = 0.0\nsamples =
1\nvalue = 6.213'),
 Text(0.32116546094112963, 0.6739130434782609, 'Company_Lenovo <=
0.5\nsquared_error = 0.023\nsamples = 3\nvalue = 6.527'),
 Text(0.3201035375323555, 0.6304347826086957, 'Weight_KG <= 1.54\nsquared_error
= 0.0\nsamples = 2\nvalue = 6.632'),
 Text(0.31904161412358134, 0.5869565217391305, 'squared_error = 0.0\nsamples =
1\nvalue = 6.653'),
 Text(0.32116546094112963, 0.5869565217391305, 'squared_error = -0.0\nsamples =
1\nvalue = 6.612'),
 Text(0.3222273843499038, 0.6304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 6.315'),
 Text(0.3254131545762262, 0.7608695652173914, 'OpSys_Windows <=
0.5\nsquared_error = 0.017\nsamples = 4\nvalue = 5.896'),
 Text(0.32435123116745207, 0.717391304347826, 'squared_error = 0.0\nsamples =
1\nvalue = 5.7'),
 Text(0.3264750779850003, 0.717391304347826, 'Weight_KG <= 2.03\nsquared_error =
0.006\nsamples = 3\nvalue = 5.961'),
 Text(0.3254131545762262, 0.6739130434782609, 'Company_HP <= 0.5\nsquared_error
= 0.001\nsamples = 2\nvalue = 6.013'),
 Text(0.32435123116745207, 0.6304347826086957, 'squared_error = 0.0\nsamples =
```

```
1\nvalue = 6.038'),
 Text(0.3264750779850003, 0.6304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 5.989'),
 Text(0.32753700139377445, 0.6739130434782609, 'squared_error = -0.0\nsamples =
1\nvalue = 5.855'),
 Text(0.7583553876020442, 0.9347826086956522, 'Ram_GB <= 14.0\nsquared_error =
0.215\nsamples = 744\nvalue = 7.143'),
 Text(0.5575532802440765, 0.8913043478260869, 'TypeName_Notebook <=
0.5\nsquared_error = 0.159\nsamples = 553\nvalue = 6.992'),
 Text(0.4469187369333643, 0.8478260869565217, 'Weight_KG <= 1.545\nsquared_error
= 0.107\nsamples = 268\nvalue = 7.196'),
 Text(0.3752175180029203, 0.8043478260869565, 'SSD <= 218.0\nsquared_error =
0.076\nsamples = 138\nvalue = 7.305'),
 Text(0.3328466184376452, 0.7608695652173914, 'Company_HP <= 0.5\nsquared_error
= 0.092\nsamples = 15\nvalue = 7.027'),
 Text(0.33178469502887104, 0.717391304347826, 'Cpu_name_Intel Core i7 <=
0.5\nsquared_error = 0.047\nsamples = 14\nvalue = 7.085'),
 Text(0.3307227716200969, 0.6739130434782609, 'PPI <= 196.028\nsquared_error =
0.026\nsamples = 13\nvalue = 7.044'),
 Text(0.3285989248025486, 0.6304347826086957, 'Company_Lenovo <=
0.5\nsquared_error = 0.007\nsamples = 6\nvalue = 6.9'),
 Text(0.32753700139377445, 0.5869565217391305, 'Weight_KG <=
1.345\nsquared_error = 0.006\nsamples = 5\nvalue = 6.879'),
 Text(0.3264750779850003, 0.5434782608695652, 'Weight_KG <= 1.255\nsquared_error
= 0.002\nsamples = 3\nvalue = 6.83'),
 Text(0.3254131545762262, 0.5, 'squared_error = 0.0\nsamples = 1\nvalue =
6.887'),
 Text(0.32753700139377445, 0.5, 'PPI <= 146.654\nsquared_error = 0.0\nsamples =
2\nvalue = 6.801'),
 Text(0.3264750779850003, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 6.801'),
 Text(0.3285989248025486, 0.45652173913043476, 'squared_error = -0.0\nsamples =
1\nvalue = 6.801'),
 Text(0.3285989248025486, 0.5434782608695652, 'squared_error = 0.002\nsamples =
2\nvalue = 6.954'),
 Text(0.32966084821132274, 0.5869565217391305, 'squared_error = 0.0\nsamples =
1\nvalue = 7.002'),
 Text(0.3328466184376452, 0.6304347826086957, 'PPI <= 282.875\nsquared_error =
0.01\nsamples = 7\nvalue = 7.167'),
 Text(0.33178469502887104, 0.5869565217391305, 'Cpu_name_Other Intel Processor
<= 0.5\nsquared_error = 0.003\nsamples = 6\nvalue = 7.132'),
 Text(0.3307227716200969, 0.5434782608695652, 'TouchScreen <= 0.5\nsquared_error
= 0.001\nsamples = 2\nvalue = 7.175'),
 Text(0.32966084821132274, 0.5, 'squared_error = 0.0\nsamples = 1\nvalue =
7.2'),
 Text(0.33178469502887104, 0.5, 'squared_error = 0.0\nsamples = 1\nvalue =
7.151'),
```

```
 Text(0.3328466184376452, 0.5434782608695652, 'squared_error = 0.003\nsamples =
4\nvalue = 7.111'),
 Text(0.3339085418464193, 0.5869565217391305, 'squared_error = 0.0\nsamples =
1\nvalue = 7.377'),
 Text(0.3328466184376452, 0.6739130434782609, 'squared_error = 0.0\nsamples =
1\nvalue = 7.625'),
 Text(0.3339085418464193, 0.717391304347826, 'squared_error = -0.0\nsamples =
1\nvalue = 6.205'),
 Text(0.41758841756819537, 0.7608695652173914, 'Company_Asus <=
0.5\nsquared_error = 0.063\nsamples = 123\nvalue = 7.339'),
 Text(0.38750974812504146, 0.717391304347826, 'Gpu_brand_Intel <=
0.5\nsquared_error = 0.052\nsamples = 113\nvalue = 7.372'),
 Text(0.3697069755093914, 0.6739130434782609, 'Company_HP <= 0.5\nsquared_error
= 0.006\nsamples = 3\nvalue = 6.922'),
 Text(0.36864505210061727, 0.6304347826086957, 'Cpu_name_Intel Core i5 <=
0.5\nsquared_error = 0.001\nsamples = 2\nvalue = 6.872'),
 Text(0.3675831286918431, 0.5869565217391305, 'squared_error = 0.0\nsamples =
1\nvalue = 6.837'),
 Text(0.3697069755093914, 0.5869565217391305, 'squared_error = 0.0\nsamples =
1\nvalue = 6.908'),
 Text(0.3707688989181655, 0.6304347826086957, 'squared_error = -0.0\nsamples =
1\nvalue = 7.02'),
 Text(0.4053125207406916, 0.6739130434782609, 'SSD <= 384.0\nsquared_error =
0.047\nsamples = 110\nvalue = 7.384'),
 Text(0.38771404393708103, 0.6304347826086957, 'PPI <= 255.281\nsquared_error =
0.046\nsamples = 86\nvalue = 7.349'),
 Text(0.37183082232693965, 0.5869565217391305, 'PPI <= 205.3\nsquared_error =
0.045\nsamples = 72\nvalue = 7.37'),
 Text(0.35572774938607554, 0.5434782608695652, 'Cpu_name_Intel Core i5 <=
0.5\nsquared_error = 0.039\nsamples = 60\nvalue = 7.339'),
 Text(0.33759208867060464, 0.5, 'Weight_KG <= 1.24\nsquared_error =
0.032\nsamples = 31\nvalue = 7.382'),
 Text(0.3307227716200969, 0.45652173913043476, 'TouchScreen <=
0.5\nsquared_error = 0.016\nsamples = 7\nvalue = 7.283'),
 Text(0.32966084821132274, 0.41304347826086957, 'Weight_KG <=
1.185\nsquared_error = 0.007\nsamples = 6\nvalue = 7.323'),
 Text(0.3285989248025486, 0.3695652173913043, 'squared_error = 0.0\nsamples =
1\nvalue = 7.438'),
 Text(0.3307227716200969, 0.3695652173913043, 'IPS <= 0.5\nsquared_error =
0.006\nsamples = 5\nvalue = 7.3'),
 Text(0.32966084821132274, 0.32608695652173914, 'Company_Dell <=
0.5\nsquared_error = 0.003\nsamples = 4\nvalue = 7.272'),
 Text(0.3285989248025486, 0.2826086956521739, 'squared_error = 0.0\nsamples =
1\nvalue = 7.217'),
 Text(0.3307227716200969, 0.2826086956521739, 'Weight_KG <= 1.215\nsquared_error
= 0.003\nsamples = 3\nvalue = 7.291'),
 Text(0.32966084821132274, 0.2391304347826087, 'squared_error = 0.002\nsamples =
```

```
2\nvalue = 7.322'),
 Text(0.33178469502887104, 0.2391304347826087, 'squared_error = -0.0\nsamples =
1\nvalue = 7.229'),
 Text(0.33178469502887104, 0.32608695652173914, 'squared_error = -0.0\nsamples =
1\nvalue = 7.408'),
 Text(0.33178469502887104, 0.41304347826086957, 'squared_error = -0.0\nsamples =
1\nvalue = 7.047'),
 Text(0.3444614057211124, 0.45652173913043476, 'Weight_KG <=
1.375\nsquared_error = 0.032\nsamples = 24\nvalue = 7.411'),
 Text(0.3380234950554191, 0.41304347826086957, 'TypeName_Ultrabook <=
0.5\nsquared_error = 0.028\nsamples = 17\nvalue = 7.451'),
 Text(0.3349704652551935, 0.3695652173913043, 'Weight_KG <= 1.325\nsquared_error
= 0.025\nsamples = 3\nvalue = 7.632'),
 Text(0.3339085418464193, 0.32608695652173914, 'squared_error = 0.0\nsamples =
1\nvalue = 7.847'),
 Text(0.3360323886639676, 0.32608695652173914, 'squared_error = 0.003\nsamples =
2\nvalue = 7.524'),
 Text(0.34107652485564477, 0.3695652173913043, 'PPI <= 161.498\nsquared_error =
0.021\nsamples = 14\nvalue = 7.413'),
 Text(0.3381562354815159, 0.32608695652173914, 'Company_Lenovo <=
0.5\nsquared_error = 0.001\nsamples = 3\nvalue = 7.492'),
 Text(0.33709431207274176, 0.2826086956521739, 'IPS <= 0.5\nsquared_error =
0.0\nsamples = 2\nvalue = 7.474'),
 Text(0.3360323886639676, 0.2391304347826087, 'squared_error = 0.0\nsamples =
1\nvalue = 7.482'),
 Text(0.3381562354815159, 0.2391304347826087, 'squared_error = 0.0\nsamples =
1\nvalue = 7.467'),
 Text(0.33921815889029006, 0.2826086956521739, 'squared_error = -0.0\nsamples =
1\nvalue = 7.528'),
 Text(0.3439968142297737, 0.32608695652173914, 'Weight_KG <=
1.256\nsquared_error = 0.024\nsamples = 11\nvalue = 7.391'),
 Text(0.34134200570783835, 0.2826086956521739, 'Weight_KG <=
1.251\nsquared_error = 0.0\nsamples = 2\nvalue = 7.514'),
 Text(0.3402800822990642, 0.2391304347826087, 'squared_error = 0.0\nsamples =
1\nvalue = 7.495'),
 Text(0.34240392911661244, 0.2391304347826087, 'squared_error = 0.0\nsamples =
1\nvalue = 7.533'),
 Text(0.346651622751709, 0.2826086956521739, 'PPI <= 170.94\nsquared_error =
0.025\nsamples = 9\nvalue = 7.363'),
 Text(0.34452777593416073, 0.2391304347826087, 'TouchScreen <=
0.5\nsquared_error = 0.003\nsamples = 3\nvalue = 7.302'),
 Text(0.3434658525253866, 0.1956521739130435, 'Company_Dell <=
0.5\nsquared_error = 0.0\nsamples = 2\nvalue = 7.264'),
 Text(0.34240392911661244, 0.15217391304347827, 'squared_error = 0.0\nsamples =
1\nvalue = 7.244'),
 Text(0.34452777593416073, 0.15217391304347827, 'squared_error = -0.0\nsamples =
1\nvalue = 7.285'),
```

```
 Text(0.3455896993429349, 0.1956521739130435, 'squared_error = 0.0\nsamples =
1\nvalue = 7.377'),
 Text(0.3487754695692573, 0.2391304347826087, 'TouchScreen <= 0.5\nsquared_error
= 0.033\nsamples = 6\nvalue = 7.394'),
 Text(0.34771354616048317, 0.1956521739130435, 'Company_Lenovo <=
0.5\nsquared_error = 0.039\nsamples = 5\nvalue = 7.384'),
 Text(0.346651622751709, 0.15217391304347827, 'squared_error = 0.064\nsamples =
3\nvalue = 7.381'),
 Text(0.3487754695692573, 0.15217391304347827, 'squared_error = 0.0\nsamples =
2\nvalue = 7.388'),
 Text(0.34983739297803146, 0.1956521739130435, 'squared_error = -0.0\nsamples =
1\nvalue = 7.446'),
 Text(0.3508993163868056, 0.41304347826086957, 'TypeName_Ultrabook <=
0.5\nsquared_error = 0.029\nsamples = 7\nvalue = 7.314'),
 Text(0.34983739297803146, 0.3695652173913043, 'squared_error = 0.012\nsamples =
2\nvalue = 7.095'),
 Text(0.35196123979557975, 0.3695652173913043, 'Weight_KG <= 1.45\nsquared_error
= 0.009\nsamples = 5\nvalue = 7.402'),
 Text(0.34983739297803146, 0.32608695652173914, 'IPS <= 0.5\nsquared_error =
0.004\nsamples = 3\nvalue = 7.338'),
 Text(0.3487754695692573, 0.2826086956521739, 'squared_error = 0.0\nsamples =
2\nvalue = 7.296'),
 Text(0.3508993163868056, 0.2826086956521739, 'squared_error = -0.0\nsamples =
1\nvalue = 7.422'),
 Text(0.35408508661312804, 0.32608695652173914, 'IPS <= 0.5\nsquared_error =
0.002\nsamples = 2\nvalue = 7.498'),
 Text(0.3530231632043539, 0.2826086956521739, 'squared_error = 0.0\nsamples =
1\nvalue = 7.543'),
 Text(0.3551470100219022, 0.2826086956521739, 'squared_error = -0.0\nsamples =
1\nvalue = 7.453'),
 Text(0.37386341010154645, 0.5, 'Weight_KG <= 1.49\nsquared_error =
0.043\nsamples = 29\nvalue = 7.292'),
 Text(0.37173956328399815, 0.45652173913043476, 'PPI <= 121.59\nsquared_error =
0.035\nsamples = 27\nvalue = 7.314'),
 Text(0.370677639875224, 0.41304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 6.999'),
 Text(0.3728014866927723, 0.41304347826086957, 'Weight_KG <=
1.415\nsquared_error = 0.032\nsamples = 26\nvalue = 7.326'),
 Text(0.370677639875224, 0.3695652173913043, 'Weight_KG <= 1.405\nsquared_error
= 0.033\nsamples = 24\nvalue = 7.315'),
 Text(0.36961571646644986, 0.32608695652173914, 'PPI <= 187.966\nsquared_error =
0.025\nsamples = 23\nvalue = 7.335'),
 Text(0.3644388398486759, 0.2826086956521739, 'IPS <= 0.5\nsquared_error =
0.025\nsamples = 21\nvalue = 7.348'),
 Text(0.3572708568394505, 0.2391304347826087, 'TouchScreen <= 0.5\nsquared_error
= 0.024\nsamples = 12\nvalue = 7.379'),
 Text(0.3530231632043539, 0.1956521739130435, 'Company_HP <= 0.5\nsquared_error
```

```
= 0.011\nsamples = 7\nvalue = 7.302'),
 Text(0.3508993163868056, 0.15217391304347827, 'Weight_KG <=
1.325\nsquared_error = 0.002\nsamples = 4\nvalue = 7.341'),
 Text(0.34983739297803146, 0.10869565217391304, 'Company_Dell <=
0.5\nsquared_error = 0.001\nsamples = 3\nvalue = 7.321'),
 Text(0.3487754695692573, 0.06521739130434782, 'squared_error = 0.0\nsamples =
1\nvalue = 7.371'),
 Text(0.3508993163868056, 0.06521739130434782, 'Weight_KG <=
1.245\nsquared_error = 0.0\nsamples = 2\nvalue = 7.296'),
 Text(0.34983739297803146, 0.021739130434782608, 'squared_error = 0.0\nsamples =
1\nvalue = 7.279'),
 Text(0.35196123979557975, 0.021739130434782608, 'squared_error = -0.0\nsamples
= 1\nvalue = 7.313'),
 Text(0.35196123979557975, 0.10869565217391304, 'squared_error = -0.0\nsamples =
1\nvalue = 7.402'),
 Text(0.3551470100219022, 0.15217391304347827, 'TypeName_Netbook <=
0.5\nsquared_error = 0.018\nsamples = 3\nvalue = 7.249'),
 Text(0.35408508661312804, 0.10869565217391304, 'squared_error = 0.0\nsamples =
1\nvalue = 7.089'),
 Text(0.35620893343067633, 0.10869565217391304, 'PPI <= 150.8\nsquared_error =
0.008\nsamples = 2\nvalue = 7.328'),
 Text(0.3551470100219022, 0.06521739130434782, 'squared_error = 0.0\nsamples =
1\nvalue = 7.236'),
 Text(0.3572708568394505, 0.06521739130434782, 'squared_error = -0.0\nsamples =
1\nvalue = 7.42'),
 Text(0.361518550474547, 0.1956521739130435, 'PPI <= 170.94\nsquared_error =
0.021\nsamples = 5\nvalue = 7.488'),
 Text(0.3593947036569987, 0.15217391304347827, 'Company_HP <= 0.5\nsquared_error
= 0.008\nsamples = 2\nvalue = 7.64'),
 Text(0.35833278024822457, 0.10869565217391304, 'squared_error = 0.0\nsamples =
1\nvalue = 7.549'),
 Text(0.36045662706577286, 0.10869565217391304, 'squared_error = -0.0\nsamples =
1\nvalue = 7.731'),
 Text(0.3636423972920953, 0.15217391304347827, 'Weight_KG <= 1.35\nsquared_error
= 0.004\nsamples = 3\nvalue = 7.387'),
 Text(0.36258047388332115, 0.10869565217391304, 'Weight_KG <= 1.3\nsquared_error
= 0.0\nsamples = 2\nvalue = 7.433'),
 Text(0.361518550474547, 0.06521739130434782, 'squared_error = 0.0\nsamples =
1\nvalue = 7.446'),
 Text(0.3636423972920953, 0.06521739130434782, 'squared_error = 0.0\nsamples =
1\nvalue = 7.421'),
 Text(0.36470432070086944, 0.10869565217391304, 'squared_error = 0.0\nsamples =
1\nvalue = 7.295'),
 Text(0.37160682285790136, 0.2391304347826087, 'TouchScreen <=
0.5\nsquared_error = 0.024\nsamples = 9\nvalue = 7.305'),
 Text(0.368952014335966, 0.1956521739130435, 'Weight_KG <= 1.34\nsquared_error =
0.008\nsamples = 5\nvalue = 7.388'),
```

```
 Text(0.3678900909271919, 0.15217391304347827, 'Company_Dell <=
0.5\nsquared_error = 0.006\nsamples = 3\nvalue = 7.401'),
 Text(0.36682816751841774, 0.10869565217391304, 'squared_error = 0.008\nsamples
= 2\nvalue = 7.404'),
 Text(0.368952014335966, 0.10869565217391304, 'squared_error = 0.0\nsamples =
1\nvalue = 7.396'),
 Text(0.3700139377447402, 0.15217391304347827, 'squared_error = 0.011\nsamples =
2\nvalue = 7.369'),
 Text(0.37426163137983676, 0.1956521739130435, 'Weight_KG <=
1.335\nsquared_error = 0.026\nsamples = 4\nvalue = 7.201'),
 Text(0.37213778456228847, 0.15217391304347827, 'Company_Lenovo <=
0.5\nsquared_error = 0.023\nsamples = 2\nvalue = 7.092'),
 Text(0.3710758611535143, 0.10869565217391304, 'squared_error = 0.0\nsamples =
1\nvalue = 7.244'),
 Text(0.3731997079710626, 0.10869565217391304, 'squared_error = 0.0\nsamples =
1\nvalue = 6.941'),
 Text(0.376385478197385, 0.15217391304347827, 'Weight_KG <= 1.385\nsquared_error
= 0.004\nsamples = 2\nvalue = 7.31'),
 Text(0.37532355478861085, 0.10869565217391304, 'squared_error = 0.0\nsamples =
1\nvalue = 7.244'),
 Text(0.37744740160615914, 0.10869565217391304, 'squared_error = 0.0\nsamples =
1\nvalue = 7.377'),
 Text(0.3747925930842238, 0.2826086956521739, 'Company_Huawei <=
0.5\nsquared_error = 0.0\nsamples = 2\nvalue = 7.204'),
 Text(0.37373066967544966, 0.2391304347826087, 'squared_error = 0.0\nsamples =
1\nvalue = 7.2'),
 Text(0.37585451649299795, 0.2391304347826087, 'squared_error = 0.0\nsamples =
1\nvalue = 7.207'),
 Text(0.37173956328399815, 0.32608695652173914, 'squared_error = -0.0\nsamples =
1\nvalue = 6.855'),
 Text(0.3749253335103206, 0.3695652173913043, 'Weight_KG <= 1.45\nsquared_error
= 0.004\nsamples = 2\nvalue = 7.457'),
 Text(0.37386341010154645, 0.32608695652173914, 'squared_error = 0.0\nsamples =
1\nvalue = 7.519'),
 Text(0.37598725691909474, 0.32608695652173914, 'squared_error = 0.0\nsamples =
1\nvalue = 7.396'),
 Text(0.37598725691909474, 0.45652173913043476, 'IPS <= 0.5\nsquared_error =
0.065\nsamples = 2\nvalue = 6.998'),
 Text(0.3749253335103206, 0.41304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 7.254'),
 Text(0.3770491803278688, 0.41304347826086957, 'squared_error = -0.0\nsamples =
1\nvalue = 6.742'),
 Text(0.3879338952678038, 0.5434782608695652, 'PPI <= 223.622\nsquared_error =
0.043\nsamples = 12\nvalue = 7.527'),
 Text(0.3839516824849008, 0.5, 'Weight_KG <= 1.41\nsquared_error =
0.022\nsamples = 8\nvalue = 7.634'),
 Text(0.38023495055419126, 0.45652173913043476, 'Weight_KG <=
```

```
1.315\nsquared_error = 0.022\nsamples = 3\nvalue = 7.562'),
 Text(0.3791730271454171, 0.41304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 7.757'),
 Text(0.3812968739629654, 0.41304347826086957, 'Ram_GB <= 10.0\nsquared_error =
0.004\nsamples = 2\nvalue = 7.464'),
 Text(0.38023495055419126, 0.3695652173913043, 'squared_error = 0.0\nsamples =
1\nvalue = 7.401'),
 Text(0.38235879737173956, 0.3695652173913043, 'squared_error = 0.0\nsamples =
1\nvalue = 7.527'),
 Text(0.3876684144156103, 0.45652173913043476, 'Company_Lenovo <=
0.5\nsquared_error = 0.016\nsamples = 5\nvalue = 7.677'),
 Text(0.385544567598062, 0.41304347826086957, 'TouchScreen <= 0.5\nsquared_error
= 0.007\nsamples = 3\nvalue = 7.591'),
 Text(0.38448264418928785, 0.3695652173913043, 'squared_error = 0.0\nsamples =
1\nvalue = 7.695'),
 Text(0.38660649100683614, 0.3695652173913043, 'TypeName_Ultrabook <=
0.5\nsquared_error = 0.002\nsamples = 2\nvalue = 7.539'),
 Text(0.385544567598062, 0.32608695652173914, 'squared_error = 0.0\nsamples =
1\nvalue = 7.495'),
 Text(0.3876684144156103, 0.32608695652173914, 'squared_error = 0.0\nsamples =
1\nvalue = 7.582'),
 Text(0.3897922612331586, 0.41304347826086957, 'IPS <= 0.5\nsquared_error =
0.003\nsamples = 2\nvalue = 7.807'),
 Text(0.38873033782438443, 0.3695652173913043, 'squared_error = 0.0\nsamples =
1\nvalue = 7.859'),
 Text(0.3908541846419327, 0.3695652173913043, 'squared_error = 0.0\nsamples =
1\nvalue = 7.754'),
 Text(0.39191610805070687, 0.5, 'Cpu_name_Intel Core i5 <= 0.5\nsquared_error =
0.017\nsamples = 4\nvalue = 7.312'),
 Text(0.3908541846419327, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 7.141'),
 Text(0.39297803145948096, 0.45652173913043476, 'IPS <= 0.5\nsquared_error =
0.01\nsamples = 3\nvalue = 7.369'),
 Text(0.39191610805070687, 0.41304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 7.352'),
 Text(0.3940399548682551, 0.41304347826086957, 'squared_error = 0.014\nsamples =
2\nvalue = 7.378'),
 Text(0.4035972655472224, 0.5869565217391305, 'Company_Dell <=
0.5\nsquared_error = 0.04\nsamples = 14\nvalue = 7.239'),
 Text(0.4004114953209, 0.5434782608695652, 'Cpu_name_Intel Core i5 <=
0.5\nsquared_error = 0.015\nsamples = 5\nvalue = 7.086'),
 Text(0.3982876485033517, 0.5, 'PPI <= 314.285\nsquared_error = 0.003\nsamples =
3\nvalue = 7.009'),
 Text(0.39722572509457754, 0.45652173913043476, 'Weight_KG <=
1.365\nsquared_error = 0.0\nsamples = 2\nvalue = 7.045'),
 Text(0.3961638016858034, 0.41304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 7.043'),
```

```
 Text(0.3982876485033517, 0.41304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 7.047'),
 Text(0.39934957191212583, 0.45652173913043476, 'squared_error = -0.0\nsamples =
1\nvalue = 6.936'),
 Text(0.40253534213844827, 0.5, 'Weight_KG <= 1.295\nsquared_error =
0.012\nsamples = 2\nvalue = 7.201'),
 Text(0.4014734187296741, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 7.313'),
 Text(0.4035972655472224, 0.45652173913043476, 'squared_error = -0.0\nsamples =
1\nvalue = 7.089'),
 Text(0.40678303577354485, 0.5434782608695652, 'OpSys_Windows <=
0.5\nsquared_error = 0.033\nsamples = 9\nvalue = 7.325'),
 Text(0.4057211123647707, 0.5, 'squared_error = 0.0\nsamples = 1\nvalue =
7.002'),
 Text(0.407844959182319, 0.5, 'Cpu_name_Intel Core i5 <= 0.5\nsquared_error =
0.022\nsamples = 8\nvalue = 7.365'),
 Text(0.4057211123647707, 0.45652173913043476, 'IPS <= 0.5\nsquared_error =
0.022\nsamples = 5\nvalue = 7.298'),
 Text(0.40465918895599656, 0.41304347826086957, 'TouchScreen <=
0.5\nsquared_error = 0.003\nsamples = 4\nvalue = 7.229'),
 Text(0.4035972655472224, 0.3695652173913043, 'squared_error = 0.0\nsamples =
1\nvalue = 7.145'),
 Text(0.4057211123647707, 0.3695652173913043, 'Weight_KG <= 1.26\nsquared_error
= 0.001\nsamples = 3\nvalue = 7.257'),
 Text(0.40465918895599656, 0.32608695652173914, 'Weight_KG <=
1.215\nsquared_error = 0.0\nsamples = 2\nvalue = 7.236'),
 Text(0.4035972655472224, 0.2826086956521739, 'squared_error = 0.0\nsamples =
1\nvalue = 7.244'),
 Text(0.4057211123647707, 0.2826086956521739, 'squared_error = -0.0\nsamples =
1\nvalue = 7.229'),
 Text(0.40678303577354485, 0.32608695652173914, 'squared_error = 0.0\nsamples =
1\nvalue = 7.298'),
 Text(0.40678303577354485, 0.41304347826086957, 'squared_error = -0.0\nsamples =
1\nvalue = 7.575'),
 Text(0.40996880599986724, 0.45652173913043476, 'Weight_KG <=
1.265\nsquared_error = 0.004\nsamples = 3\nvalue = 7.476'),
 Text(0.4089068825910931, 0.41304347826086957, 'Weight_KG <=
1.235\nsquared_error = 0.0\nsamples = 2\nvalue = 7.518'),
 Text(0.407844959182319, 0.3695652173913043, 'squared_error = 0.0\nsamples =
1\nvalue = 7.533'),
 Text(0.40996880599986724, 0.3695652173913043, 'squared_error = 0.0\nsamples =
1\nvalue = 7.503'),
 Text(0.4110307294086414, 0.41304347826086957, 'squared_error = -0.0\nsamples =
1\nvalue = 7.393'),
 Text(0.4229109975443021, 0.6304347826086957, 'PPI <= 157.923\nsquared_error =
0.03\nsamples = 24\nvalue = 7.511'),
 Text(0.41527842304373797, 0.5869565217391305, 'Weight_KG <= 1.51\nsquared_error
```

= 0.027\nsamples = 7\nvalue = 7.618'),
 Text(0.4142164996349638, 0.5434782608695652, 'Weight_KG <= 1.035\nsquared_error
= 0.009\nsamples = 5\nvalue = 7.7'),
 Text(0.4131545762261897, 0.5, 'squared_error = 0.0\nsamples = 1\nvalue =
7.549'),
 Text(0.41527842304373797, 0.5, 'Weight_KG <= 1.44\nsquared_error =
0.004\nsamples = 4\nvalue = 7.737'),
 Text(0.4142164996349638, 0.45652173913043476, 'IPS <= 0.5\nsquared_error =
0.002\nsamples = 3\nvalue = 7.768'),
 Text(0.4131545762261897, 0.41304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 7.74'),
 Text(0.41527842304373797, 0.41304347826086957, 'Company_LG <=
0.5\nsquared_error = 0.002\nsamples = 2\nvalue = 7.782'),
 Text(0.4142164996349638, 0.3695652173913043, 'squared_error = 0.0\nsamples =
1\nvalue = 7.824'),
 Text(0.4163403464525121, 0.3695652173913043, 'squared_error = 0.0\nsamples =
1\nvalue = 7.74'),
 Text(0.4163403464525121, 0.45652173913043476, 'squared_error = -0.0\nsamples =
1\nvalue = 7.644'),
 Text(0.4163403464525121, 0.5434782608695652, 'squared_error = 0.015\nsamples =
2\nvalue = 7.413'),
 Text(0.43054357204486626, 0.5869565217391305, 'Weight_KG <=
1.355\nsquared_error = 0.024\nsamples = 17\nvalue = 7.467'),
 Text(0.42563217627928585, 0.5434782608695652, 'Weight_KG <= 1.25\nsquared_error
= 0.026\nsamples = 12\nvalue = 7.423'),
 Text(0.4221809252007699, 0.5, 'Weight_KG <= 1.075\nsquared_error =
0.023\nsamples = 10\nvalue = 7.459'),
 Text(0.4184641932700604, 0.45652173913043476, 'TypeName_Ultrabook <=
0.5\nsquared_error = 0.014\nsamples = 4\nvalue = 7.357'),
 Text(0.41740226986128626, 0.41304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 7.554'),
 Text(0.41952611667883455, 0.41304347826086957, 'Company_Lenovo <=
0.5\nsquared_error = 0.001\nsamples = 3\nvalue = 7.292'),
 Text(0.4184641932700604, 0.3695652173913043, 'OpSys_Windows <=
0.5\nsquared_error = 0.0\nsamples = 2\nvalue = 7.316'),
 Text(0.41740226986128626, 0.32608695652173914, 'squared_error = 0.0\nsamples =
1\nvalue = 7.32'),
 Text(0.41952611667883455, 0.32608695652173914, 'squared_error = 0.0\nsamples =
1\nvalue = 7.313'),
 Text(0.4205880400876087, 0.3695652173913043, 'squared_error = 0.0\nsamples =
1\nvalue = 7.244'),
 Text(0.42589765713147937, 0.45652173913043476, 'Weight_KG <=
1.165\nsquared_error = 0.018\nsamples = 6\nvalue = 7.526'),
 Text(0.42377381031393113, 0.41304347826086957, 'PPI <= 193.013\nsquared_error =
0.01\nsamples = 3\nvalue = 7.603'),
 Text(0.422711886905157, 0.3695652173913043, 'Weight_KG <= 1.13\nsquared_error =
0.002\nsamples = 2\nvalue = 7.538'),

```
 Text(0.42164996349638284, 0.32608695652173914, 'squared_error = 0.0\nsamples =
1\nvalue = 7.49'),
 Text(0.42377381031393113, 0.32608695652173914, 'squared_error = 0.0\nsamples =
1\nvalue = 7.585'),
 Text(0.4248357337227052, 0.3695652173913043, 'squared_error = 0.0\nsamples =
1\nvalue = 7.733'),
 Text(0.42802150394902766, 0.41304347826086957, 'Weight_KG <=
1.215\nsquared_error = 0.015\nsamples = 3\nvalue = 7.45'),
 Text(0.4269595805402535, 0.3695652173913043, 'TouchScreen <= 0.5\nsquared_error
= 0.003\nsamples = 2\nvalue = 7.372'),
 Text(0.42589765713147937, 0.32608695652173914, 'squared_error = 0.0\nsamples =
1\nvalue = 7.43'),
 Text(0.42802150394902766, 0.32608695652173914, 'squared_error = -0.0\nsamples =
1\nvalue = 7.313'),
 Text(0.4290834273578018, 0.3695652173913043, 'squared_error = -0.0\nsamples =
1\nvalue = 7.607'),
 Text(0.4290834273578018, 0.5, 'Weight_KG <= 1.3\nsquared_error = 0.003\nsamples
= 2\nvalue = 7.245'),
 Text(0.42802150394902766, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 7.302'),
 Text(0.43014535076657595, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 7.188'),
 Text(0.4354549678104467, 0.5434782608695652, 'PPI <= 193.233\nsquared_error =
0.002\nsamples = 5\nvalue = 7.572'),
 Text(0.4333311209928984, 0.5, 'Cpu_name_Intel Core i5 <= 0.5\nsquared_error =
0.001\nsamples = 3\nvalue = 7.539'),
 Text(0.43226919758412424, 0.45652173913043476, 'PPI <= 162.064\nsquared_error =
0.001\nsamples = 2\nvalue = 7.554'),
 Text(0.4312072741753501, 0.41304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 7.522'),
 Text(0.4333311209928984, 0.41304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 7.586'),
 Text(0.43439304440167253, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 7.509'),
 Text(0.43757881462799497, 0.5, 'IPS <= 0.5\nsquared_error = 0.0\nsamples =
2\nvalue = 7.621'),
 Text(0.4365168912192208, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 7.621'),
 Text(0.4386407380367691, 0.45652173913043476, 'squared_error = -0.0\nsamples =
1\nvalue = 7.621'),
 Text(0.44766708701134933, 0.717391304347826, 'Cpu_name_Other Intel Processor <=
0.5\nsquared_error = 0.052\nsamples = 10\nvalue = 6.974'),
 Text(0.4466051636025752, 0.6739130434782609, 'SSD <= 640.0\nsquared_error =
0.019\nsamples = 8\nvalue = 7.07'),
 Text(0.44554324019380104, 0.6304347826086957, 'Weight_KG <=
1.285\nsquared_error = 0.012\nsamples = 7\nvalue = 7.035'),
 Text(0.44288843167186565, 0.5869565217391305, 'Weight_KG <=
```

```
1.175\nsquared_error = 0.013\nsamples = 5\nvalue = 7.001'),
 Text(0.4407645848543174, 0.5434782608695652, 'PPI <= 161.498\nsquared_error =
0.005\nsamples = 2\nvalue = 7.108'),
 Text(0.43970266144554326, 0.5, 'squared_error = 0.0\nsamples = 1\nvalue =
7.034'),
 Text(0.4418265082630915, 0.5, 'squared_error = 0.0\nsamples = 1\nvalue =
7.182'),
 Text(0.44501227848941394, 0.5434782608695652, 'TypeName_Ultrabook <=
0.5\nsquared_error = 0.005\nsamples = 3\nvalue = 6.93'),
 Text(0.4439503550806398, 0.5, 'squared_error = 0.0\nsamples = 1\nvalue =
6.833'),
 Text(0.4460742018981881, 0.5, 'squared_error = 0.001\nsamples = 2\nvalue =
6.979'),
 Text(0.4481980487157364, 0.5869565217391305, 'PPI <= 161.498\nsquared_error =
0.001\nsamples = 2\nvalue = 7.119'),
 Text(0.44713612530696223, 0.5434782608695652, 'squared_error = 0.0\nsamples =
1\nvalue = 7.084'),
 Text(0.4492599721245105, 0.5434782608695652, 'squared_error = -0.0\nsamples =
1\nvalue = 7.155'),
 Text(0.44766708701134933, 0.6304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 7.313'),
 Text(0.4487290104201235, 0.6739130434782609, 'squared_error = 0.0\nsamples =
2\nvalue = 6.592'),
 Text(0.5186199558638083, 0.8043478260869565, 'TypeName_Workstation <=
0.5\nsquared_error = 0.114\nsamples = 130\nvalue = 7.08'),
 Text(0.4872963264087078, 0.7608695652173914, 'PPI <= 134.279\nsquared_error =
0.073\nsamples = 115\nvalue = 7.006'),
 Text(0.4556315125771554, 0.717391304347826, 'TypeName_Gaming <=
0.5\nsquared_error = 0.053\nsamples = 22\nvalue = 7.217'),
 Text(0.4535076657596071, 0.6739130434782609, 'Gpu_brand_Intel <=
0.5\nsquared_error = 0.008\nsamples = 2\nvalue = 6.818'),
 Text(0.45244574235083296, 0.6304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 6.907'),
 Text(0.45456958916838125, 0.6304347826086957, 'squared_error = -0.0\nsamples =
1\nvalue = 6.729'),
 Text(0.45775535939470363, 0.6739130434782609, 'OpSys_Windows <=
0.5\nsquared_error = 0.04\nsamples = 20\nvalue = 7.257'),
 Text(0.45669343598592954, 0.6304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 6.855'),
 Text(0.4588172828034778, 0.6304347826086957, 'Ram_GB <= 10.0\nsquared_error =
0.033\nsamples = 19\nvalue = 7.278'),
 Text(0.45244574235083296, 0.5869565217391305, 'SSD <= 64.0\nsquared_error =
0.026\nsamples = 14\nvalue = 7.219'),
 Text(0.4513838189420588, 0.5434782608695652, 'squared_error = 0.004\nsamples =
2\nvalue = 7.013'),
 Text(0.4535076657596071, 0.5434782608695652, 'Company_Asus <=
0.5\nsquared_error = 0.021\nsamples = 12\nvalue = 7.253'),
```

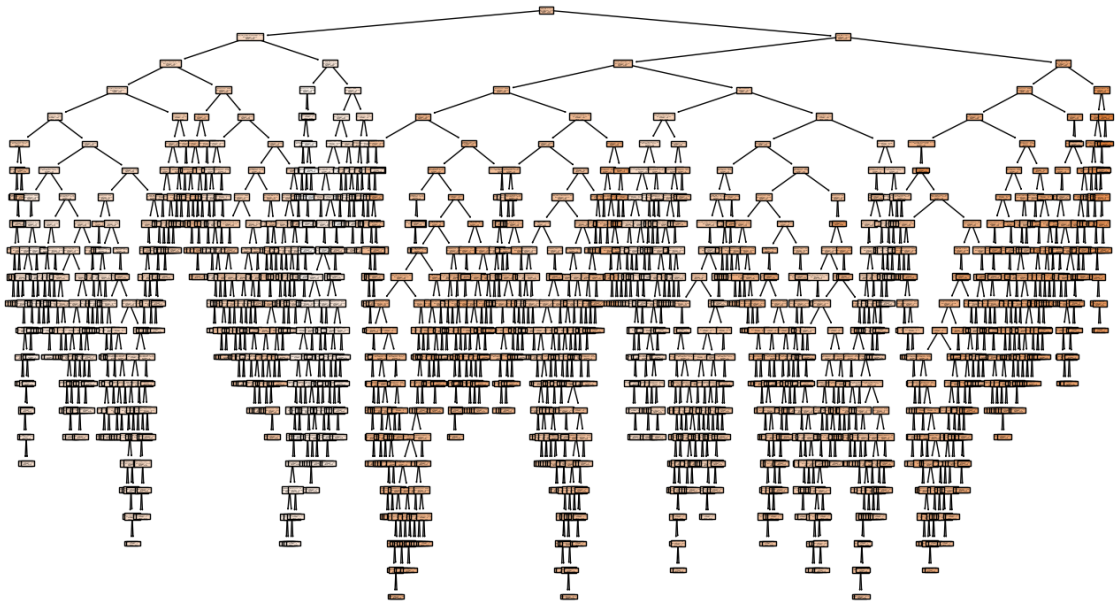
```
 Text(0.4481980487157364, 0.5, 'Weight_KG <= 2.65\nsquared_error =
0.015\nsamples = 8\nvalue = 7.195'),
 Text(0.44713612530696223, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 7.377'),
 Text(0.4492599721245105, 0.45652173913043476, 'Weight_KG <=
2.725\nsquared_error = 0.012\nsamples = 7\nvalue = 7.169'),
 Text(0.4460742018981881, 0.41304347826086957, 'Cpu_name_Intel Core i7 <=
0.5\nsquared_error = 0.01\nsamples = 4\nvalue = 7.135'),
 Text(0.44501227848941394, 0.3695652173913043, 'squared_error = 0.0\nsamples =
1\nvalue = 6.999'),
 Text(0.44713612530696223, 0.3695652173913043, 'SSD <= 192.0\nsquared_error =
0.005\nsamples = 3\nvalue = 7.18'),
 Text(0.4460742018981881, 0.32608695652173914, 'squared_error = 0.007\nsamples =
2\nvalue = 7.167'),
 Text(0.4481980487157364, 0.32608695652173914, 'squared_error = 0.0\nsamples =
1\nvalue = 7.207'),
 Text(0.45244574235083296, 0.41304347826086957, 'Weight_KG <=
3.125\nsquared_error = 0.01\nsamples = 3\nvalue = 7.216'),
 Text(0.4513838189420588, 0.3695652173913043, 'Company_HP <= 0.5\nsquared_error
= 0.001\nsamples = 2\nvalue = 7.282'),
 Text(0.45032189553328467, 0.32608695652173914, 'squared_error = 0.0\nsamples =
1\nvalue = 7.321'),
 Text(0.45244574235083296, 0.32608695652173914, 'squared_error = 0.0\nsamples =
1\nvalue = 7.244'),
 Text(0.4535076657596071, 0.3695652173913043, 'squared_error = 0.0\nsamples =
1\nvalue = 7.083'),
 Text(0.4588172828034778, 0.5, 'Weight_KG <= 3.36\nsquared_error =
0.013\nsamples = 4\nvalue = 7.368'),
 Text(0.45775535939470363, 0.45652173913043476, 'Cpu_name_Intel Core i7 <=
0.5\nsquared_error = 0.002\nsamples = 3\nvalue = 7.43'),
 Text(0.45669343598592954, 0.41304347826086957, 'Cpu_name_Intel Core i5 <=
0.5\nsquared_error = 0.0\nsamples = 2\nvalue = 7.457'),
 Text(0.4556315125771554, 0.3695652173913043, 'squared_error = 0.0\nsamples =
1\nvalue = 7.435'),
 Text(0.45775535939470363, 0.3695652173913043, 'squared_error = 0.0\nsamples =
1\nvalue = 7.479'),
 Text(0.4588172828034778, 0.41304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 7.377'),
 Text(0.4598792062122519, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 7.18'),
 Text(0.46518882325612265, 0.5869565217391305, 'Cpu_name_Intel Core i5 <=
0.5\nsquared_error = 0.015\nsamples = 5\nvalue = 7.444'),
 Text(0.4641268998473485, 0.5434782608695652, 'SSD <= 128.0\nsquared_error =
0.004\nsamples = 4\nvalue = 7.5'),
 Text(0.46306497643857436, 0.5, 'Weight_KG <= 3.76\nsquared_error =
0.001\nsamples = 3\nvalue = 7.467'),
 Text(0.4620030530298002, 0.45652173913043476, 'Weight_KG <= 3.18\nsquared_error
```

= 0.0\nsamples = 2\nvalue = 7.481'),
 Text(0.46094112962102607, 0.41304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 7.495'),
 Text(0.46306497643857436, 0.41304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 7.467'),
 Text(0.4641268998473485, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 7.438'),
 Text(0.46518882325612265, 0.5, 'squared_error = -0.0\nsamples = 1\nvalue =
7.6'),
 Text(0.4662507466648968, 0.5434782608695652, 'squared_error = 0.0\nsamples =
1\nvalue = 7.222'),
 Text(0.5189611402402602, 0.717391304347826, 'SSD <= 384.0\nsquared_error =
0.064\nsamples = 93\nvalue = 6.956'),
 Text(0.5026465122453043, 0.6739130434782609, 'Cpu_name_Intel Core i7 <=
0.5\nsquared_error = 0.056\nsamples = 89\nvalue = 6.936'),
 Text(0.4839550009955532, 0.6304347826086957, 'SSD <= 64.0\nsquared_error =
0.038\nsamples = 37\nvalue = 6.842'),
 Text(0.47209132541315457, 0.5869565217391305, 'Weight_KG <=
2.475\nsquared_error = 0.024\nsamples = 10\nvalue = 6.689'),
 Text(0.46943651689121924, 0.5434782608695652, 'Gpu_brand_Nvidia <=
0.5\nsquared_error = 0.007\nsamples = 6\nvalue = 6.582'),
 Text(0.46731267007367094, 0.5, 'Weight_KG <= 2.175\nsquared_error =
0.001\nsamples = 3\nvalue = 6.512'),
 Text(0.4662507466648968, 0.45652173913043476, 'PPI <= 153.429\nsquared_error =
0.0\nsamples = 2\nvalue = 6.494'),
 Text(0.46518882325612265, 0.41304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 6.497'),
 Text(0.46731267007367094, 0.41304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 6.491'),
 Text(0.4683745934824451, 0.45652173913043476, 'squared_error = -0.0\nsamples =
1\nvalue = 6.55'),
 Text(0.4715603637087675, 0.5, 'OpSys_Others <= 0.5\nsquared_error =
0.004\nsamples = 3\nvalue = 6.651'),
 Text(0.4704984402999934, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 6.708'),
 Text(0.4726222871175417, 0.45652173913043476, 'squared_error = 0.003\nsamples =
2\nvalue = 6.623'),
 Text(0.4747461339350899, 0.5434782608695652, 'Company_Dell <=
0.5\nsquared_error = 0.007\nsamples = 4\nvalue = 6.849'),
 Text(0.47368421052631576, 0.5, 'squared_error = 0.0\nsamples = 1\nvalue =
6.925'),
 Text(0.47580805734386405, 0.5, 'Weight_KG <= 2.605\nsquared_error =
0.007\nsamples = 3\nvalue = 6.823'),
 Text(0.4747461339350899, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 6.708'),
 Text(0.4768699807526382, 0.45652173913043476, 'HDD <= 500.0\nsquared_error =
0.001\nsamples = 2\nvalue = 6.881'),

```
 Text(0.47580805734386405, 0.41304347826086957, 'squared_error = 0.0\nsamples =
1\nvalue = 6.855'),
 Text(0.47793190416141235, 0.41304347826086957, 'squared_error = -0.0\nsamples =
1\nvalue = 6.907'),
 Text(0.4958186765779518, 0.5869565217391305, 'Company_Asus <=
0.5\nsquared_error = 0.031\nsamples = 27\nvalue = 6.899'),
 Text(0.49475675316917767, 0.5434782608695652, 'Company_MSI <=
0.5\nsquared_error = 0.022\nsamples = 26\nvalue = 6.879'),
 Text(0.48875024888829893, 0.5, 'Company_Dell <= 0.5\nsquared_error =
0.022\nsamples = 21\nvalue = 6.853'),
 Text(0.4831087807791863, 0.45652173913043476, 'Weight_KG <= 1.65\nsquared_error
= 0.015\nsamples = 12\nvalue = 6.78'),
 Text(0.48005575097896064, 0.41304347826086957, 'TouchScreen <=
0.5\nsquared_error = 0.024\nsamples = 3\nvalue = 6.696'),
 Text(0.4789938275701865, 0.3695652173913043, 'squared_error = 0.0\nsamples =
1\nvalue = 6.646'),
 Text(0.4811176743877348, 0.3695652173913043, 'squared_error = 0.035\nsamples =
2\nvalue = 6.721'),
 Text(0.48616181057941193, 0.41304347826086957, 'TouchScreen <=
0.5\nsquared_error = 0.008\nsamples = 9\nvalue = 6.808'),
 Text(0.4832415212052831, 0.3695652173913043, 'SSD <= 192.0\nsquared_error =
0.008\nsamples = 7\nvalue = 6.786'),
 Text(0.48058671268334774, 0.32608695652173914, 'OpSys_Windows <=
0.5\nsquared_error = 0.009\nsamples = 3\nvalue = 6.833'),
 Text(0.4795247892745736, 0.2826086956521739, 'squared_error = 0.01\nsamples =
2\nvalue = 6.796'),
 Text(0.48164863609212183, 0.2826086956521739, 'squared_error = -0.0\nsamples =
1\nvalue = 6.907'),
 Text(0.4858963297272184, 0.32608695652173914, 'Weight_KG <= 2.1\nsquared_error
= 0.004\nsamples = 4\nvalue = 6.751'),
 Text(0.4837724829096701, 0.2826086956521739, 'Weight_KG <= 1.77\nsquared_error
= 0.0\nsamples = 2\nvalue = 6.812'),
 Text(0.482710559500896, 0.2391304347826087, 'squared_error = 0.0\nsamples =
1\nvalue = 6.801'),
 Text(0.48483440631844427, 0.2391304347826087, 'squared_error = 0.0\nsamples =
1\nvalue = 6.823'),
 Text(0.4880201765447667, 0.2826086956521739, 'OpSys_Others <=
0.5\nsquared_error = 0.001\nsamples = 2\nvalue = 6.689'),
 Text(0.48695825313599256, 0.2391304347826087, 'squared_error = 0.0\nsamples =
1\nvalue = 6.72'),
 Text(0.48908209995354085, 0.2391304347826087, 'squared_error = -0.0\nsamples =
1\nvalue = 6.658'),
 Text(0.48908209995354085, 0.3695652173913043, 'Weight_KG <= 1.89\nsquared_error
= 0.0\nsamples = 2\nvalue = 6.886'),
 Text(0.4880201765447667, 0.32608695652173914, 'squared_error = 0.0\nsamples =
1\nvalue = 6.907'),
 Text(0.490144023362315, 0.32608695652173914, 'squared_error = 0.0\nsamples =
```

```
1\nvalue = 6.866'),
 Text(0.4943917169974116, 0.45652173913043476, 'Weight_KG <= 1.61\nsquared_error
= 0.016\nsamples = 9\nvalue = 6.95'),
 Text(0.4922678701798633, 0.41304347826086957, 'TypeName_Ultrabook <=
0.5\nsquared_error = 0.0\nsamples = 2\nvalue = 7.068'),
 Text(0.49120594677108914, 0.3695652173913043, 'squared_error = 0.0\nsamples =
1\nvalue = 7.089'),
 Text(0.49332979358863743, 0.3695652173913043, 'squared_error = -0.0\nsamples =
1\nvalue = 7.047'),
 Text(0.49651556381495987, 0.41304347826086957, 'PPI <= 153.429\nsquared_error =
0.015\nsamples = 7\nvalue = 6.917'),
 Text(0.4954536404061857, 0.3695652173913043, 'Weight_KG <= 2.635\nsquared_error
= 0.009\nsamples = 6\nvalue = 6.952'),
 Text(0.4943917169974116, 0.32608695652173914, 'IPS <= 0.5\nsquared_error =
0.007\nsamples = 5\nvalue = 6.925'),
 Text(0.4922678701798633, 0.2826086956521739, 'Weight_KG <= 2.59\nsquared_error
= 0.003\nsamples = 2\nvalue = 6.836'),
 Text(0.49120594677108914, 0.2391304347826087, 'squared_error = 0.0\nsamples =
1\nvalue = 6.779'),
 Text(0.49332979358863743, 0.2391304347826087, 'squared_error = 0.0\nsamples =
1\nvalue = 6.893'),
 Text(0.49651556381495987, 0.2826086956521739, 'SSD <= 192.0\nsquared_error =
0.0\nsamples = 3\nvalue = 6.984'),
 Text(0.4954536404061857, 0.2391304347826087, 'squared_error = 0.0\nsamples =
1\nvalue = 6.956'),
 Text(0.49757748722373396, 0.2391304347826087, 'Weight_KG <= 2.09\nsquared_error
= 0.0\nsamples = 2\nvalue = 6.999'),
 Text(0.49651556381495987, 0.1956521739130435, 'squared_error = 0.0\nsamples =
1\nvalue = 7.0'),
 Text(0.4986394106325081, 0.1956521739130435, 'squared_error = 0.0\nsamples =
1\nvalue = 6.998'),
 Text(0.49651556381495987, 0.32608695652173914, 'squared_error = -0.0\nsamples =
1\nvalue = 7.086'),
 Text(0.49757748722373396, 0.3695652173913043, 'squared_error = -0.0\nsamples =
1\nvalue = 6.708'),
 Text(0.5007632574500565, 0.5, 'Weight_KG <= 2.3\nsquared_error = 0.004\nsamples
= 5\nvalue = 6.989'),
 Text(0.49970133404128225, 0.45652173913043476, 'SSD <= 192.0\nsquared_error =
0.002\nsamples = 4\nvalue = 6.964'),
 Text(0.4986394106325081, 0.41304347826086957, 'squared_error = 0.0\nsamples =
2\nvalue = 7.007'),
 Text(0.5007632574500565, 0.41304347826086957, 'squared_error = 0.0\nsamples =
2\nvalue = 6.92'),
 Text(0.5018251808588305, 0.45652173913043476, 'squared_error = 0.0\nsamples =
1\nvalue = 7.089'),
 Text(0.49688059998672596, 0.5434782608695652, 'squared_error = -0.0\nsamples =
1\nvalue = 7.408'),
```

```
 Text(0.5213380234950554, 0.6304347826086957, 'Company_Lenovo <=
0.5\nsquared_error = 0.059\nsamples = 52\nvalue = 7.003'),
 Text(0.5127098957987655, 0.5869565217391305, 'TouchScreen <= 0.5\nsquared_error
= 0.06\nsamples = 41\nvalue = 6.968'),
 Text(0.5087276830158625, 0.5434782608695652, 'PPI <= 166.799\nsquared_error =
0.06\nsamples = 33\nvalue = 7.008'),
 Text(0.5076657596070884, 0.5, 'Weight_KG <= 1.86\nsquared_error =
0.056\nsamples = 32\nvalue = 6.995'),
 Text(0.505010951085153, 0.45652173913043476, 'Weight_KG <= 1.765\nsquared_error
= 0.032\nsamples = 5\nvalue = 6.783'),
 Text(0.5028871042676047, 0.41304347826086957, 'IPS <= 0.5\nsquared_error =
0.009\nsamples = 3\nvalue = 6.914'),
 Text(0.5018251808588305, 0.3695652173913043, 'Gpu_brand_Intel <=
0.5\nsquared_error = 0.001\nsamples = 2\nvalue = 6.979'),
 Text(0.5007632574500565, 0.32608695652173914, 'squared_error = 0.0\nsamples =
1\nvalue = 6.956'),
 …]
```



[169]:
```
y_pred_DT2 = DTReg.predict(X_test_new)
y_pred_DT2
```

[169]:
```
array([6.58755001, 5.80283554, 5.69373214, 6.14632926, 7.43838353,
       7.40792432, 6.02803753, 6.96129605, 7.14435199, 5.69709349,
       7.00215595, 6.70808408, 7.46064564, 6.75576892, 7.08924316,
       7.07434541, 7.08924316, 5.79073552, 7.56475701, 7.62554629,
       7.62070509, 7.3125535 , 6.63856779, 7.15461536, 6.2126061 ,
```

```
7.64921632, 7.40792432, 6.15060277, 6.91821043, 6.69703425,
7.5822241 , 5.29330482, 6.88653164, 6.99759598, 7.01557358,
6.80128303, 7.50659178, 7.33628566, 7.58578882, 7.34245848,
7.46064564, 7.03438793, 5.70044357, 5.82600011, 6.10702289,
6.10924758, 6.72022016, 7.70930833, 7.47477218, 7.28961052,
7.78280726, 7.50604218, 5.93753621, 5.433722  , 6.39359075,
7.11441166, 6.19236249, 7.08924316, 5.95064255, 6.74405919,
7.40376987, 7.01557358, 5.50125821, 7.04664728, 6.95559261,
7.08924316, 6.02102335, 7.43248381, 6.8554088 , 5.82600011,
6.82762923, 6.51471269, 6.61875233, 6.37161185, 7.43543802,
6.62273632, 7.37775891, 8.05484022, 7.3714893 , 6.57969556,
6.96602419, 7.18159194, 7.54960917, 6.06367703, 6.93634274,
6.41836494, 6.70808408, 7.27862894, 5.98645201, 6.71538339,
6.39359075, 6.46131191, 6.65801105, 7.69824628, 6.35048494,
5.66642669, 6.37161185, 6.07534603, 5.7235851 , 7.42595366,
6.29156914, 7.16935002, 5.88332239, 7.70930833, 6.8134446 ,
7.16692778, 7.27924987, 6.07534603, 6.80121629, 5.91079664,
6.63200178, 7.00215595, 6.52502966, 7.46679948, 6.12905021,
6.58755001, 5.96357934, 7.52023456, 6.63856779, 6.52800436,
6.58755001, 7.158514  , 7.08590146, 6.30809844, 6.53813982,
7.20711886, 6.42810527, 5.7235851 , 7.9956436 , 7.74022952,
6.69207134, 6.66822825, 6.34388043, 7.23633934, 7.37713371,
6.43205546, 7.14356986, 7.16692778, 7.62578042, 7.46679948,
5.96357934, 5.63121178, 6.80128303, 6.37842618, 7.08924316,
7.80343506, 6.89770494, 5.93753621, 7.37762381, 6.47697236,
7.85353216, 7.62578042, 6.96602419, 7.30585332, 7.1500422 ,
7.08924316, 6.83720405, 7.64921632, 7.00215595, 7.0724219 ,
5.433722  , 5.98896142, 6.72934606, 6.56252884, 6.86797441,
7.47873483, 7.62505834, 6.55890677, 7.14045304, 7.30709344,
6.10255859, 7.08590146, 5.92692603, 6.74993119, 7.62364195,
6.97887428, 7.58832368, 7.64921632, 6.78105763, 6.19234204,
6.67834211, 6.90765527, 7.30585332, 6.59167373, 6.08449941,
7.16935002, 6.69207134, 7.58832368, 6.88653164, 5.7651911 ,
5.98645201, 7.08254857, 5.70044357, 7.23561914, 7.04664728,
7.30182234])
```

[170]:
```
print('R2_score: ', metrics.r2_score(y_test,y_pred_DT2))
print('MAE: ', metrics.mean_absolute_error(y_test,y_pred_DT2))
print('MSE: ',metrics.mean_squared_error(y_test,y_pred_DT2))

# Calculate R-squared scores
train_r2 = metrics.r2_score(y_train, DTReg.predict(X_train_new))
test_r2 = metrics.r2_score(y_test, y_pred_DT2)

# Print the results
print(f'Training R-squared: {train_r2}')
print(f'Test R-squared: {test_r2}')
```

```
R2_score:  0.8091050519375838
MAE:  0.1984279578203809
MSE:  0.0730270761308065
Training R-squared: 0.9961583612673353
Test R-squared: 0.8091050519375838
```

- Overfitting case
- So will be doing post-prunning

Decision trees have the potential to become overly complex and capture noise in the training data, leading to overfitting. Pruning is a technique used to prevent overfitting by removing parts of the tree that do not contribute significantly to improving the model's performance on unseen data.

Cost-complexity pruning is a specific pruning technique that involves adding a penalty term for the complexity of the tree during the pruning process. The cost-complexity parameter (usually denoted by alpha) controls the trade-off between simplicity and accuracy. A smaller alpha value results in a more complex tree, while a larger alpha value leads to a simpler tree.

The pruning path refers to the sequence of subtrees obtained by iteratively applying cost-complexity pruning. As you increase the alpha parameter, nodes with the least contribution to the model's performance are pruned, leading to a simpler tree.

The cost-complexity pruning path is the sequence of subtrees generated at different values of the cost-complexity parameter (alpha). By examining this path, you can observe how the complexity of the tree changes as you adjust the alpha parameter.

[171]:
```
path = DTReg.cost_complexity_pruning_path(X_train_new,y_train)
ccp_alphas = path.ccp_alphas
```

[172]:
```
ccp_alphas
```

[172]:
```
array([0.00000000e+00, 2.56977481e-17, 2.01384900e-12, 1.25921536e-11,
       1.39658331e-11, 1.81593908e-11, 1.08578035e-10, 1.90413356e-10,
       3.52052738e-10, 4.91049146e-10, 5.34151157e-10, 1.67264385e-09,
       1.75885874e-09, 2.06848314e-09, 2.62165042e-09, 3.72228741e-09,
       3.85508881e-09, 4.84153453e-09, 5.07372610e-09, 5.49805278e-09,
       6.03670399e-09, 6.72374014e-09, 6.97093999e-09, 8.29167813e-09,
       9.24814100e-09, 1.05268403e-08, 1.07749518e-08, 1.27531889e-08,
       1.59605995e-08, 1.65552061e-08, 1.68770720e-08, 1.83991883e-08,
       2.02572985e-08, 2.02949504e-08, 2.39815717e-08, 2.41668325e-08,
       2.62926181e-08, 3.02610142e-08, 3.03537206e-08, 3.32670950e-08,
       3.36607143e-08, 3.58476842e-08, 3.60721856e-08, 3.90442722e-08,
       4.00282623e-08, 4.60335999e-08, 4.74879668e-08, 4.86785873e-08,
       4.86910357e-08, 5.08932644e-08, 5.11643383e-08, 5.23330908e-08,
       5.37271012e-08, 5.42630695e-08, 5.53730905e-08, 5.56520158e-08,
       5.65651438e-08, 5.78521642e-08, 6.35469169e-08, 7.00542549e-08,
       7.05705843e-08, 7.15306337e-08, 7.21342811e-08, 7.77018312e-08,
       8.15640031e-08, 9.02165761e-08, 9.09601582e-08, 9.37314605e-08,
       9.69145507e-08, 1.01675494e-07, 1.08762764e-07, 1.09008558e-07,
       1.09228389e-07, 1.09742801e-07, 1.13009402e-07, 1.14150870e-07,
```

1.32559363e-07, 1.34562733e-07, 1.43311328e-07, 1.50199936e-07,
1.53136042e-07, 1.54082113e-07, 1.54787726e-07, 1.56280941e-07,
1.58780695e-07, 1.63283386e-07, 1.63337567e-07, 1.64420985e-07,
1.71573972e-07, 1.76675041e-07, 1.77531344e-07, 1.78850997e-07,
1.81054931e-07, 1.85316848e-07, 1.85638328e-07, 1.88317599e-07,
1.95116566e-07, 1.96251279e-07, 1.96409850e-07, 2.04139991e-07,
2.05605348e-07, 2.10412434e-07, 2.11266674e-07, 2.18867937e-07,
2.34762038e-07, 2.37111577e-07, 2.42013416e-07, 2.43161933e-07,
2.51474987e-07, 2.70512288e-07, 2.76276037e-07, 2.76765876e-07,
2.81234079e-07, 2.93143598e-07, 2.97168179e-07, 3.05246503e-07,
3.34775594e-07, 3.34802449e-07, 3.44603646e-07, 3.45143668e-07,
3.45521139e-07, 3.49893431e-07, 3.51011992e-07, 3.55294589e-07,
3.59174222e-07, 3.66855623e-07, 3.88336132e-07, 3.89064011e-07,
3.95050501e-07, 3.99723604e-07, 4.01974697e-07, 4.18360173e-07,
4.25115991e-07, 4.42601212e-07, 4.55189622e-07, 4.75609002e-07,
5.01415827e-07, 5.41167389e-07, 5.57683634e-07, 6.21428919e-07,
6.37333933e-07, 6.37673777e-07, 6.49051124e-07, 6.50141377e-07,
6.54774760e-07, 6.69229367e-07, 6.98153750e-07, 6.98153750e-07,
7.36896358e-07, 7.53361319e-07, 7.54901576e-07, 7.59550798e-07,
7.71404039e-07, 7.95876239e-07, 8.13237971e-07, 8.20257121e-07,
8.27645081e-07, 8.47474407e-07, 8.47474407e-07, 8.48944263e-07,
9.03302891e-07, 9.19157992e-07, 9.35016165e-07, 9.38633596e-07,
9.44401419e-07, 9.60393383e-07, 9.73672452e-07, 9.82642479e-07,
1.00775148e-06, 1.05182372e-06, 1.06244289e-06, 1.06498771e-06,
1.06746674e-06, 1.08254600e-06, 1.10735286e-06, 1.11398799e-06,
1.11703035e-06, 1.11959890e-06, 1.12408742e-06, 1.12982707e-06,
1.18575715e-06, 1.19186694e-06, 1.19636740e-06, 1.20941946e-06,
1.24755902e-06, 1.25914044e-06, 1.25949972e-06, 1.26052708e-06,
1.30812667e-06, 1.35653339e-06, 1.39713467e-06, 1.44281515e-06,
1.47024955e-06, 1.49653563e-06, 1.51204836e-06, 1.54091192e-06,
1.57444504e-06, 1.69855122e-06, 1.71437809e-06, 1.73798038e-06,
1.74953584e-06, 1.75024002e-06, 1.75507429e-06, 1.78403879e-06,
1.80234509e-06, 1.84246838e-06, 1.88365006e-06, 1.89239758e-06,
1.90553729e-06, 1.91094760e-06, 1.93766337e-06, 1.95491831e-06,
1.96739692e-06, 1.96848255e-06, 2.07565210e-06, 2.07996271e-06,
2.08569657e-06, 2.09713587e-06, 2.17498717e-06, 2.22506360e-06,
2.23863273e-06, 2.23987439e-06, 2.24384033e-06, 2.26074679e-06,
2.27649148e-06, 2.28488227e-06, 2.34054420e-06, 2.35955089e-06,
2.36540015e-06, 2.41478402e-06, 2.48650287e-06, 2.50735206e-06,
2.60385004e-06, 2.60737169e-06, 2.65525276e-06, 2.66704626e-06,
2.66923816e-06, 2.69011621e-06, 2.69826611e-06, 2.70300183e-06,
2.70642146e-06, 2.71852973e-06, 2.73119793e-06, 2.73996876e-06,
2.75489866e-06, 2.77139259e-06, 2.87858467e-06, 2.93485988e-06,
2.93911484e-06, 2.94106043e-06, 3.09822763e-06, 3.10435599e-06,
3.11418892e-06, 3.16705505e-06, 3.17816754e-06, 3.18549922e-06,
3.20217328e-06, 3.24442095e-06, 3.25348737e-06, 3.29259740e-06,
3.34807353e-06, 3.35520226e-06, 3.37231973e-06, 3.37679051e-06,

3.39679205e-06, 3.44012499e-06, 3.44393439e-06, 3.45411929e-06,
3.49841999e-06, 3.50453813e-06, 3.51876825e-06, 3.53691241e-06,
3.56697419e-06, 3.61556926e-06, 3.65147640e-06, 3.67099704e-06,
3.69373861e-06, 3.75694543e-06, 3.83012343e-06, 3.83293056e-06,
3.85318614e-06, 3.86173326e-06, 3.89712274e-06, 3.96485410e-06,
4.01044497e-06, 4.01848463e-06, 4.02947040e-06, 4.05526958e-06,
4.06841934e-06, 4.10670451e-06, 4.12361412e-06, 4.19769743e-06,
4.25480240e-06, 4.26526992e-06, 4.29623339e-06, 4.34597386e-06,
4.36549375e-06, 4.38397591e-06, 4.40001029e-06, 4.41142198e-06,
4.42106528e-06, 4.45015964e-06, 4.52741888e-06, 4.54528821e-06,
4.56415195e-06, 4.66505036e-06, 4.67909918e-06, 4.75621332e-06,
4.84827150e-06, 4.94413105e-06, 4.96643471e-06, 4.97470592e-06,
5.01846214e-06, 5.01846214e-06, 5.03880687e-06, 5.05942188e-06,
5.13118831e-06, 5.15370659e-06, 5.18683544e-06, 5.23439261e-06,
5.45771138e-06, 5.48889055e-06, 5.58154477e-06, 5.69863386e-06,
5.75088821e-06, 5.76158151e-06, 5.82041131e-06, 5.82928854e-06,
5.86046928e-06, 5.87424157e-06, 5.89226396e-06, 6.02375378e-06,
6.04384335e-06, 6.04873244e-06, 6.16317347e-06, 6.19947147e-06,
6.21258856e-06, 6.27341419e-06, 6.28837366e-06, 6.33601433e-06,
6.39005184e-06, 6.39225892e-06, 6.69387695e-06, 6.71058636e-06,
6.75607227e-06, 6.78456714e-06, 6.79124698e-06, 6.81775071e-06,
7.02297287e-06, 7.06847412e-06, 7.10395453e-06, 7.10477515e-06,
7.10632801e-06, 7.19605816e-06, 7.23193772e-06, 7.23957343e-06,
7.24977841e-06, 7.25556039e-06, 7.28653527e-06, 7.35980711e-06,
7.36109233e-06, 7.36511000e-06, 7.37169169e-06, 7.39688761e-06,
7.43623749e-06, 7.45317302e-06, 7.57309283e-06, 7.58589472e-06,
7.69227905e-06, 7.73107004e-06, 7.79078750e-06, 7.91976934e-06,
7.92207886e-06, 8.01179476e-06, 8.07165537e-06, 8.14780068e-06,
8.16639752e-06, 8.18674961e-06, 8.18829465e-06, 8.25133840e-06,
8.38690567e-06, 8.38940271e-06, 8.41152317e-06, 8.44601767e-06,
8.48792549e-06, 8.57135206e-06, 8.64973384e-06, 8.77209001e-06,
8.84715702e-06, 8.98494306e-06, 9.00001580e-06, 9.04450806e-06,
9.15195876e-06, 9.16933789e-06, 9.27664757e-06, 9.44211230e-06,
9.46373580e-06, 9.46937484e-06, 9.47145962e-06, 9.50980007e-06,
9.58049270e-06, 9.58492391e-06, 9.60357869e-06, 9.63794984e-06,
9.72074875e-06, 9.75974313e-06, 9.79612204e-06, 9.80310402e-06,
1.00381279e-05, 1.01300013e-05, 1.01814779e-05, 1.01986184e-05,
1.02407614e-05, 1.03356227e-05, 1.03578777e-05, 1.03586353e-05,
1.07535802e-05, 1.07671260e-05, 1.08091891e-05, 1.12083070e-05,
1.12342553e-05, 1.12659633e-05, 1.13170017e-05, 1.14009084e-05,
1.14343366e-05, 1.15758588e-05, 1.15898934e-05, 1.16269144e-05,
1.16646714e-05, 1.17760846e-05, 1.18154752e-05, 1.20557643e-05,
1.21181133e-05, 1.21521585e-05, 1.21536756e-05, 1.23093087e-05,
1.24018663e-05, 1.24921733e-05, 1.25074475e-05, 1.27582644e-05,
1.28210060e-05, 1.29923721e-05, 1.30738974e-05, 1.31282431e-05,
1.31605345e-05, 1.32787072e-05, 1.34755167e-05, 1.34860550e-05,
1.36377251e-05, 1.36725944e-05, 1.40102424e-05, 1.40569084e-05,

1.41537311e-05, 1.42286862e-05, 1.43858230e-05, 1.44038760e-05,
1.44967060e-05, 1.47261907e-05, 1.48174496e-05, 1.48963210e-05,
1.48976641e-05, 1.49520028e-05, 1.51108691e-05, 1.52458693e-05,
1.53536175e-05, 1.54576126e-05, 1.55729069e-05, 1.58142984e-05,
1.58299319e-05, 1.59915883e-05, 1.60073008e-05, 1.60322833e-05,
1.62678112e-05, 1.62882344e-05, 1.63530191e-05, 1.63668644e-05,
1.66488888e-05, 1.66862940e-05, 1.67766640e-05, 1.67855894e-05,
1.69625440e-05, 1.72637422e-05, 1.74483365e-05, 1.75434134e-05,
1.78053592e-05, 1.79896680e-05, 1.82587563e-05, 1.85330276e-05,
1.85768342e-05, 1.85904585e-05, 1.87105960e-05, 1.88290645e-05,
1.88365160e-05, 1.90613619e-05, 1.91555042e-05, 1.92619763e-05,
1.97128941e-05, 1.98517814e-05, 1.99276169e-05, 1.99882500e-05,
2.00270459e-05, 2.02088440e-05, 2.05204544e-05, 2.06002629e-05,
2.06565233e-05, 2.08233650e-05, 2.10011964e-05, 2.10206773e-05,
2.10485973e-05, 2.13339286e-05, 2.15712454e-05, 2.16580883e-05,
2.17426942e-05, 2.19284031e-05, 2.21316658e-05, 2.25440820e-05,
2.26635402e-05, 2.27484369e-05, 2.27528995e-05, 2.28348978e-05,
2.28824173e-05, 2.28842492e-05, 2.29631309e-05, 2.30962077e-05,
2.36444882e-05, 2.38722248e-05, 2.46971395e-05, 2.51855727e-05,
2.55798939e-05, 2.55967450e-05, 2.56779517e-05, 2.58084297e-05,
2.58450692e-05, 2.59521648e-05, 2.60478636e-05, 2.61392966e-05,
2.61911885e-05, 2.68455995e-05, 2.72336985e-05, 2.75755686e-05,
2.76804702e-05, 2.77172031e-05, 2.77217136e-05, 2.77334540e-05,
2.77663322e-05, 2.87517570e-05, 2.89551508e-05, 2.91318916e-05,
2.92054728e-05, 2.92409018e-05, 2.95141983e-05, 2.96086983e-05,
2.99029086e-05, 3.07269751e-05, 3.09281832e-05, 3.10875715e-05,
3.11973774e-05, 3.13076758e-05, 3.15075898e-05, 3.15898607e-05,
3.18557289e-05, 3.25489775e-05, 3.31231026e-05, 3.32371077e-05,
3.34254684e-05, 3.36341787e-05, 3.39480631e-05, 3.40200240e-05,
3.40758146e-05, 3.41215653e-05, 3.43380733e-05, 3.44031533e-05,
3.44897026e-05, 3.48248562e-05, 3.49680424e-05, 3.49797478e-05,
3.50989926e-05, 3.52826199e-05, 3.53274291e-05, 3.53852250e-05,
3.54763720e-05, 3.66301914e-05, 3.66573028e-05, 3.77411994e-05,
3.77589795e-05, 3.79291954e-05, 3.81479909e-05, 3.83532705e-05,
3.85630677e-05, 3.85999326e-05, 3.86670771e-05, 3.90418530e-05,
4.00499143e-05, 4.00728384e-05, 4.08917669e-05, 4.11100691e-05,
4.13177332e-05, 4.14039665e-05, 4.14646532e-05, 4.16503736e-05,
4.17388493e-05, 4.17903604e-05, 4.24152749e-05, 4.24891350e-05,
4.26385448e-05, 4.27427778e-05, 4.29595626e-05, 4.32202943e-05,
4.38336718e-05, 4.38530115e-05, 4.39501562e-05, 4.40207018e-05,
4.42836624e-05, 4.49324315e-05, 4.58473481e-05, 4.59053589e-05,
4.63556333e-05, 4.64379741e-05, 4.64897575e-05, 4.74319449e-05,
4.85988583e-05, 4.88207338e-05, 4.89269232e-05, 4.96733875e-05,
5.04745500e-05, 5.09633412e-05, 5.18027604e-05, 5.29850869e-05,
5.30417221e-05, 5.31445927e-05, 5.36116482e-05, 5.40039493e-05,
5.50233370e-05, 5.50820055e-05, 5.51977117e-05, 5.54763718e-05,
5.57018345e-05, 5.60553062e-05, 5.70413920e-05, 5.75932052e-05,

5.82126216e-05, 5.87654868e-05, 5.90051851e-05, 5.93255748e-05,
6.00198002e-05, 6.02734795e-05, 6.06079712e-05, 6.09083992e-05,
6.14960970e-05, 6.19676110e-05, 6.28737395e-05, 6.28948116e-05,
6.31901690e-05, 6.40992144e-05, 6.45005656e-05, 6.48903054e-05,
6.52947346e-05, 6.53468525e-05, 6.63394370e-05, 6.69228658e-05,
6.77023654e-05, 6.87966512e-05, 6.90683824e-05, 6.96293932e-05,
6.99463852e-05, 7.00158372e-05, 7.02121353e-05, 7.05980569e-05,
7.12961229e-05, 7.15292933e-05, 7.16040865e-05, 7.22541356e-05,
7.31663617e-05, 7.51438705e-05, 7.66689807e-05, 7.67742186e-05,
7.70348283e-05, 7.90332532e-05, 7.99899242e-05, 8.20437857e-05,
8.27400126e-05, 8.58442538e-05, 8.65913076e-05, 8.66423727e-05,
8.80338153e-05, 8.94733971e-05, 8.94825950e-05, 8.96962669e-05,
9.00640156e-05, 9.03169883e-05, 9.31735868e-05, 9.47516714e-05,
9.59425485e-05, 9.81988585e-05, 9.85249524e-05, 1.00920175e-04,
1.02248992e-04, 1.02396105e-04, 1.04031052e-04, 1.05753605e-04,
1.07729690e-04, 1.08207158e-04, 1.08794004e-04, 1.09143959e-04,
1.12885762e-04, 1.13112960e-04, 1.13126650e-04, 1.13326420e-04,
1.16842197e-04, 1.18400917e-04, 1.18436100e-04, 1.21712228e-04,
1.23763346e-04, 1.24780653e-04, 1.24961483e-04, 1.25202670e-04,
1.25359791e-04, 1.25808157e-04, 1.28445965e-04, 1.30399531e-04,
1.30673937e-04, 1.31480949e-04, 1.35829017e-04, 1.37807725e-04,
1.39434475e-04, 1.41265569e-04, 1.42090989e-04, 1.42286245e-04,
1.42327562e-04, 1.42889988e-04, 1.44366305e-04, 1.45808056e-04,
1.47511961e-04, 1.47679495e-04, 1.53325617e-04, 1.53835664e-04,
1.53958880e-04, 1.54180741e-04, 1.54520648e-04, 1.55177598e-04,
1.57294559e-04, 1.61735572e-04, 1.64185244e-04, 1.64299420e-04,
1.66015725e-04, 1.66779584e-04, 1.67238635e-04, 1.80835535e-04,
1.87773074e-04, 1.93043396e-04, 1.95464316e-04, 1.97490085e-04,
2.00758203e-04, 2.08973960e-04, 2.10900505e-04, 2.17574388e-04,
2.21161311e-04, 2.43325921e-04, 2.43483712e-04, 2.49865784e-04,
2.50176508e-04, 2.52439667e-04, 2.53731999e-04, 2.59472562e-04,
2.60691413e-04, 2.62100643e-04, 2.64284607e-04, 2.67975759e-04,
2.70138000e-04, 2.72043867e-04, 2.73110776e-04, 2.73812831e-04,
2.76997444e-04, 2.77295067e-04, 2.80134094e-04, 2.83639430e-04,
2.86296261e-04, 2.91660534e-04, 3.03476321e-04, 3.09262632e-04,
3.12867213e-04, 3.13659811e-04, 3.15074571e-04, 3.16558886e-04,
3.19510633e-04, 3.23859158e-04, 3.27526491e-04, 3.30388174e-04,
3.30669797e-04, 3.30810449e-04, 3.48166121e-04, 3.66306307e-04,
3.70679090e-04, 3.86829301e-04, 3.91750421e-04, 4.19078867e-04,
4.22411302e-04, 4.25871150e-04, 4.26271899e-04, 4.31630565e-04,
4.33818711e-04, 4.36981348e-04, 4.41128600e-04, 4.43452212e-04,
4.45881033e-04, 4.53650365e-04, 4.62726042e-04, 4.79996971e-04,
5.06995788e-04, 5.11098173e-04, 5.32952729e-04, 5.64593917e-04,
5.73453339e-04, 6.42278205e-04, 6.54734497e-04, 6.62737472e-04,
6.83581933e-04, 7.03697061e-04, 7.37526036e-04, 7.46898075e-04,
7.83961066e-04, 8.27853566e-04, 9.14680960e-04, 9.26663023e-04,
9.32963631e-04, 9.62120334e-04, 1.01529334e-03, 1.09468823e-03,

```
       1.24825374e-03, 1.56897260e-03, 1.59168172e-03, 1.72397669e-03,
       1.76664175e-03, 1.93053895e-03, 2.42268753e-03, 2.70436100e-03,
       3.38248038e-03, 4.00799306e-03, 4.60133624e-03, 8.27533525e-03,
       1.03189885e-02, 1.96275595e-02, 2.48399387e-02, 4.47194619e-02,
       1.82175896e-01])
```

[173]:
```
alphaList=[]

for alpha in ccp_alphas:
    reg = DecisionTreeRegressor(ccp_alpha=alpha, random_state=0)
    reg.fit(X_train_new,y_train)
    alphaList.append(reg)
```

[174]:
```
alphaList
```

[174]:
```
[DecisionTreeRegressor(random_state=0),
 DecisionTreeRegressor(ccp_alpha=2.5697748128755883e-17, random_state=0),
 DecisionTreeRegressor(ccp_alpha=2.0138490031691803e-12, random_state=0),
 DecisionTreeRegressor(ccp_alpha=1.2592153560571671e-11, random_state=0),
 DecisionTreeRegressor(ccp_alpha=1.3965833111231348e-11, random_state=0),
 DecisionTreeRegressor(ccp_alpha=1.8159390808429732e-11, random_state=0),
 DecisionTreeRegressor(ccp_alpha=1.0857803545150091e-10, random_state=0),
 DecisionTreeRegressor(ccp_alpha=1.90413355968792e-10, random_state=0),
 DecisionTreeRegressor(ccp_alpha=3.520527377757975e-10, random_state=0),
 DecisionTreeRegressor(ccp_alpha=4.910491462918104e-10, random_state=0),
 DecisionTreeRegressor(ccp_alpha=5.341511569002326e-10, random_state=0),
 DecisionTreeRegressor(ccp_alpha=1.6726438502289894e-09, random_state=0),
 DecisionTreeRegressor(ccp_alpha=1.7588587415932922e-09, random_state=0),
 DecisionTreeRegressor(ccp_alpha=2.0684831416523e-09, random_state=0),
 DecisionTreeRegressor(ccp_alpha=2.6216504201613897e-09, random_state=0),
 DecisionTreeRegressor(ccp_alpha=3.7222874138020764e-09, random_state=0),
 DecisionTreeRegressor(ccp_alpha=3.855088814781194e-09, random_state=0),
 DecisionTreeRegressor(ccp_alpha=4.841534530328934e-09, random_state=0),
 DecisionTreeRegressor(ccp_alpha=5.073726103109027e-09, random_state=0),
 DecisionTreeRegressor(ccp_alpha=5.498052780982521e-09, random_state=0),
 DecisionTreeRegressor(ccp_alpha=6.036703986197447e-09, random_state=0),
 DecisionTreeRegressor(ccp_alpha=6.7237401372940494e-09, random_state=0),
 DecisionTreeRegressor(ccp_alpha=6.9709399916115266e-09, random_state=0),
 DecisionTreeRegressor(ccp_alpha=8.2916781273062e-09, random_state=0),
 DecisionTreeRegressor(ccp_alpha=9.248140995224309e-09, random_state=0),
 DecisionTreeRegressor(ccp_alpha=1.0526840300395423e-08, random_state=0),
 DecisionTreeRegressor(ccp_alpha=1.0774951820874391e-08, random_state=0),
 DecisionTreeRegressor(ccp_alpha=1.2753188884686742e-08, random_state=0),
 DecisionTreeRegressor(ccp_alpha=1.5960599512717263e-08, random_state=0),
 DecisionTreeRegressor(ccp_alpha=1.655520612540866e-08, random_state=0),
 DecisionTreeRegressor(ccp_alpha=1.687707203967946e-08, random_state=0),
 DecisionTreeRegressor(ccp_alpha=1.839918831718329e-08, random_state=0),
```

DecisionTreeRegressor(ccp_alpha=2.025729853983091e-08, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.0294950372858637e-08, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.39815716892997e-08, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.4166832543977668e-08, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.629261813051695e-08, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.026101422727967e-08, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.035372061394769e-08, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.326709497387202e-08, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.366071434605001e-08, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.584768417651707e-08, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.60721855889542e-08, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.904427219792327e-08, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.002826225914525e-08, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.603359989841081e-08, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.748796683190554e-08, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.8678587278824737e-08, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.86910357177289e-08, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.089326435065551e-08, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.116433831883022e-08, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.233309075302429e-08, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.372710115420928e-08, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.4263069533804636e-08, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.5373090535571115e-08, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.565201576969625e-08, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.656514383110604e-08, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.785216415063852e-08, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.35469168562033e-08, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.005425485597782e-08, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.057058429048004e-08, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.153063369031388e-08, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.213428114341433e-08, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.770183117783817e-08, random_state=0),
DecisionTreeRegressor(ccp_alpha=8.156400308863384e-08, random_state=0),
DecisionTreeRegressor(ccp_alpha=9.021657607343861e-08, random_state=0),
DecisionTreeRegressor(ccp_alpha=9.096015815983924e-08, random_state=0),
DecisionTreeRegressor(ccp_alpha=9.373146054692935e-08, random_state=0),
DecisionTreeRegressor(ccp_alpha=9.691455072333983e-08, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.0167549380695192e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.0876276402872188e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.090085582290656e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.092283893134845e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.0974280100213712e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.1300940199866723e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.1415087014430976e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.3255936251751175e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.345627330904216e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.4331132836440197e-07, random_state=0),

```
DecisionTreeRegressor(ccp_alpha=1.5019993575194774e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.5313604188372953e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.5408211288425184e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.5478772609165863e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.5628094114617647e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.5878069497907937e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.6328338635856203e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.6333756652705208e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.644209851043276e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.715739715092458e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.7667504144258883e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.7753134370094415e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.7885099653079323e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.8105493076180977e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.853168482273263e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.8563832778880285e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.8831759890761272e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.9511656605457963e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.9625127893038078e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.964098499046943e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.0413999076564615e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.056053478165365e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.1041243372529306e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.1126667447817132e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.1886793689855344e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.3476203802721607e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.3711157658249022e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.420134155654301e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.431619334899566e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.5147498703627847e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.705122884914425e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.762760372672636e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.767658762102318e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.812340787768627e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.931435975681485e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.9716817860467423e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.052465029641349e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.3477559384569367e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.3480244938658885e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.446036460164581e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.4514366802307593e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.455211388403876e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.498934309062625e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.510119916391483e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.5529458892418523e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.591742219601528e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.6685562288516806e-07, random_state=0),
```

```
DecisionTreeRegressor(ccp_alpha=3.88336132334294e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.890640111885 3017e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.950505008240672e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.997236037236881e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.0197469654758587e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.18360173434497e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.251159906766774e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.4260121172109016e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.5518962168511057e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.756090023660993e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.014158273171734e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.41167388774606e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.576836337667197e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.214289186525807e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.373339333129251e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.376737773332401e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.490511244587396e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.501413765940463e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.547747602125578e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.692293670886709e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.981537502858614e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.98153750324408e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.36896358365948e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.53361318535581e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.54901575662968e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.595507980371434e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.714040387921493e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.958762388656619e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=8.132379707819693e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=8.202571214386973e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=8.276450811847816e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=8.474744070630835e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=8.474744070952056e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=8.489442632050473e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=9.03302891183662e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=9.191579922533659e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=9.350161650712707e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=9.386335958304692e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=9.444014186244909e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=9.603933828940918e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=9.736724521411412e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=9.826424791930009e-07, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.0077514834190187e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.0518237203394306e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.0624428866869743e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.0649877113249127e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.0674667379452234e-06, random_state=0),
```

DecisionTreeRegressor(ccp_alpha=1.0825459990592554e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.1073528608231308e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.113987987377006e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.117030350542903e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.1195988990913586e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.1240874199230374e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.1298270692478831e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.1857571467174254e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.191866937643378e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.196367397179861e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.2094194564838187e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.247559019453172e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.2591404407440472e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.25949971763706e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.2605270793071783e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.3081266683554764e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.3565333879397798e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.3971346656760665e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.4428151534020639e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.4702495534413033e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.4965356295675902e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.5120483555558235e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.5409119210938873e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.574445040871713e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.6985512248278356e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.714378088736487e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.7379803783605183e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.749535838852549e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.7502400218164477e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.7550742920206058e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.784038786139011e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.8023450880375252e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.8424683848382893e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.8836500626112268e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.8923975833081714e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.9055372926841798e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.9109476031960155e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.9376633748443843e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.9549183067781305e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.9673969177103186e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.968482545612943e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.0756521035955708e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.079962705573406e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.08569656560033e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.097135865560494e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.174987168907754e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.2250635951539197e-06, random_state=0),

```
DecisionTreeRegressor(ccp_alpha=2.238632731086529e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.2398743871905632e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.2438403323967204e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.2607467895736704e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.2764914786624403e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.2848822741285557e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.340544195692741e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.3595508901854877e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.3654001469268986e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.414784021090274e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.4865028656082624e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.5073520614343504e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.6038500436749484e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.6073716877419264e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.6552527637107967e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.667046256301982e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.6692381582562765e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.690116209826761e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.6982661090222937e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.7030018268526355e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.7064214614213174e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.7185297301364955e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.7311979269571626e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.739968755235409e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.7548986635299987e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.771392589109168e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.8785846731596056e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.93485987979612e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.939114835992207e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.9410604289688707e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.0982276261632994e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.1043559861214388e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.1141889164314864e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.167055045616068e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.1781675365941054e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.1854992205460385e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.2021732761353724e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.2444209471252547e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.25348736903224e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.2925973950540877e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.34807352710952e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.3552022579496304e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.3723197334124197e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.3767905097562918e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.3967920518947314e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.4401249881020695e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.443934387464227e-06, random_state=0),
```

```
DecisionTreeRegressor(ccp_alpha=3.454119293283893e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.498419988074166e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.5045381280699974e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.5187682466716466e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.5369124092900596e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.5669741888889457e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.615569259784089e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.6514764020739433e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.6709970364554013e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.6937386110444496e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.7569454317019114e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.830123432995215e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.8329305561415626e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.853186144335645e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.86173325615202e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.897122741757228e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.9648540996297414e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.0104449676605094e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.018484625273653e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.029470398831128e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.055269581445369e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.068419336559378e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.106704509188483e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.123614119023004e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.1976974270817655e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.254802402582694e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.265269917562865e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.296233390974797e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.345973858266515e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.365493746291137e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.383975914333078e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.400010285540596e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.411421979910052e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.421065276572357e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.450159642227231e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.527418884185627e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.545288206948122e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.564151954618717e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.665050359019787e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.679099178742774e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.756213322176052e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.84827149556175e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.944131049263058e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.966434708215737e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.974705916859591e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.018462142712177e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.018462142718602e-06, random_state=0),
```

```
DecisionTreeRegressor(ccp_alpha=5.0388068713384845e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.0594218757030925e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.131188308297216e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.153706594884026e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.186835437277125e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.234392611780972e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.45771138030081e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.488890549248918e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.581544773396508e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.6986338641475565e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.750888212040653e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.761581511341941e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.820411312649663e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.829288544201485e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.8604692755148585e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.874241569351067e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.892263957061059e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.023753781498609e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.043843352660943e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.0487324434889306e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.163173467909575e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.1994714655989106e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.212588560603223e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.273414189055513e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.288373664538796e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.336014325559119e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.390051844923251e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.392258921140839e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.693876951241833e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.710586356216641e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.756072271341881e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.784567138425819e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.791246983590666e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.817750714549025e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.022972869231516e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.0684741233426236e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.103954526507831e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.104775154593307e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.106328012161779e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.1960581645551e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.231937720215971e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.23957343308889e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.249778410558745e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.255560389188603e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.286535272673742e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.359807108966693e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.361092325850381e-06, random_state=0),
```

```
DecisionTreeRegressor(ccp_alpha=7.365110002785346e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.371691689481097e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.3968876127681765e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.436237489807914e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.4531730206333266e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.573092825294787e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.585894719918134e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.692279052003197e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.731070041368597e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.79078750327997e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.919769338482304e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.922078857166697e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=8.01179476235584e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=8.071655365594012e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=8.147800677984172e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=8.166397524890068e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=8.186749606714997e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=8.188294645268196e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=8.251338396470587e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=8.386905669594242e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=8.389402712950737e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=8.411523172078806e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=8.446017665223724e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=8.48792549290319e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=8.57135206365913e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=8.649733839418 22e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=8.772090014860112e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=8.847157021917763e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=8.984943064029928e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=9.000015801789871e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=9.044508064047678e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=9.151958755080038e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=9.169337891531864e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=9.276647569091273e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=9.442112302638709e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=9.463735798643523e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=9.469374838907398e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=9.471459616159171e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=9.509800070397588e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=9.580492696416207e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=9.58492390545545e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=9.603578694375002e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=9.637949835123105e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=9.720748749845411e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=9.759743130552756e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=9.796122035584804e-06, random_state=0),
DecisionTreeRegressor(ccp_alpha=9.803104017271332e-06, random_state=0),
```

DecisionTreeRegressor(ccp_alpha=1.0038127901511601e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.0130001307259501e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.0181477925705746e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.0198618389590226e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.0240761443190816e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.033562270822159e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.0357877651458459e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.035863527911166e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.0753580193272344e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.0767125993857234e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.0809189085104829e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.1208307010854998e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.1234255324067405e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.126596327232157e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.1317001679076246e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.1400908388180397e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.1434336626595232e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.1575858819927827e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.158989343660393e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.1626914449030628e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.1664671383484009e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.1776084631872748e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.1815475203410287e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.2055764281043884e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.2118113289751683e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.215215851400045e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.2153675612330075e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.2309308667857279e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.2401866295332471e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.2492173289112589e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.250744753180703e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.2758264420778617e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.2821006019374682e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.2992372075003715e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.3073897374007205e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.3128243127606725e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.3160534452968826e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.3278707210511704e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.3475516716915572e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.348605499823569e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.3637725133005016e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.3672594412532293e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.4010242421938765e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.4056908376296106e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.4153731062108756e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.4228686168379996e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.4385822950195152e-05, random_state=0),

```
DecisionTreeRegressor(ccp_alpha=1.4403875961686732e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.4496706033044935e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.4726190735556007e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.481744956013997e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.489632101015158e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.4897664077497485e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.4952002838649815e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.5110869129345866e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.5245869329124436e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.5353617532430804e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.545761255219706e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.5572906854535373e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.581429840237732e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.5829931918414203e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.5991588346430555e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.6007300802808142e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.603228327293491e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.626781123051414e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.6288234355736068e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.6353019089728968e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.636686441776675e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.6648888832772463e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.6686293989868348e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.6776664012961792e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.678558940321146e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.6962544049021838e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.726374223274777e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.744833647553969e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.754341338205464e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.7805359209484968e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.798966797329467e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.8258756313392716e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.853302758141646e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.8576834221895105e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.8590458489057013e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.8710595960583514e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.8829064518917527e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.8836516048761432e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.906136186724588e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.9155504233381647e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.92619763196307e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.9712894100028834e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.9851781427989932e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.9927616882705515e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=1.998824997893148e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.002704591727522e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.020884400286625e-05, random_state=0),
```

```
DecisionTreeRegressor(ccp_alpha=2.052045440059979e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.0600262888853457e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.0656523323724067e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.0823365023384264e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.100119638021267e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.1020677304065235e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.1048597311784726e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.1333928586232532e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.1571245388060702e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.1658088259973018e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.1742694203656988e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.1928403067222327e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.2131665775384573e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.2544081973030502e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.2663540165720892e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.2748436850993598e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.275289949339308e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.283489775381985e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.2882417293316274e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.288424924233833e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.2963130931020214e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.3096207674479112e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.364448823969017e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.387222484162917e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.469713954642925e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.5185572651990833e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.557989393209492e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.559674497373555e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.56779516967083e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.5808429652937178e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.58450691710587e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.5952164776081036e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.6047863586795814e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.613929664142146e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.6191188507206732e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.6845599493826164e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.7233698488770148e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.7575568605680118e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.7680470242768044e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.771720308673654e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.7721713610667588e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.7733454031260702e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.7766332192127225e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.8751757042716797e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.8955150830097242e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.9131891588622406e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.9205472832721943e-05, random_state=0),
```

```
DecisionTreeRegressor(ccp_alpha=2.924090178589239e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.9514198325740937e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.960869831091375e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=2.9902908562490738e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.072697506487916e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.092818317304072e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.1087571514691375e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.119737738966632e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.130767584353596e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.150758980560444e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.1589860734100305e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.1855728883024425e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.2548977502741335e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.312310263351258e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.323710768740651e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.342546839475881e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.3634178721984684e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.394806313533927e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.402002402533114e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.407581459051843e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.412156526436241e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.433807330028473e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.4403153276849454e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.448970264380744e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.4824856213392637e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.496804237676314e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.497974784805259e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.509899255903249e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.5282619900521565e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.532742910203058e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.53852250166277e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.5476371998254815e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.663019140943946e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.665730279565244e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.774119942640177e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.7758979515993686e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.792919538099322e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.8147990941411236e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.8353270480424486e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.856306767979405e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.859993262750497e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.866707711008652e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=3.904185301387699e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.004991426243942e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.007283842164685e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.089176690304007e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.111006914594587e-05, random_state=0),
```

```
DecisionTreeRegressor(ccp_alpha=4.13177331660756e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.140396647700113e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.146465317370901e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.165037360003821e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.173884928893175e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.179036039742434e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.2415274854081816e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.248913504806605e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.263854480658753e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.274277775803254e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.295956262161671e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.322029434437571e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.383367179829567e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.385301145317864e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.395015619804662e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.402070178109431e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.428366236338385e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.4932431508244795e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.584734810069937e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.59053588898731e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.635563330299738e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.643797409743656e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.648975753971658e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.743194491074203e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.8598858325268565e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.882073384643755e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.892692316919507e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=4.967338747868921e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.047454995217549e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.0963341188050096e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.1802760354893105e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.298508693917726e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.304172206982316e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.314459271671008e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.361164822786452e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.400394931061301e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.502333695702361e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.5082005511578985e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.519771170810012e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.547637181021889e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.5701834468106154e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.6055306166737536e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.7041391986482334e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.7593205206994886e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.8212621610052925e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.876548682161526e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=5.900518506891063e-05, random_state=0),
```

```
DecisionTreeRegressor(ccp_alpha=5.932557477783258e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.001980020367995e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.027347953238183e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.0607971201708517e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.0908399158416234e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.149609702544209e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.196761098722106e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.287373949755025e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.289481160240854e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.319016895183964e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.409921444248175e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.45005655935015e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.489030542267066e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.529473461033375e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.534685253957227e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.633943702624457e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.692286584895202e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.770236542509776e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.879665116551689e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.906838235801036e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.96293932434295e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=6.994638522024333e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.001583719431028e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.021213528065775e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.059805688753646e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.129612287146795e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.152929333627758e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.160408652761007e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.225413564215718e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.316636166166188e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.514387046340234e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.666898066671425e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.677421862294271e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.703482829751273e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.903325322935925e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=7.998992416539141e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=8.204378570312867e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=8.274001255798556e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=8.584425378142346e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=8.659130764361838e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=8.664237270211152e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=8.80338153311734e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=8.947339711145463e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=8.948259498033068e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=8.969626691567696e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=9.006401556857217e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=9.031698826558137e-05, random_state=0),
```

```
DecisionTreeRegressor(ccp_alpha=9.317358677147684e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=9.475167143102211e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=9.59425484515343e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=9.819885850252688e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=9.85249524152754e-05, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00010092017482749127, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00010224899196700351, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00010239610506346892, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00010403105184654958, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00010575360540536315, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00010772969018731549, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00010820715763390428, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00010879400400445897, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00010914395854154905, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00011288857617391756, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00011311295997817379, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00011312665016252019, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00011332641986763 91, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00011684219719210691, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00011840091703047299, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00011843610007071306, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00012171222842 91673, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00012376334593 92279, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00012478065304 68994, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00012496148279599726, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00012520266957340994, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00012535979119656 89, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00012580815739623478, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00012844596501919 71, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00013039539530631 73886, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00013067393700594 96, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00013148094896666052, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00013582901731545 54, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00013780772506073 86, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00013943447454159061, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00014126556889247388, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00014209098916 2386, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00014228624523845927, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00014232756243079284, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00014288998772488815, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00014436630496097677, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00014580805630565034, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00014751196111813876, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00014767949477603233, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00015332561737153 33, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00015383566447604066, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00015395887974112106, random_state=0),
```

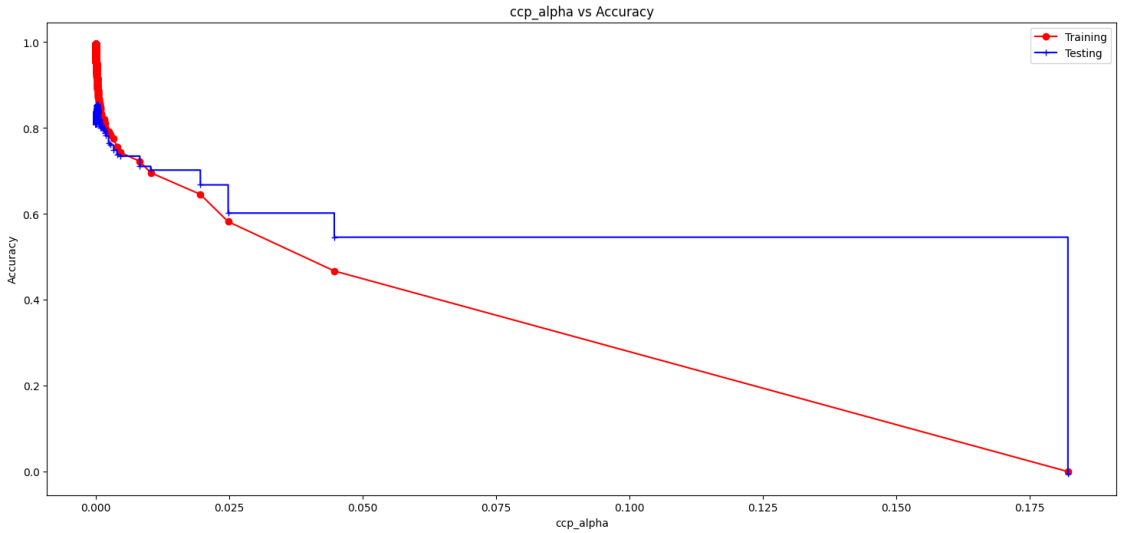
```
DecisionTreeRegressor(ccp_alpha=0.00015418074118209234, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00015452064791708072, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00015517759791202864, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00015729455949049572, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0001617355715313759, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0001641852444795216, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00016429941974433257, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00016601572454690591, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00016677958364856574, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00016723863474485314, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00018083553500479932, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00018777307389628288, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00019304339584122002, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00019546431561496446, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00019749008467934267, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00020075820278929338, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00020897396004107235, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00021090050051432409, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00021757438796624095, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00022116131112426903, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00024332592091418634, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00024348371194029699, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00024986578389237174, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0002501765081459903, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0002524396672352312, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0002537319986064912, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00025947256208999766, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0002606914133753402, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0002621006432259607, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00026428460665435635, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00026797575905864487, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0002701379997368631, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00027204386678782784, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00027311077614802456, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0002738128314965512, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.000276969744374790 12, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0002772950673824528, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00028013409444237755, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00028363943048492947, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.000286296261039648, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00029166053367461937, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.000303476320731687, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0003092626322686897, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00031286721294513774, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0003136598111998935, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.00031507457113994506, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0003165588863221606, random_state=0),
```

```
DecisionTreeRegressor(ccp_alpha=0.0003195106329899099, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0003238591577233688, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0003275264906067321, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0003303038817388572346, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0003306697971842704 3), random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0003308104487656050 4), random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0003481661213537250 4), random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0003663063074026089 6), random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0003706790901807060 6), random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0003868293010789446 5), random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0003917504205232510 5), random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0004190788673534248, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0004224113019586032 3), random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0004258711503542779 4), random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0004262718990903343, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0004316305651570569, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0004338187107791559, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0004369813480648097 4), random_state=0),
DecisionTreeRegressor(ccp_alpha=0.000441128600014143, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0004434522116206836 5), random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0004458810332177666, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0004536503647673785, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0004627260415482732, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0004799969705322048 4), random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0005069957882166418, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0005110981734039255, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0005329527285247265, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0005645939171769893, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0005734533392733015, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0006422782050491873, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0006547344973780258, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0006627374717288379, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0006835819331433598, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0007036970612517696, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0007375260359882107, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0007468980745584039, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0007839610664333065, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0008278535663032395, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0009146809601933989, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0009266630229479251, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0009329636309037461, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.000962120333621102, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.001015293342164764, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0010946882332499323, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0012482537425252595, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0015689725995184653, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0015916817230829852, random_state=0),
```

```
DecisionTreeRegressor(ccp_alpha=0.0017239766904731434, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0017666417542394341, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0019305389528915043, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0024226875289412187, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0027043610030568557, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0033824803783138657, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.004007993055969517, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.0046013362381538345, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.008275335247743588, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.010318988521363062, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.01962755945268886, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.024839938739329742, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.044719461910841876, random_state=0),
DecisionTreeRegressor(ccp_alpha=0.18217589570305892, random_state=0)]
```

[175]:
```
train_score = [reg.score(X_train_new, y_train)for reg in alphaList]
test_score = [reg.score(X_test_new,y_test) for reg in alphaList]
```

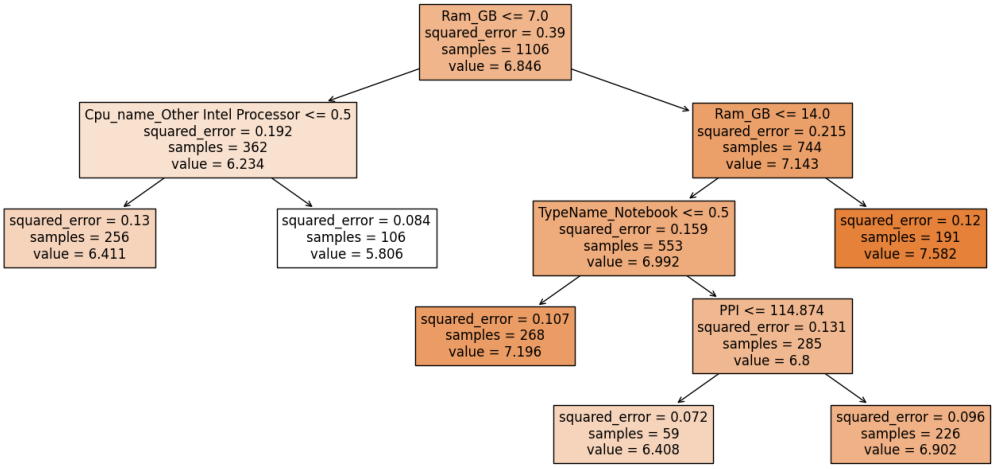
[176]:
```
plt.figure(figsize = (18,8))
plt.title('ccp_alpha vs Accuracy')
plt.plot(ccp_alphas, train_score, marker = 'o',label = 'Training', color =␣
 ↪'red')
plt.plot(ccp_alphas, test_score, marker = '+',label = 'Testing', color =␣
 ↪'blue', drawstyle = 'steps-post')
plt.xlabel('ccp_alpha')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



- Possible values of ccp-alphas can lie between 0.0025 to 0.0075

[177]:
```
reg = DecisionTreeRegressor(ccp_alpha= 0.0085, random_state=0)
reg.fit(X_train_new, y_train)
plt.figure(figsize = (18,8))
tree.plot_tree(reg, filled=True, feature_names=train_new.columns)
```

[177]:
```
[Text(0.5, 0.9, 'Ram_GB <= 7.0\nsquared_error = 0.39\nsamples = 1106\nvalue =
6.846'),
 Text(0.25, 0.7, 'Cpu_name_Other Intel Processor <= 0.5\nsquared_error =
0.192\nsamples = 362\nvalue = 6.234'),
 Text(0.125, 0.5, 'squared_error = 0.13\nsamples = 256\nvalue = 6.411'),
 Text(0.375, 0.5, 'squared_error = 0.084\nsamples = 106\nvalue = 5.806'),
 Text(0.75, 0.7, 'Ram_GB <= 14.0\nsquared_error = 0.215\nsamples = 744\nvalue =
7.143'),
 Text(0.625, 0.5, 'TypeName_Notebook <= 0.5\nsquared_error = 0.159\nsamples =
553\nvalue = 6.992'),
 Text(0.5, 0.3, 'squared_error = 0.107\nsamples = 268\nvalue = 7.196'),
 Text(0.75, 0.3, 'PPI <= 114.874\nsquared_error = 0.131\nsamples = 285\nvalue =
6.8'),
 Text(0.625, 0.1, 'squared_error = 0.072\nsamples = 59\nvalue = 6.408'),
 Text(0.875, 0.1, 'squared_error = 0.096\nsamples = 226\nvalue = 6.902'),
 Text(0.875, 0.5, 'squared_error = 0.12\nsamples = 191\nvalue = 7.582')]
```



[178]:
```
# Hyperparameter tunning

param = {
    'RandomForest':{'model' : RandomForestRegressor(),
                    'param':{
```

```
                        'n_estimators':[int(x) for x in np.
 ↪linspace(100,1200,10)],
                        'criterion':['squared_error', 'absolute_error'],
                        'max_depth':[int(x) for x in np.linspace(1,30,5)],
                        'max_features':['auto','sqrt', 'log2'],
                        'ccp_alpha' : [x for x in np.linspace(0.0025,0.0125,5)],
                        'min_samples_split':[2,5,10,14],
                        'min_samples_leaf':[2,5,10,14]
                    }
                },
    'DecisionTree':{'model':DecisionTreeRegressor(),
                     'param':{'criterion':["squared_error", "absolute_error"],
                        'max_depth':[int(x) for x in np.linspace(1,30,5)],
                        'max_features':['auto','sqrt', 'log2'],
                        'ccp_alpha' : [x for x in np.linspace(0.0025,0.0125,5)],
                        'min_samples_split':[2,5,10,14],
                        'min_samples_leaf':[2,5,10,14]
                         }
                    }
}
```

[179]:
```
param
```

[179]:
```
{'RandomForest': {'model': RandomForestRegressor(),
  'param': {'n_estimators': [100,
    222,
    344,
    466,
    588,
    711,
    833,
    955,
    1077,
    1200],
   'criterion': ['squared_error', 'absolute_error'],
   'max_depth': [1, 8, 15, 22, 30],
   'max_features': ['auto', 'sqrt', 'log2'],
   'ccp_alpha': [0.0025, 0.005, 0.0075, 0.01, 0.0125],
   'min_samples_split': [2, 5, 10, 14],
   'min_samples_leaf': [2, 5, 10, 14]}},
 'DecisionTree': {'model': DecisionTreeRegressor(),
  'param': {'criterion': ['squared_error', 'absolute_error'],
   'max_depth': [1, 8, 15, 22, 30],
   'max_features': ['auto', 'sqrt', 'log2'],
   'ccp_alpha': [0.0025, 0.005, 0.0075, 0.01, 0.0125],
   'min_samples_split': [2, 5, 10, 14],
   'min_samples_leaf': [2, 5, 10, 14]}}}
```

[180]:
```
type(param)
```

[180]:
```
dict
```

[182]:
```
for modelname,mp in param.items():
    print(mp['model'])
    print(mp['param'])
```

```
RandomForestRegressor()
{'n_estimators': [100, 222, 344, 466, 588, 711, 833, 955, 1077, 1200],
'criterion': ['squared_error', 'absolute_error'], 'max_depth': [1, 8, 15, 22,
30], 'max_features': ['auto', 'sqrt', 'log2'], 'ccp_alpha': [0.0025, 0.005,
0.0075, 0.01, 0.0125], 'min_samples_split': [2, 5, 10, 14], 'min_samples_leaf':
[2, 5, 10, 14]}
DecisionTreeRegressor()
{'criterion': ['squared_error', 'absolute_error'], 'max_depth': [1, 8, 15, 22,
30], 'max_features': ['auto', 'sqrt', 'log2'], 'ccp_alpha': [0.0025, 0.005,
0.0075, 0.01, 0.0125], 'min_samples_split': [2, 5, 10, 14], 'min_samples_leaf':
[2, 5, 10, 14]}
```

[183]:
```
score = []
for modelname,mp in param.items():
    clf = RandomizedSearchCV(estimator = mp['model'],
                             param_distributions = mp['param'],
                             cv = 5,
                             n_iter = 10,
                             scoring = 'neg_mean_squared_error',
                             verbose = 2)

    clf.fit(X_train_new,y_train)

    score.append({
        'model_name':modelname,
        'best_score':clf.best_score_,
        'best_estimator':clf.best_estimator_
    })
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END ccp_alpha=0.005, criterion=squared_error, max_depth=15,
max_features=log2, min_samples_leaf=5, min_samples_split=14, n_estimators=466;
total time=   1.1s
[CV] END ccp_alpha=0.005, criterion=squared_error, max_depth=15,
max_features=log2, min_samples_leaf=5, min_samples_split=14, n_estimators=466;
total time=   0.8s
[CV] END ccp_alpha=0.005, criterion=squared_error, max_depth=15,
max_features=log2, min_samples_leaf=5, min_samples_split=14, n_estimators=466;
total time=   0.8s
[CV] END ccp_alpha=0.005, criterion=squared_error, max_depth=15,
```

```
max_features=log2, min_samples_leaf=5, min_samples_split=14, n_estimators=466;
total time=   0.8s
[CV] END ccp_alpha=0.005, criterion=squared_error, max_depth=15,
max_features=log2, min_samples_leaf=5, min_samples_split=14, n_estimators=466;
total time=   0.8s
[CV] END ccp_alpha=0.0075, criterion=absolute_error, max_depth=8,
max_features=log2, min_samples_leaf=2, min_samples_split=5, n_estimators=466;
total time=   3.4s
[CV] END ccp_alpha=0.0075, criterion=absolute_error, max_depth=8,
max_features=log2, min_samples_leaf=2, min_samples_split=5, n_estimators=466;
total time=   3.6s
[CV] END ccp_alpha=0.0075, criterion=absolute_error, max_depth=8,
max_features=log2, min_samples_leaf=2, min_samples_split=5, n_estimators=466;
total time=   4.9s
[CV] END ccp_alpha=0.0075, criterion=absolute_error, max_depth=8,
max_features=log2, min_samples_leaf=2, min_samples_split=5, n_estimators=466;
total time=   3.6s
[CV] END ccp_alpha=0.0075, criterion=absolute_error, max_depth=8,
max_features=log2, min_samples_leaf=2, min_samples_split=5, n_estimators=466;
total time=   3.4s
[CV] END ccp_alpha=0.0075, criterion=squared_error, max_depth=22,
max_features=log2, min_samples_leaf=2, min_samples_split=10, n_estimators=1077;
total time=   1.9s
[CV] END ccp_alpha=0.0075, criterion=squared_error, max_depth=22,
max_features=log2, min_samples_leaf=2, min_samples_split=10, n_estimators=1077;
total time=   2.7s
[CV] END ccp_alpha=0.0075, criterion=squared_error, max_depth=22,
max_features=log2, min_samples_leaf=2, min_samples_split=10, n_estimators=1077;
total time=   2.7s
[CV] END ccp_alpha=0.0075, criterion=squared_error, max_depth=22,
max_features=log2, min_samples_leaf=2, min_samples_split=10, n_estimators=1077;
total time=   1.9s
[CV] END ccp_alpha=0.0075, criterion=squared_error, max_depth=22,
max_features=log2, min_samples_leaf=2, min_samples_split=10, n_estimators=1077;
total time=   1.9s
[CV] END ccp_alpha=0.0125, criterion=absolute_error, max_depth=30,
max_features=log2, min_samples_leaf=10, min_samples_split=14, n_estimators=1077;
total time=   7.5s
[CV] END ccp_alpha=0.0125, criterion=absolute_error, max_depth=30,
max_features=log2, min_samples_leaf=10, min_samples_split=14, n_estimators=1077;
total time=   7.1s
[CV] END ccp_alpha=0.0125, criterion=absolute_error, max_depth=30,
max_features=log2, min_samples_leaf=10, min_samples_split=14, n_estimators=1077;
total time=   6.6s
[CV] END ccp_alpha=0.0125, criterion=absolute_error, max_depth=30,
max_features=log2, min_samples_leaf=10, min_samples_split=14, n_estimators=1077;
total time=   7.5s
[CV] END ccp_alpha=0.0125, criterion=absolute_error, max_depth=30,
```

```
max_features=log2, min_samples_leaf=10, min_samples_split=14, n_estimators=1077;
total time=   6.3s
[CV] END ccp_alpha=0.0125, criterion=squared_error, max_depth=1,
max_features=log2, min_samples_leaf=5, min_samples_split=2, n_estimators=100;
total time=   0.2s
[CV] END ccp_alpha=0.0125, criterion=squared_error, max_depth=1,
max_features=log2, min_samples_leaf=5, min_samples_split=2, n_estimators=100;
total time=   0.2s
[CV] END ccp_alpha=0.0125, criterion=squared_error, max_depth=1,
max_features=log2, min_samples_leaf=5, min_samples_split=2, n_estimators=100;
total time=   0.2s
[CV] END ccp_alpha=0.0125, criterion=squared_error, max_depth=1,
max_features=log2, min_samples_leaf=5, min_samples_split=2, n_estimators=100;
total time=   0.2s
[CV] END ccp_alpha=0.0125, criterion=squared_error, max_depth=1,
max_features=log2, min_samples_leaf=5, min_samples_split=2, n_estimators=100;
total time=   0.2s
[CV] END ccp_alpha=0.0125, criterion=absolute_error, max_depth=1,
max_features=auto, min_samples_leaf=14, min_samples_split=10, n_estimators=711;
total time=   0.0s
[CV] END ccp_alpha=0.0125, criterion=absolute_error, max_depth=1,
max_features=auto, min_samples_leaf=14, min_samples_split=10, n_estimators=711;
total time=   0.0s
[CV] END ccp_alpha=0.0125, criterion=absolute_error, max_depth=1,
max_features=auto, min_samples_leaf=14, min_samples_split=10, n_estimators=711;
total time=   0.0s
[CV] END ccp_alpha=0.0125, criterion=absolute_error, max_depth=1,
max_features=auto, min_samples_leaf=14, min_samples_split=10, n_estimators=711;
total time=   0.0s
[CV] END ccp_alpha=0.0125, criterion=absolute_error, max_depth=1,
max_features=auto, min_samples_leaf=14, min_samples_split=10, n_estimators=711;
total time=   0.0s
[CV] END ccp_alpha=0.005, criterion=squared_error, max_depth=22,
max_features=auto, min_samples_leaf=14, min_samples_split=2, n_estimators=100;
total time=   0.0s
[CV] END ccp_alpha=0.005, criterion=squared_error, max_depth=22,
max_features=auto, min_samples_leaf=14, min_samples_split=2, n_estimators=100;
total time=   0.0s
[CV] END ccp_alpha=0.005, criterion=squared_error, max_depth=22,
max_features=auto, min_samples_leaf=14, min_samples_split=2, n_estimators=100;
total time=   0.0s
[CV] END ccp_alpha=0.005, criterion=squared_error, max_depth=22,
max_features=auto, min_samples_leaf=14, min_samples_split=2, n_estimators=100;
total time=   0.0s
[CV] END ccp_alpha=0.005, criterion=squared_error, max_depth=22,
max_features=auto, min_samples_leaf=14, min_samples_split=2, n_estimators=100;
total time=   0.0s
[CV] END ccp_alpha=0.01, criterion=squared_error, max_depth=30,
```

```
max_features=sqrt, min_samples_leaf=5, min_samples_split=14, n_estimators=466;
total time=   1.3s
[CV] END ccp_alpha=0.01, criterion=squared_error, max_depth=30,
max_features=sqrt, min_samples_leaf=5, min_samples_split=14, n_estimators=466;
total time=   1.3s
[CV] END ccp_alpha=0.01, criterion=squared_error, max_depth=30,
max_features=sqrt, min_samples_leaf=5, min_samples_split=14, n_estimators=466;
total time=   1.0s
[CV] END ccp_alpha=0.01, criterion=squared_error, max_depth=30,
max_features=sqrt, min_samples_leaf=5, min_samples_split=14, n_estimators=466;
total time=   0.8s
[CV] END ccp_alpha=0.01, criterion=squared_error, max_depth=30,
max_features=sqrt, min_samples_leaf=5, min_samples_split=14, n_estimators=466;
total time=   0.8s
[CV] END ccp_alpha=0.01, criterion=absolute_error, max_depth=1,
max_features=auto, min_samples_leaf=2, min_samples_split=5, n_estimators=1077;
total time=   0.0s
[CV] END ccp_alpha=0.01, criterion=absolute_error, max_depth=1,
max_features=auto, min_samples_leaf=2, min_samples_split=5, n_estimators=1077;
total time=   0.0s
[CV] END ccp_alpha=0.01, criterion=absolute_error, max_depth=1,
max_features=auto, min_samples_leaf=2, min_samples_split=5, n_estimators=1077;
total time=   0.0s
[CV] END ccp_alpha=0.01, criterion=absolute_error, max_depth=1,
max_features=auto, min_samples_leaf=2, min_samples_split=5, n_estimators=1077;
total time=   0.0s
[CV] END ccp_alpha=0.01, criterion=absolute_error, max_depth=1,
max_features=auto, min_samples_leaf=2, min_samples_split=5, n_estimators=1077;
total time=   0.0s
[CV] END ccp_alpha=0.0025, criterion=absolute_error, max_depth=15,
max_features=sqrt, min_samples_leaf=10, min_samples_split=2, n_estimators=1077;
total time=   7.5s
[CV] END ccp_alpha=0.0025, criterion=absolute_error, max_depth=15,
max_features=sqrt, min_samples_leaf=10, min_samples_split=2, n_estimators=1077;
total time=   8.4s
[CV] END ccp_alpha=0.0025, criterion=absolute_error, max_depth=15,
max_features=sqrt, min_samples_leaf=10, min_samples_split=2, n_estimators=1077;
total time=   7.1s
[CV] END ccp_alpha=0.0025, criterion=absolute_error, max_depth=15,
max_features=sqrt, min_samples_leaf=10, min_samples_split=2, n_estimators=1077;
total time=   8.4s
[CV] END ccp_alpha=0.0025, criterion=absolute_error, max_depth=15,
max_features=sqrt, min_samples_leaf=10, min_samples_split=2, n_estimators=1077;
total time=   7.6s
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END ccp_alpha=0.01, criterion=squared_error, max_depth=1,
max_features=log2, min_samples_leaf=14, min_samples_split=10; total time=   0.0s
[CV] END ccp_alpha=0.01, criterion=squared_error, max_depth=1,
```

```
max_features=log2, min_samples_leaf=14, min_samples_split=10; total time=   0.0s
[CV] END ccp_alpha=0.01, criterion=squared_error, max_depth=1,
max_features=log2, min_samples_leaf=14, min_samples_split=10; total time=   0.0s
[CV] END ccp_alpha=0.01, criterion=squared_error, max_depth=1,
max_features=log2, min_samples_leaf=14, min_samples_split=10; total time=   0.0s
[CV] END ccp_alpha=0.01, criterion=squared_error, max_depth=1,
max_features=log2, min_samples_leaf=14, min_samples_split=10; total time=   0.0s
[CV] END ccp_alpha=0.0125, criterion=squared_error, max_depth=8,
max_features=log2, min_samples_leaf=14, min_samples_split=10; total time=   0.0s
[CV] END ccp_alpha=0.0125, criterion=squared_error, max_depth=8,
max_features=log2, min_samples_leaf=14, min_samples_split=10; total time=   0.0s
[CV] END ccp_alpha=0.0125, criterion=squared_error, max_depth=8,
max_features=log2, min_samples_leaf=14, min_samples_split=10; total time=   0.0s
[CV] END ccp_alpha=0.0125, criterion=squared_error, max_depth=8,
max_features=log2, min_samples_leaf=14, min_samples_split=10; total time=   0.0s
[CV] END ccp_alpha=0.0125, criterion=squared_error, max_depth=8,
max_features=log2, min_samples_leaf=14, min_samples_split=10; total time=   0.0s
[CV] END ccp_alpha=0.0125, criterion=squared_error, max_depth=8,
max_features=log2, min_samples_leaf=10, min_samples_split=2; total time=   0.0s
[CV] END ccp_alpha=0.0125, criterion=squared_error, max_depth=8,
max_features=log2, min_samples_leaf=10, min_samples_split=2; total time=   0.0s
[CV] END ccp_alpha=0.0125, criterion=squared_error, max_depth=8,
max_features=log2, min_samples_leaf=10, min_samples_split=2; total time=   0.0s
[CV] END ccp_alpha=0.0125, criterion=squared_error, max_depth=8,
max_features=log2, min_samples_leaf=10, min_samples_split=2; total time=   0.0s
[CV] END ccp_alpha=0.0125, criterion=squared_error, max_depth=8,
max_features=log2, min_samples_leaf=10, min_samples_split=2; total time=   0.0s
[CV] END ccp_alpha=0.01, criterion=squared_error, max_depth=1,
max_features=sqrt, min_samples_leaf=10, min_samples_split=2; total time=   0.0s
[CV] END ccp_alpha=0.01, criterion=squared_error, max_depth=1,
max_features=sqrt, min_samples_leaf=10, min_samples_split=2; total time=   0.0s
[CV] END ccp_alpha=0.01, criterion=squared_error, max_depth=1,
max_features=sqrt, min_samples_leaf=10, min_samples_split=2; total time=   0.0s
[CV] END ccp_alpha=0.01, criterion=squared_error, max_depth=1,
max_features=sqrt, min_samples_leaf=10, min_samples_split=2; total time=   0.0s
[CV] END ccp_alpha=0.01, criterion=squared_error, max_depth=1,
max_features=sqrt, min_samples_leaf=10, min_samples_split=2; total time=   0.0s
[CV] END ccp_alpha=0.0025, criterion=absolute_error, max_depth=1,
max_features=auto, min_samples_leaf=2, min_samples_split=14; total time=   0.0s
[CV] END ccp_alpha=0.0025, criterion=absolute_error, max_depth=1,
max_features=auto, min_samples_leaf=2, min_samples_split=14; total time=   0.0s
[CV] END ccp_alpha=0.0025, criterion=absolute_error, max_depth=1,
max_features=auto, min_samples_leaf=2, min_samples_split=14; total time=   0.0s
[CV] END ccp_alpha=0.0025, criterion=absolute_error, max_depth=1,
max_features=auto, min_samples_leaf=2, min_samples_split=14; total time=   0.0s
[CV] END ccp_alpha=0.0025, criterion=absolute_error, max_depth=1,
max_features=auto, min_samples_leaf=2, min_samples_split=14; total time=   0.0s
[CV] END ccp_alpha=0.0125, criterion=squared_error, max_depth=1,
```

```
max_features=auto, min_samples_leaf=2, min_samples_split=2; total time=   0.0s
[CV] END ccp_alpha=0.0125, criterion=squared_error, max_depth=1,
max_features=auto, min_samples_leaf=2, min_samples_split=2; total time=   0.0s
[CV] END ccp_alpha=0.0125, criterion=squared_error, max_depth=1,
max_features=auto, min_samples_leaf=2, min_samples_split=2; total time=   0.0s
[CV] END ccp_alpha=0.0125, criterion=squared_error, max_depth=1,
max_features=auto, min_samples_leaf=2, min_samples_split=2; total time=   0.0s
[CV] END ccp_alpha=0.0125, criterion=squared_error, max_depth=1,
max_features=auto, min_samples_leaf=2, min_samples_split=2; total time=   0.0s
[CV] END ccp_alpha=0.005, criterion=absolute_error, max_depth=15,
max_features=auto, min_samples_leaf=5, min_samples_split=5; total time=   0.0s
[CV] END ccp_alpha=0.005, criterion=absolute_error, max_depth=15,
max_features=auto, min_samples_leaf=5, min_samples_split=5; total time=   0.0s
[CV] END ccp_alpha=0.005, criterion=absolute_error, max_depth=15,
max_features=auto, min_samples_leaf=5, min_samples_split=5; total time=   0.0s
[CV] END ccp_alpha=0.005, criterion=absolute_error, max_depth=15,
max_features=auto, min_samples_leaf=5, min_samples_split=5; total time=   0.0s
[CV] END ccp_alpha=0.005, criterion=absolute_error, max_depth=15,
max_features=auto, min_samples_leaf=5, min_samples_split=5; total time=   0.0s
[CV] END ccp_alpha=0.0025, criterion=squared_error, max_depth=1,
max_features=sqrt, min_samples_leaf=5, min_samples_split=10; total time=   0.0s
[CV] END ccp_alpha=0.0025, criterion=squared_error, max_depth=1,
max_features=sqrt, min_samples_leaf=5, min_samples_split=10; total time=   0.0s
[CV] END ccp_alpha=0.0025, criterion=squared_error, max_depth=1,
max_features=sqrt, min_samples_leaf=5, min_samples_split=10; total time=   0.0s
[CV] END ccp_alpha=0.0025, criterion=squared_error, max_depth=1,
max_features=sqrt, min_samples_leaf=5, min_samples_split=10; total time=   0.0s
[CV] END ccp_alpha=0.0025, criterion=squared_error, max_depth=1,
max_features=sqrt, min_samples_leaf=5, min_samples_split=10; total time=   0.0s
[CV] END ccp_alpha=0.0125, criterion=squared_error, max_depth=15,
max_features=sqrt, min_samples_leaf=5, min_samples_split=14; total time=   0.0s
[CV] END ccp_alpha=0.0125, criterion=squared_error, max_depth=15,
max_features=sqrt, min_samples_leaf=5, min_samples_split=14; total time=   0.0s
[CV] END ccp_alpha=0.0125, criterion=squared_error, max_depth=15,
max_features=sqrt, min_samples_leaf=5, min_samples_split=14; total time=   0.0s
[CV] END ccp_alpha=0.0125, criterion=squared_error, max_depth=15,
max_features=sqrt, min_samples_leaf=5, min_samples_split=14; total time=   0.0s
[CV] END ccp_alpha=0.0125, criterion=squared_error, max_depth=15,
max_features=sqrt, min_samples_leaf=5, min_samples_split=14; total time=   0.0s
[CV] END ccp_alpha=0.0025, criterion=squared_error, max_depth=1,
max_features=sqrt, min_samples_leaf=2, min_samples_split=14; total time=   0.0s
[CV] END ccp_alpha=0.0025, criterion=squared_error, max_depth=1,
max_features=sqrt, min_samples_leaf=2, min_samples_split=14; total time=   0.0s
[CV] END ccp_alpha=0.0025, criterion=squared_error, max_depth=1,
max_features=sqrt, min_samples_leaf=2, min_samples_split=14; total time=   0.0s
[CV] END ccp_alpha=0.0025, criterion=squared_error, max_depth=1,
max_features=sqrt, min_samples_leaf=2, min_samples_split=14; total time=   0.0s
[CV] END ccp_alpha=0.0025, criterion=squared_error, max_depth=1,
```

```
max_features=sqrt, min_samples_leaf=2, min_samples_split=14; total time=   0.0s
```

[184]:
```
score
```

[184]:
```
[{'model_name': 'RandomForest',
  'best_score': -0.08097049654499999,
  'best_estimator': RandomForestRegressor(ccp_alpha=0.0025,
criterion='absolute_error',
                        max_depth=15, max_features='sqrt', min_samples_leaf=10,
                        n_estimators=1077)},
 {'model_name': 'DecisionTree',
  'best_score': -0.1763036032638747,
  'best_estimator': DecisionTreeRegressor(ccp_alpha=0.0125, max_depth=8,
max_features='log2',
                        min_samples_leaf=14, min_samples_split=10)}]
```

[185]:
```
RFReg =RandomForestRegressor(ccp_alpha=0.0025, criterion='absolute_error',␣
 ↪max_depth=15,
                        max_features='log2', min_samples_leaf=2,
                        min_samples_split=14, n_estimators=955)

RFReg.fit(X_train_new,y_train)
```

[185]:
```
RandomForestRegressor(ccp_alpha=0.0025, criterion='absolute_error',
                      max_depth=15, max_features='log2', min_samples_leaf=2,
                      min_samples_split=14, n_estimators=955)
```

[186]:
```
y_pred_RFReg = RFReg.predict(X_test_new)
y_pred_RFReg
```

[186]:
```
array([6.62238826, 5.87748347, 6.21519475, 6.32303501, 7.01262633,
       7.43565693, 6.265146  , 6.44257791, 7.12181773, 5.8802623 ,
       7.03245281, 7.06656823, 7.39155023, 6.91298474, 7.52126364,
       6.95486988, 6.94562322, 5.86046426, 7.44085156, 7.50521054,
       7.2843852 , 7.20394123, 6.62752788, 7.17891932, 6.45147446,
       7.50287194, 7.04420971, 6.16476401, 6.9650465 , 6.5380618 ,
       7.07815414, 6.21845327, 6.83316259, 7.01551763, 7.10508221,
       6.82185225, 6.98792385, 7.09622899, 7.38862792, 7.46330444,
       7.38511163, 7.38043149, 6.00444598, 5.93801224, 6.26410244,
       6.10094767, 7.00093073, 7.37396234, 7.14896144, 6.9632562 ,
       7.63656503, 7.49483743, 5.96549115, 5.80337623, 6.57636881,
       6.84545713, 6.2398696 , 7.17829359, 6.18364462, 6.45214764,
       7.21942675, 7.10374438, 5.81864902, 7.00953203, 6.92733807,
       6.46286105, 6.175197  , 7.20896878, 6.63156294, 5.90055166,
       6.99115388, 6.66858123, 6.80369468, 6.43066776, 7.12731081,
       7.19705507, 7.33592919, 7.69837273, 7.23142625, 6.49235997,
       6.51641999, 7.22190434, 7.23403325, 6.26403854, 7.14213642,
```

```
       6.53319773, 7.09489583, 7.33943667, 6.31301078, 6.79499154,
       6.62786525, 6.39355986, 6.70047279, 7.47822421, 6.41534904,
       5.88772622, 6.43066776, 6.27673003, 5.90740988, 7.36826161,
       6.52136044, 7.23666134, 6.14307636, 7.37398744, 6.67296743,
       7.16234761, 7.22956706, 6.56981094, 7.006902  , 5.92985734,
       6.67566415, 6.93326619, 6.64489416, 7.50858253, 6.22211336,
       6.62238826, 5.83999662, 7.59962092, 6.49065958, 6.71465251,
       6.55424455, 7.42891325, 6.94277083, 6.48912638, 6.60342705,
       7.23242506, 6.44524266, 5.93158576, 7.47827439, 7.38406622,
       6.57105867, 7.11907095, 6.28385587, 7.28658886, 7.22682281,
       6.51597451, 7.00849992, 7.16234761, 7.28257752, 7.50858253,
       5.86813833, 5.78814419, 6.83349787, 6.43219589, 6.46106144,
       7.40397476, 6.53197777, 5.8694885 , 7.24014738, 6.3359045 ,
       7.62884489, 7.28098503, 6.53229439, 7.24790733, 6.91057316,
       6.46286105, 7.28849672, 7.51775283, 7.19793781, 7.45887314,
       5.76152517, 6.36734205, 6.65448079, 6.51575714, 6.94481255,
       7.06962237, 7.4532459 , 6.33117441, 7.41854095, 7.03692304,
       6.31474486, 7.02575495, 6.07230422, 7.06391261, 6.82145355,
       7.24686471, 7.52371946, 7.50121258, 7.02681637, 6.42215278,
       6.88288305, 7.16388485, 7.24236038, 7.0600126 , 6.21924028,
       6.84790146, 6.55731323, 7.20019874, 6.47177361, 6.10373967,
       6.31182559, 7.11783105, 6.12755753, 7.01358898, 7.34633943,
       7.42563054])
```

[187]:
```
print('R2_sore: ', metrics.r2_score(y_test,y_pred_RFReg))
print('MAE: ', metrics.mean_absolute_error(y_test,y_pred_RFReg))
print('MSE: ',metrics.mean_squared_error(y_test,y_pred_RFReg))

# Calculate R-squared scores
train_r2 = metrics.r2_score(y_train, RFReg.predict(X_train_new))
test_r2 = metrics.r2_score(y_test, y_pred_RFReg)

# Print the results
print(f'Training R-squared: {train_r2}')
print(f'Test R-squared: {test_r2}')
```

```
R2_sore:  0.843576670828858
MAE:  0.19346204418824947
MSE:  0.05983991972527323
Training R-squared: 0.8441895055225832
Test R-squared: 0.843576670828858
```

[188]:
```
RFRegV2 =RandomForestRegressor(ccp_alpha=0.0025, criterion='absolute_error',␣
 ↪max_depth=30,
                        max_features='log2', min_samples_leaf=5,
                        min_samples_split=10, n_estimators=466)
```

```
RFRegV2.fit(X_train_new,y_train)
```

[188]:
```
RandomForestRegressor(ccp_alpha=0.0025, criterion='absolute_error',
                      max_depth=30, max_features='log2', min_samples_leaf=5,
                      min_samples_split=10, n_estimators=466)
```

[189]:
```
y_pred_RFRegV2 = RFRegV2.predict(X_test_new)
y_pred_RFRegV2
```

[189]:
```
array([6.60905746, 5.89585776, 6.21757104, 6.32763394, 6.99737209,
       7.43694958, 6.25055572, 6.43400286, 7.15066845, 5.89843323,
       7.01400336, 7.08023648, 7.37867288, 6.92975573, 7.50941379,
       6.95032984, 6.94778378, 5.88536615, 7.43419874, 7.48776344,
       7.27859389, 7.17984159, 6.63736616, 7.19404401, 6.46391145,
       7.48751566, 7.03937502, 6.18313352, 6.95738436, 6.55329953,
       7.07962002, 6.22879809, 6.84249174, 7.02570877, 7.12508318,
       6.81727015, 6.9930152 , 7.08164908, 7.38860945, 7.45402568,
       7.37200422, 7.38380397, 6.00274983, 5.956635  , 6.2468907 ,
       6.11824064, 7.01330987, 7.37989347, 7.12627572, 6.96041176,
       7.615476  , 7.49750525, 6.01611472, 5.83401773, 6.60634872,
       6.83391565, 6.23340698, 7.16879669, 6.17900936, 6.46359894,
       7.22471911, 7.12508318, 5.82899203, 7.01738841, 6.9290928 ,
       6.45338262, 6.1836427 , 7.21836178, 6.60604498, 5.92422321,
       6.98664266, 6.65884594, 6.81580346, 6.42418716, 7.14442194,
       7.20972615, 7.3249972 , 7.65978529, 7.22139491, 6.49114359,
       6.50520053, 7.21156991, 7.22768161, 6.26477835, 7.10231207,
       6.52428528, 7.09995413, 7.3332059 , 6.31816397, 6.81260725,
       6.63389783, 6.40216878, 6.71843566, 7.46648549, 6.41973559,
       5.90138065, 6.42418716, 6.2635238 , 5.93006275, 7.32431197,
       6.51770903, 7.24221446, 6.15422291, 7.36037444, 6.68079604,
       7.18746104, 7.22490074, 6.51868717, 6.98561962, 5.95971435,
       6.67528753, 6.90168601, 6.63686453, 7.50249702, 6.22196427,
       6.60905746, 5.88368987, 7.59456562, 6.48265675, 6.73254483,
       6.54223781, 7.40099244, 6.94605649, 6.4715694 , 6.60682915,
       7.26608311, 6.44354019, 5.93489615, 7.46371998, 7.38490351,
       6.56621308, 7.08990031, 6.28580947, 7.28959388, 7.24037962,
       6.51669351, 7.00689652, 7.18746104, 7.28446507, 7.50249702,
       5.88362169, 5.81517546, 6.83383122, 6.41051092, 6.45189008,
       7.39967483, 6.5332914 , 5.90787583, 7.23339796, 6.32886557,
       7.60625924, 7.29279243, 6.52567264, 7.24659289, 6.90487763,
       6.45489372, 7.29911369, 7.50914616, 7.21775521, 7.46390314,
       5.80143288, 6.37189614, 6.65983792, 6.51467653, 6.9421189 ,
       7.08657341, 7.43927067, 6.31912011, 7.41598517, 7.03377001,
       6.31839689, 7.02325352, 6.12700653, 7.0676174 , 6.82308395,
       7.25673219, 7.50594525, 7.49261415, 6.97853998, 6.43469953,
       6.86821535, 7.18108036, 7.25280596, 7.07060093, 6.220213  ,
       6.82764797, 6.55526891, 7.17595581, 6.46789042, 6.15925247,
```

```
       6.32407398, 7.14651456, 6.13051803, 7.01590818, 7.34412336,
       7.42192892])
```

[190]:
```
print('R2_sore: ', metrics.r2_score(y_test,y_pred_RFRegV2))
print('MAE: ', metrics.mean_absolute_error(y_test,y_pred_RFRegV2))
print('MSE: ',metrics.mean_squared_error(y_test,y_pred_RFRegV2))

# Calculate R-squared scores
train_r2 = metrics.r2_score(y_train, RFRegV2.predict(X_train_new))
test_r2 = metrics.r2_score(y_test, y_pred_RFRegV2)

# Print the results
print(f'Training R-squared: {train_r2}')
print(f'Test R-squared: {test_r2}')
```

```
R2_sore:  0.83572357883684
MAE:  0.19721247060802555
MSE:  0.06284412886011013
Training R-squared: 0.8312606138533705
Test R-squared: 0.83572357883684
```

[191]:
```
train_new
```

[191]:

	Ram_GB	Weight_KG	TouchScreen	IPS	PPI	HDD	SSD	\
0	8	1.37	0	1	226.992481	0	128	
1	8	1.34	0	0	127.669173	0	0	
2	8	1.86	0	0	141.217949	0	256	
3	16	1.83	0	1	220.519481	0	512	
4	8	1.37	0	1	226.992481	0	256	
…	…	…	…	…	…	…	…	
1298	4	1.80	1	1	157.357143	0	128	
1299	16	1.30	1	1	276.090226	0	512	
1300	2	1.50	0	0	111.928571	0	0	
1301	6	2.19	0	0	100.448718	1000	0	
1302	4	2.20	0	0	100.448718	500	0	

	Company_Apple	Company_Asus	Company_Chuwi	…	TypeName_Ultrabook	\
0	1	0	0	…	1	
1	1	0	0	…	1	
2	0	0	0	…	0	
3	1	0	0	…	1	
4	1	0	0	…	1	
…	…	…	…	…	…	
1298	0	0	0	…	0	
1299	0	0	0	…	0	
1300	0	0	0	…	0	
1301	0	0	0	…	0	

```
1302              0             1              0  …                    0

      TypeName_Workstation  OpSys_Others  OpSys_Windows  \
0                        0             0              0
1                        0             0              0
2                        0             1              0
3                        0             0              0
4                        0             0              0
…                      …             …              …
1298                     0             0              1
1299                     0             0              1
1300                     0             0              1
1301                     0             0              1
1302                     0             0              1

      Cpu_name_Intel Core i3  Cpu_name_Intel Core i5  Cpu_name_Intel Core i7  \
0                          0                       1                       0
1                          0                       1                       0
2                          0                       1                       0
3                          0                       0                       1
4                          0                       1                       0
…                        …                       …                     …
1298                       0                       0                       1
1299                       0                       0                       1
1300                       0                       0                       0
1301                       0                       0                       1
1302                       0                       0                       0

      Cpu_name_Other Intel Processor  Gpu_brand_Intel  Gpu_brand_Nvidia
0                                  0                1                 0
1                                  0                1                 0
2                                  0                1                 0
3                                  0                0                 0
4                                  0                1                 0
…                                …                …               …
1298                               0                1                 0
1299                               0                1                 0
1300                               1                1                 0
1301                               0                0                 0
1302                               1                1                 0

[1302 rows x 38 columns]
```

[192]:
```
X_train_new.columns
```

[192]:
```
Index(['Ram_GB', 'Weight_KG', 'TouchScreen', 'IPS', 'PPI', 'HDD', 'SSD',
       'Company_Apple', 'Company_Asus', 'Company_Chuwi', 'Company_Dell',
```

```
       'Company_Fujitsu', 'Company_Google', 'Company_HP', 'Company_Huawei',
       'Company_LG', 'Company_Lenovo', 'Company_MSI', 'Company_Mediacom',
       'Company_Microsoft', 'Company_Razer', 'Company_Samsung',
       'Company_Toshiba', 'Company_Vero', 'Company_Xiaomi', 'TypeName_Gaming',
       'TypeName_Netbook', 'TypeName_Notebook', 'TypeName_Ultrabook',
       'TypeName_Workstation', 'OpSys_Others', 'OpSys_Windows',
       'Cpu_name_Intel Core i3', 'Cpu_name_Intel Core i5',
       'Cpu_name_Intel Core i7', 'Cpu_name_Other Intel Processor',
       'Gpu_brand_Intel', 'Gpu_brand_Nvidia'],
      dtype='object')
```

[193]:
```
train_new.columns
```

[193]:
```
Index(['Ram_GB', 'Weight_KG', 'TouchScreen', 'IPS', 'PPI', 'HDD', 'SSD',
       'Company_Apple', 'Company_Asus', 'Company_Chuwi', 'Company_Dell',
       'Company_Fujitsu', 'Company_Google', 'Company_HP', 'Company_Huawei',
       'Company_LG', 'Company_Lenovo', 'Company_MSI', 'Company_Mediacom',
       'Company_Microsoft', 'Company_Razer', 'Company_Samsung',
       'Company_Toshiba', 'Company_Vero', 'Company_Xiaomi', 'TypeName_Gaming',
       'TypeName_Netbook', 'TypeName_Notebook', 'TypeName_Ultrabook',
       'TypeName_Workstation', 'OpSys_Others', 'OpSys_Windows',
       'Cpu_name_Intel Core i3', 'Cpu_name_Intel Core i5',
       'Cpu_name_Intel Core i7', 'Cpu_name_Other Intel Processor',
       'Gpu_brand_Intel', 'Gpu_brand_Nvidia'],
      dtype='object')
```

[194]:
```
# Prediction on whole dataset

predicted = []

npArrar_df = np.array(train_new)
for i in range(len(npArrar_df)):
    predicted.append(RFRegV2.predict([npArrar_df[i]]))

predicted
```

[194]:
```
[array([7.18041068]),
 array([6.98561962]),
 array([6.70418938]),
 array([7.44129676]),
 array([7.23339796]),
 array([6.20205659]),
 array([7.20816778]),
 array([6.98561962]),
 array([7.46004034]),
 array([7.0986606]),
 array([6.26980068]),
```

```
array([6.18201003]),
array([7.38937499]),
array([6.38073428]),
array([7.10554448]),
array([7.23339796]),
array([6.94977508]),
array([7.44129676]),
array([6.40776766]),
array([7.15355587]),
array([5.82899203]),
array([6.98513681]),
array([6.1346071]),
array([7.08023648]),
array([7.22100551]),
array([6.20847476]),
array([6.98561962]),
array([6.83815078]),
array([7.2798448]),
array([6.71618469]),
array([5.92398442]),
array([6.16366146]),
array([6.3427927]),
array([7.49516663]),
array([6.98561962]),
array([5.84027749]),
array([6.10674346]),
array([6.82308395]),
array([6.43132904]),
array([6.8610859]),
array([6.16474044]),
array([7.41115269]),
array([6.44571222]),
array([6.65400677]),
array([6.8927744]),
array([7.23339796]),
array([6.1836427]),
array([7.14442194]),
array([6.60808531]),
array([6.05879167]),
array([6.15925247]),
array([6.99017398]),
array([6.43412708]),
array([7.11382692]),
array([6.19431737]),
array([6.95083281]),
array([6.34201864]),
array([6.53609979]),
```

```
array([7.43927067]),
array([6.27505417]),
array([7.09335921]),
array([7.09995413]),
array([6.53930892]),
array([7.04281337]),
array([6.90549535]),
array([7.42993017]),
array([6.220213]),
array([5.88362169]),
array([6.51669351]),
array([7.03946912]),
array([6.97853998]),
array([7.29911369]),
array([6.94977508]),
array([7.03542395]),
array([6.32818886]),
array([7.02979907]),
array([6.50683258]),
array([6.90627048]),
array([6.30090852]),
array([6.81619603]),
array([7.06127473]),
array([7.28075763]),
array([6.96382954]),
array([6.37189614]),
array([6.20495651]),
array([7.32517188]),
array([7.21352027]),
array([6.89916128]),
array([7.32810026]),
array([7.32598041]),
array([6.83213353]),
array([6.80058137]),
array([7.19676587]),
array([7.02175224]),
array([7.32351079]),
array([7.10014514]),
array([6.6860266]),
array([6.29817959]),
array([6.40018771]),
array([7.21775521]),
array([6.97297688]),
array([6.32407398]),
array([6.54223781]),
array([7.29735776]),
array([7.2810356]),
```

```
array([6.72115576]),
array([6.25055572]),
array([7.28866609]),
array([7.42678297]),
array([7.04415427]),
array([7.36896489]),
array([7.38257857]),
array([7.22399243]),
array([6.56094969]),
array([7.22529038]),
array([7.22310407]),
array([7.0947209]),
array([6.98052951]),
array([6.30232558]),
array([6.99162994]),
array([6.31077347]),
array([7.42426641]),
array([6.69121578]),
array([7.21403044]),
array([6.31531806]),
array([5.89791976]),
array([6.71513823]),
array([6.22879809]),
array([7.09335921]),
array([6.87792818]),
array([6.50697245]),
array([6.27719851]),
array([7.25880997]),
array([6.39577358]),
array([6.60334798]),
array([7.03618988]),
array([5.89619061]),
array([7.08657341]),
array([6.36387673]),
array([6.86197795]),
array([6.37199906]),
array([6.9290928]),
array([7.05339031]),
array([6.46372713]),
array([6.33567708]),
array([7.32351079]),
array([7.17984159]),
array([5.98410393]),
array([7.51472737]),
array([6.18980248]),
array([7.43298273]),
array([7.03840297]),
```

```
array([6.51783357]),
array([7.49476768]),
array([7.0167774]),
array([6.63636143]),
array([6.62556143]),
array([7.21156991]),
array([7.20459139]),
array([6.22889693]),
array([6.19163527]),
array([6.68153369]),
array([6.84006166]),
array([7.36929407]),
array([5.93128846]),
array([7.41802735]),
array([5.93128846]),
array([7.12851072]),
array([6.92460065]),
array([6.60905746]),
array([7.25890659]),
array([6.53771381]),
array([6.13051803]),
array([6.234167]),
array([6.86070437]),
array([6.30594867]),
array([6.26688266]),
array([7.68365978]),
array([6.78838065]),
array([7.26973561]),
array([6.93184044]),
array([7.22490074]),
array([7.48751566]),
array([6.69451911]),
array([6.73477151]),
array([7.27949516]),
array([7.39320405]),
array([6.89541365]),
array([7.22471911]),
array([6.93036975]),
array([7.51334883]),
array([5.92490782]),
array([7.06388423]),
array([6.59279008]),
array([6.96141713]),
array([6.74004449]),
array([7.72654101]),
array([7.03116372]),
array([7.520514]),
```

```
array([6.98664266]),
array([7.54398524]),
array([6.49273532]),
array([6.80058137]),
array([7.49784028]),
array([7.20456444]),
array([7.2430624]),
array([6.23595736]),
array([7.38031606]),
array([7.33092082]),
array([6.68906113]),
array([6.71843566]),
array([7.49376069]),
array([6.18313352]),
array([6.66695614]),
array([7.41136094]),
array([7.29911369]),
array([6.468076]),
array([7.02579264]),
array([7.10189974]),
array([7.40044879]),
array([6.66792597]),
array([7.04998497]),
array([7.33796387]),
array([6.66849101]),
array([7.67869183]),
array([6.97153399]),
array([6.90168601]),
array([6.86308898]),
array([6.84812652]),
array([6.79626666]),
array([6.17900936]),
array([6.21608894]),
array([7.46390314]),
array([6.95785435]),
array([7.07273427]),
array([6.95032984]),
array([6.10674346]),
array([6.3945048]),
array([7.71669841]),
array([7.01330987]),
array([6.40216878]),
array([6.93737633]),
array([6.82404186]),
array([7.01653192]),
array([7.32249058]),
array([6.90456686]),
```

```
array([6.52389805]),
array([7.59291165]),
array([6.60905746]),
array([7.27859389]),
array([6.93314963]),
array([7.63453179]),
array([6.21612779]),
array([7.38780446]),
array([6.46437969]),
array([7.20224048]),
array([6.50737266]),
array([7.00531981]),
array([7.66227311]),
array([7.29279243]),
array([6.93036975]),
array([6.26841946]),
array([6.55769417]),
array([6.51467653]),
array([6.6619579]),
array([7.02325352]),
array([7.37768305]),
array([6.32763394]),
array([6.82878211]),
array([6.98328277]),
array([7.27859389]),
array([7.28809399]),
array([6.81580346]),
array([7.49261415]),
array([6.67065727]),
array([6.62221828]),
array([6.81053016]),
array([6.82244726]),
array([6.21892913]),
array([6.73254483]),
array([6.60604498]),
array([6.20664202]),
array([7.00976031]),
array([6.63686453]),
array([7.0676174]),
array([6.82764797]),
array([7.00546102]),
array([7.03937502]),
array([7.37642855]),
array([6.69759029]),
array([5.85324653]),
array([7.13577339]),
array([6.32992159]),
```

```
array([6.9930152]),
array([6.88432296]),
array([7.12731506]),
array([6.84249174]),
array([7.6122287]),
array([6.32575893]),
array([7.42993017]),
array([6.35117072]),
array([7.49376069]),
array([6.45096798]),
array([6.4348109]),
array([7.13742484]),
array([5.90153438]),
array([7.40695144]),
array([7.01423016]),
array([6.79644895]),
array([6.26034109]),
array([7.07865771]),
array([7.50790567]),
array([7.46950548]),
array([6.72569341]),
array([5.94581597]),
array([6.77898819]),
array([7.49526839]),
array([5.88571835]),
array([6.94605649]),
array([5.86760145]),
array([7.0973766]),
array([6.43942691]),
array([7.01590818]),
array([6.63389783]),
array([6.54355993]),
array([5.84027749]),
array([6.41947426]),
array([7.21726761]),
array([7.3479432]),
array([7.43018377]),
array([7.26608311]),
array([6.9433016]),
array([6.98985269]),
array([6.25055572]),
array([7.20032765]),
array([7.03377001]),
array([6.99280009]),
array([6.92330298]),
array([5.89998823]),
array([6.87746626]),
```

```
array([5.91986575]),
array([6.56763457]),
array([6.52571764]),
array([7.48751566]),
array([7.33092082]),
array([7.36562827]),
array([5.81517546]),
array([7.40099244]),
array([6.0478833]),
array([6.89230384]),
array([7.39320405]),
array([7.00689652]),
array([7.18746104]),
array([7.37867288]),
array([7.24297288]),
array([6.91138767]),
array([5.90153438]),
array([6.90177037]),
array([6.51826734]),
array([7.07190282]),
array([6.21303014]),
array([6.92636247]),
array([6.51593358]),
array([6.68432526]),
array([6.41745808]),
array([6.19022345]),
array([6.79532151]),
array([6.87664875]),
array([6.94977508]),
array([7.32656107]),
array([7.28959388]),
array([6.27485368]),
array([7.3146007]),
array([6.29825463]),
array([7.23165921]),
array([6.43469953]),
array([6.19519108]),
array([5.9614582]),
array([5.85119025]),
array([6.71843566]),
array([7.29070232]),
array([6.2402527]),
array([7.13856465]),
array([6.94342354]),
array([7.46258075]),
array([7.42679734]),
array([6.41879947]),
```

```
array([6.93005871]),
array([7.36037444]),
array([7.49494899]),
array([7.13577339]),
array([7.50249702]),
array([6.57766946]),
array([7.25663955]),
array([6.81958028]),
array([6.75558692]),
array([6.97437309]),
array([6.95439896]),
array([7.44404589]),
array([7.50914616]),
array([5.93006275]),
array([7.49376069]),
array([6.83391565]),
array([6.84249174]),
array([7.49476768]),
array([7.38380397]),
array([7.46885532]),
array([7.38490351]),
array([6.24885253]),
array([5.83801333]),
array([7.42884299]),
array([7.38552624]),
array([6.26669865]),
array([7.22797019]),
array([7.20868206]),
array([6.45329446]),
array([6.37967948]),
array([7.32611852]),
array([6.50384758]),
array([7.25280596]),
array([7.52357811]),
array([6.01594156]),
array([6.98046664]),
array([6.42086895]),
array([7.68765842]),
array([7.13970729]),
array([6.95173732]),
array([7.10734473]),
array([7.44023042]),
array([6.09017119]),
array([5.8398953]),
array([7.1075149]),
array([6.40894252]),
array([7.58519132]),
```

```
array([7.22399765]),
array([7.14442194]),
array([5.95076218]),
array([7.33765182]),
array([7.51017599]),
array([6.46391145]),
array([6.39546112]),
array([6.60682915]),
array([7.2608686]),
array([6.10996651]),
array([7.14124169]),
array([6.93399322]),
array([6.26525211]),
array([6.73670575]),
array([7.03025877]),
array([7.49353335]),
array([6.66192705]),
array([7.4668496]),
array([7.37989347]),
array([6.96041176]),
array([6.90541264]),
array([6.12700653]),
array([6.99478303]),
array([6.51868717]),
array([7.37789203]),
array([6.47118512]),
array([6.30198707]),
array([6.11540466]),
array([6.6526154]),
array([6.94610293]),
array([6.99696176]),
array([5.91791072]),
array([6.25509469]),
array([6.67740685]),
array([7.31181189]),
array([7.40334549]),
array([7.42192892]),
array([6.71120172]),
array([7.05814749]),
array([7.22991102]),
array([7.04863408]),
array([6.48022314]),
array([7.36241316]),
array([6.94778378]),
array([6.66237389]),
array([7.10283513]),
array([7.08857515]),
```

```
array([6.20833382]),
array([6.73283245]),
array([5.92398442]),
array([6.26477835]),
array([6.96061326]),
array([7.50594525]),
array([7.23277528]),
array([7.47912153]),
array([7.47102762]),
array([6.87513894]),
array([7.50541596]),
array([7.13880064]),
array([6.37619694]),
array([7.60160661]),
array([7.14766081]),
array([6.97556155]),
array([7.15113096]),
array([6.71081354]),
array([6.89139636]),
array([5.84583958]),
array([6.68483295]),
array([7.20003698]),
array([5.93570365]),
array([5.86559764]),
array([7.02922571]),
array([6.96169369]),
array([6.5300073]),
array([7.38776648]),
array([7.05951307]),
array([7.15610715]),
array([6.60219496]),
array([6.65884594]),
array([7.06917459]),
array([7.00434951]),
array([5.80472553]),
array([6.43722666]),
array([7.46704693]),
array([6.87174267]),
array([7.50249702]),
array([7.28792682]),
array([7.32670908]),
array([6.95738436]),
array([6.325609]),
array([6.68219278]),
array([6.93825175]),
array([6.59511677]),
array([6.31900069]),
```

```
array([6.51727848]),
array([6.90487763]),
array([7.60625924]),
array([6.77503661]),
array([7.08990031]),
array([5.93301987]),
array([6.99949319]),
array([6.99596517]),
array([6.83815078]),
array([6.23455241]),
array([7.13714297]),
array([7.02809233]),
array([7.38257857]),
array([6.41681786]),
array([7.35995008]),
array([6.7900644]),
array([6.19240168]),
array([6.34072678]),
array([6.57109859]),
array([6.32763394]),
array([6.2703247]),
array([7.07962002]),
array([6.31912011]),
array([6.90487763]),
array([7.67869183]),
array([6.33080605]),
array([6.83322009]),
array([5.89585776]),
array([5.91794803]),
array([6.60604498]),
array([6.44482546]),
array([6.22388628]),
array([6.08866114]),
array([6.32001706]),
array([7.21475004]),
array([7.14272132]),
array([6.80472051]),
array([7.19282147]),
array([6.46388327]),
array([6.49719536]),
array([5.9867695]),
array([6.49603856]),
array([6.92236043]),
array([6.21936981]),
array([6.48281861]),
array([6.87162938]),
array([6.5332914]),
```

```
array([5.88368987]),
array([7.46930158]),
array([7.65978529]),
array([7.66227311]),
array([6.81549985]),
array([5.94223886]),
array([6.55329953]),
array([6.42727916]),
array([7.16283756]),
array([5.87549965]),
array([7.58468622]),
array([7.58679204]),
array([7.37258442]),
array([6.63405547]),
array([7.04429821]),
array([6.65983792]),
array([6.83842707]),
array([6.2635238]),
array([7.09105282]),
array([6.19637806]),
array([6.30103073]),
array([7.38860945]),
array([6.96142893]),
array([7.26653767]),
array([7.36350922]),
array([7.00689652]),
array([6.21098586]),
array([5.93381863]),
array([7.572723]),
array([6.15314436]),
array([7.50183236]),
array([7.01432705]),
array([6.49801912]),
array([7.19531973]),
array([5.87599623]),
array([7.01014885]),
array([7.18772695]),
array([6.1253182]),
array([6.12639675]),
array([6.29504073]),
array([7.24037962]),
array([6.42486017]),
array([7.29123915]),
array([7.49750525]),
array([6.01611472]),
array([6.78224249]),
array([6.96041176]),
```

```
array([6.49998244]),
array([6.68771539]),
array([6.45489372]),
array([6.79468105]),
array([5.83056681]),
array([6.40808885]),
array([6.65814882]),
array([6.9930152]),
array([7.46059746]),
array([7.40802497]),
array([6.69020269]),
array([6.29978198]),
array([6.17444574]),
array([6.69941829]),
array([7.28832959]),
array([5.79038902]),
array([7.33132729]),
array([7.4122291]),
array([7.12508318]),
array([6.41494383]),
array([7.46808106]),
array([6.94658797]),
array([6.19737698]),
array([5.84169635]),
array([7.09592637]),
array([7.46521824]),
array([7.10628947]),
array([7.35044916]),
array([7.69122388]),
array([6.7900644]),
array([7.22490074]),
array([7.15066845]),
array([6.11824064]),
array([6.77503661]),
array([6.94977508]),
array([7.06917459]),
array([7.24221446]),
array([7.75424191]),
array([6.96109216]),
array([5.92237464]),
array([6.51770903]),
array([6.27759669]),
array([7.18746104]),
array([6.31839689]),
array([6.66842415]),
array([6.78369442]),
array([6.9433016]),
```

```
array([6.18471356]),
array([7.37789203]),
array([6.78838065]),
array([7.22170252]),
array([6.27167826]),
array([6.58397717]),
array([7.4256457]),
array([7.20957429]),
array([6.42703002]),
array([7.35404529]),
array([7.14084692]),
array([7.03025877]),
array([6.03424201]),
array([7.25909987]),
array([6.42035034]),
array([7.38846877]),
array([7.60037438]),
array([6.4254982]),
array([6.22388628]),
array([6.39822975]),
array([5.92175756]),
array([5.84894936]),
array([6.75843377]),
array([7.25541818]),
array([7.37405308]),
array([7.00194923]),
array([5.87504759]),
array([7.45162121]),
array([6.08758028]),
array([7.47172347]),
array([6.59688742]),
array([6.90010757]),
array([6.41160316]),
array([6.36821311]),
array([6.49552335]),
array([6.11002345]),
array([6.86821535]),
array([7.07273427]),
array([7.50941379]),
array([6.95959609]),
array([6.40052919]),
array([7.12508318]),
array([6.78972802]),
array([6.53916007]),
array([5.92367135]),
array([7.02907023]),
array([6.92975573]),
```

```
array([6.90487763]),
array([6.19391389]),
array([5.93850416]),
array([6.99497839]),
array([7.42440925]),
array([7.01561728]),
array([7.22378884]),
array([7.75803248]),
array([7.18746104]),
array([6.25471253]),
array([6.77856016]),
array([6.46359894]),
array([6.425283]),
array([6.31816397]),
array([7.57970545]),
array([6.50652061]),
array([6.41973559]),
array([6.43400286]),
array([7.0258515]),
array([6.57003572]),
array([7.47437279]),
array([6.84774551]),
array([7.43419874]),
array([6.19737698]),
array([6.9421189]),
array([6.45415514]),
array([7.1338083]),
array([7.07309431]),
array([7.59993931]),
array([5.83401773]),
array([7.4245867]),
array([6.59813535]),
array([6.81460068]),
array([7.15145839]),
array([6.16143428]),
array([6.92330298]),
array([6.3020305]),
array([6.92282089]),
array([6.57145988]),
array([7.26933244]),
array([6.82798411]),
array([7.37867288]),
array([7.69314906]),
array([6.54736355]),
array([7.12508318]),
array([7.43694958]),
array([7.12851804]),
```

```
array([7.25673219]),
array([6.31463755]),
array([6.65286782]),
array([6.05879167]),
array([7.36919238]),
array([7.35233944]),
array([5.89843323]),
array([7.09352365]),
array([7.36252893]),
array([6.67783592]),
array([7.03176694]),
array([7.00446455]),
array([7.01830938]),
array([7.32431197]),
array([6.40543483]),
array([7.51210433]),
array([6.48565237]),
array([7.75278764]),
array([7.63800399]),
array([7.45961145]),
array([6.13993482]),
array([7.4382188]),
array([7.45047611]),
array([7.01059355]),
array([6.81260725]),
array([7.64700937]),
array([7.01922504]),
array([7.13627479]),
array([5.8854403]),
array([7.38860945]),
array([7.00711368]),
array([6.90845022]),
array([7.25478304]),
array([5.874513]),
array([6.67147691]),
array([7.32249058]),
array([6.58552446]),
array([7.44715489]),
array([6.86307373]),
array([7.22490074]),
array([7.05093828]),
array([6.17781047]),
array([7.07879646]),
array([6.46372713]),
array([6.38737164]),
array([7.52969767]),
array([6.50536038]),
```

```
array([7.26465794]),
array([7.5386453]),
array([7.22139491]),
array([6.81756738]),
array([6.83815078]),
array([7.34107539]),
array([7.50949286]),
array([5.85378141]),
array([7.60625924]),
array([6.53922566]),
array([6.18903201]),
array([6.89181006]),
array([7.39898427]),
array([7.03069777]),
array([7.01400336]),
array([6.12978883]),
array([7.21156991]),
array([6.28191615]),
array([5.891836]),
array([6.61590593]),
array([7.71322125]),
array([7.49621727]),
array([6.36720658]),
array([6.96041176]),
array([7.07762307]),
array([7.54350514]),
array([7.49376069]),
array([6.234167]),
array([5.86024192]),
array([6.6805306]),
array([7.48776344]),
array([7.74324603]),
array([6.57326265]),
array([6.64893155]),
array([7.35912189]),
array([6.69532196]),
array([6.03653858]),
array([7.19404401]),
array([7.23265754]),
array([7.35912189]),
array([6.00683627]),
array([7.3332059]),
array([6.71884086]),
array([7.24659289]),
array([6.98455631]),
array([6.85614246]),
array([5.92422321]),
```

```
array([7.33549841]),
array([6.72430518]),
array([6.16423892]),
array([6.69230008]),
array([6.88531284]),
array([6.84302641]),
array([6.80027849]),
array([7.49516663]),
array([6.55042791]),
array([7.13176349]),
array([7.33132729]),
array([6.9651036]),
array([7.12627572]),
array([6.41901386]),
array([6.19121419]),
array([7.02450121]),
array([6.4715694]),
array([7.41598517]),
array([7.29332623]),
array([5.89998823]),
array([7.18108036]),
array([6.60895273]),
array([6.68079604]),
array([6.99737209]),
array([6.56959915]),
array([5.87916531]),
array([6.81727015]),
array([5.95971435]),
array([6.99737209]),
array([6.15314436]),
array([7.04835483]),
array([5.88063312]),
array([6.23340698]),
array([6.69859462]),
array([7.44274183]),
array([7.32815493]),
array([7.22378884]),
array([7.53560088]),
array([7.10687472]),
array([6.93616488]),
array([6.81881912]),
array([6.22388628]),
array([6.71485019]),
array([7.41451426]),
array([7.66820465]),
array([7.40916128]),
array([7.10569869]),
```

```
array([6.98161208]),
array([7.43286205]),
array([7.3333715]),
array([5.97830577]),
array([6.8215021]),
array([7.40090753]),
array([6.95785435]),
array([7.10231207]),
array([6.71938076]),
array([6.99082853]),
array([6.2468907]),
array([6.74499289]),
array([6.9829555]),
array([7.45402568]),
array([7.31737005]),
array([7.01059355]),
array([6.41016751]),
array([7.20972615]),
array([7.37309026]),
array([7.17595581]),
array([7.27107011]),
array([6.46388327]),
array([7.35044916]),
array([6.87915643]),
array([6.42261174]),
array([6.52567264]),
array([5.93006275]),
array([6.85580449]),
array([5.90084715]),
array([7.50822348]),
array([6.80425419]),
array([7.18830918]),
array([6.20833382]),
array([7.15066845]),
array([7.21802481]),
array([7.70494113]),
array([7.04808154]),
array([6.06739951]),
array([6.43301251]),
array([6.72755343]),
array([7.01738841]),
array([7.20700728]),
array([7.28282112]),
array([6.69759029]),
array([6.24299844]),
array([5.86905761]),
array([7.20742178]),
```

```
array([7.33048608]),
array([7.10628947]),
array([6.46789042]),
array([7.26431404]),
array([7.71687983]),
array([7.22459526]),
array([6.4398009]),
array([7.36472946]),
array([6.01611472]),
array([7.10628947]),
array([7.2608686]),
array([6.82188135]),
array([7.41885572]),
array([6.54996002]),
array([6.15422291]),
array([6.46388327]),
array([6.26982057]),
array([7.71342802]),
array([6.61318115]),
array([7.48293189]),
array([7.3484249]),
array([7.70362728]),
array([6.97688213]),
array([7.45709288]),
array([6.83612618]),
array([6.49798063]),
array([7.46648549]),
array([6.47214685]),
array([7.50003811]),
array([7.28446507]),
array([6.84601322]),
array([6.4179103]),
array([6.54831071]),
array([6.57492204]),
array([6.42418716]),
array([6.67528753]),
array([7.12508318]),
array([6.61764043]),
array([6.90116157]),
array([6.96086695]),
array([6.34771261]),
array([6.00274983]),
array([6.87162938]),
array([7.08023648]),
array([7.09109616]),
array([7.34412336]),
array([5.93203161]),
```

```
 array([7.40785571]),
 array([7.3787461]),
 …]
```

[195]:
```
predicted_price_euros = [np.exp(predicted[i][0]) for i in range(len(predicted))]
```

[196]:
```
predicted_price_euros
```

[196]:
```
[1313.4475540760347,
 1080.9760024052525,
 815.8164447810298,
 1704.9597040911785,
 1384.9204183050958,
 493.763468598171,
 1350.4157388245512,
 1080.9760024052525,
 1737.2181428292806,
 1210.3448505430486,
 528.3720501335043,
 483.9637617474174,
 1618.6940935868868,
 590.3610367601405,
 1218.7054606070317,
 1384.9204183050958,
 1042.9151306534152,
 1704.9597040911785,
 606.5381674573597,
 1278.6445810923815,
 340.0157817862232,
 1080.4542231896996,
 461.55771137332937,
 1188.2494806152295,
 1367.8637677439192,
 496.9427149978447,
 1080.9760024052525,
 932.7626514305545,
 1450.762851940882,
 825.661344545351,
 373.8985200043724,
 475.1646898709014,
 568.3814150905255,
 1799.3245537476723,
 1080.9760024052525,
 343.8747500236931,
 448.8745543079951,
 918.8142183795542,
 620.9987329482266,
```

954.4028876055584,
475.67765807281125,
1654.332173263316,
629.9952150581653,
775.8869025276975,
985.1307788681321,
1384.9204183050958,
484.7545587459919,
1267.0187001766658,
741.0627563519558,
427.85812892723305,
473.0743057859573,
1085.910391628238,
622.7387437029274,
1228.841237728272,
489.9568718496522,
1044.018839853881,
567.9416246681066,
689.591774194327,
1701.5087986156927,
531.1551543017787,
1203.9453146387855,
1211.9114811798452,
691.8083163994087,
1144.6032689396886,
997.7426231592772,
1685.6898637537254,
502.81031924660056,
359.1074642123964,
676.3383805097549,
1140.7818332782947,
1073.3501100300045,
1478.9884999012297,
1042.9151306534152,
1136.1764897001153,
560.1411788779343,
1129.8035705273237,
669.7018284120147,
998.5163049096423,
545.0668912261593,
912.5072468712318,
1165.9304624839758,
1452.0877561130226,
1057.6762238832964,
585.1663361004263,
495.1974212082641,
1518.0347980736565,

1357.663204331335,
991.4428207361976,
1522.486694493617,
1519.2626657907995,
927.1668460037471,
898.3694288358022,
1335.1058643932586,
1120.7487142220946,
1515.5152939635507,
1212.1429932206304,
801.1326975701271,
543.5814680786091,
601.9580184351081,
1363.4250266416757,
1067.395539253371,
557.8410049347151,
693.8375201856817,
1476.3937778107002,
1452.4914516222366,
829.775977953833,
518.3007736548176,
1463.617060784934,
1680.393000060767,
1146.1391067473971,
1585.991252360015,
1607.7300646312974,
1371.9555725874231,
706.9427489350959,
1373.7374696006825,
1370.7373291792699,
1205.585836881705,
1075.48769713502,
545.8398275500825,
1087.4925749434599,
550.4705564142234,
1676.1695006350928,
805.3007225938599,
1358.3560187645235,
552.9779110782393,
364.2788898133291,
824.7977692873059,
507.1455734754743,
1203.9453146387855,
970.6133412508578,
669.795506450975,
532.2953525329262,
1420.5650167580686,

599.3067534694961,
737.5603981601658,
1137.0470517733922,
363.6495416645651,
1195.8032433493279,
580.4924101558782,
955.2546452920003,
585.2265604871487,
1021.5667911722026,
1156.7739293542747,
641.4473657113691,
564.3513823444443,
1515.5152939635507,
1312.7002956456465,
397.0665634900571,
1834.8671745050133,
487.74975815108104,
1690.8433941016708,
1139.5662331078508,
677.1098868622697,
1798.606858865751,
1115.1870177978128,
762.3162035082945,
754.1274816033363,
1355.0178586973223,
1345.5947561649305,
507.1957002855442,
488.64451981265177,
797.5413582943823,
934.5467577246015,
1586.5134146725713,
376.63948607342365,
1665.7443376397343,
376.63948607342365,
1247.0184205671314,
1016.9880507167499,
741.7835264811018,
1420.702281331965,
690.7056868832022,
459.6742248655835,
509.8757126836992,
954.0388233006136,
547.8210421942607,
526.8324995129652,
2172.556325115714,
887.4752688501577,
1436.1706997219453,

1024.377546371479,
1373.2023115429727,
1785.610521037956,
807.9653021025781,
841.1512640291088,
1450.2556917580205,
1624.9040506087595,
987.7342206468996,
1372.9529079381655,
1022.8721196986284,
1832.3394808165297,
374.24393533809223,
1168.9769350400736,
729.8142693521597,
1055.1277507766408,
845.5983594032114,
2267.744513125731,
1131.346411931344,
1845.5156430353122,
1082.0824539337796,
1889.3445361560157,
660.3270987299311,
898.3694288358022,
1804.1417615618827,
1345.5584892752813,
1398.369790785932,
510.7893922080494,
1604.096675737608,
1526.7870271187765,
803.567452623691,
827.52197124031,
1796.7965969248598,
484.5077942624535,
785.9994870862392,
1654.676733769224,
1478.9884999012297,
644.2430087973806,
1125.2861512350175,
1214.2716869574294,
1636.7188119711811,
786.7621415009778,
1152.8414173804056,
1537.578219843214,
787.2068179839915,
2161.789924871148,
1065.856508498526,
993.9491130597922,

956.3165504367854,
942.1142214656429,
894.501573437588,
482.5137235441877,
500.7409680778384,
1743.9416425825036,
1051.3752570960316,
1179.368341713957,
1043.4938549456324,
448.8745543079951,
598.5468478753252,
2245.533500157505,
1111.3267671923213,
603.1517265232828,
1030.0641222282784,
919.6947836896793,
1114.9132921974578,
1513.9699430972366,
996.8166608402264,
681.2286828682506,
1984.0820713117116,
741.7835264811018,
1448.949206451504,
1025.7195319670325,
2068.402385703651,
500.76042111781396,
1616.1538849693034,
641.8660864170454,
1342.4350996553783,
670.0636182795458,
1102.4825878491702,
2126.5859062545355,
1469.668921044023,
1022.8721196986284,
527.6427562941689,
704.6450263367666,
674.975594394837,
782.0806747729232,
1122.4325406799699,
1599.8786323753195,
559.8304317486877,
924.0647138173211,
1078.452871511061,
1448.949206451504,
1462.7799595872232,
912.1490973976976,
1794.7376790713108,

788.9139674425074,
751.6105303851612,
907.3517200510172,
918.229410591983,
502.16519085078437,
839.2803794279235,
739.5522850465026,
496.0327858001821,
1107.3890478357503,
762.699815511753,
1173.3490836769477,
923.0172948196746,
1102.638270808074,
1140.674484501239,
1597.872842083816,
810.4505185401498,
348.36351969876154,
1256.1080787504413,
561.1125945245992,
1089.0000852060293,
976.840082781196,
1245.5283037797878,
936.8205426959101,
2022.7812491707346,
558.7817275351819,
1685.6898637537254,
573.1633314098805,
1796.7965969248598,
633.315035303936,
623.1647335204616,
1258.1841914801105,
365.59800218519194,
1647.3964932911565,
1112.3499880379607,
894.6646481574625,
523.3974360591277,
1186.374992467266,
1822.3928549412083,
1753.7392169905318,
833.5497697701766,
382.15105952529206,
879.1787148876789,
1799.5076741842013,
359.861181746034,
1039.0441542140748,
353.4003140806631,
1208.7917729028509,

626.0479123724342,
1114.218099592872,
760.4404711127163,
694.7554598033995,
343.8747500236931,
613.6803953204377,
1362.7603766282643,
1552.9990339159708,
1686.117408548426,
1430.9346449513491,
1036.1856455173208,
1085.5615485879512,
518.3007736548176,
1339.8696944198916,
1134.2988820757657,
1088.7658531878383,
1015.6691919510063,
365.03317224017223,
970.1650954521986,
372.36171920278866,
711.6844088327546,
682.4693681552416,
1785.610521037956,
1526.7870271187765,
1580.7082265184547,
335.3502343777735,
1637.6088533758732,
423.2162585545038,
984.6673234474105,
1624.9040506087595,
1104.2222549840492,
1322.7405474628765,
1601.4630198880757,
1398.244604810441,
1003.6389962017721,
365.59800218519194,
994.03296799385,
677.4036560151152,
1178.3881592908879,
499.21164181558225,
1018.7813856591196,
675.8246020519568,
799.7708600357863,
612.444350239108,
487.9551297411244,
893.6565316966029,
969.3723022118188,

1042.9151306534152,
1520.1451018668563,
1464.9756257293882,
531.0486709763491,
1502.0719061536317,
543.6222611475871,
1382.514489321366,
623.0953336597305,
490.38514103345227,
388.175751157877,
347.6479224812543,
827.52197124031,
1466.6003565358035,
512.988125053973,
1259.6191024124912,
1036.31200928287,
1741.6369834368136,
1680.4171351649043,
613.2664294590534,
1022.5540158149273,
1572.4252324451666,
1798.9329953929055,
1256.1080787504413,
1812.5627669306161,
718.8620415023177,
1417.4851428801378,
915.6006377423907,
858.8436734130141,
1068.886884167358,
1047.748610009593,
1709.6533155617074,
1824.6549112215598,
376.1781179133787,
1796.7965969248598,
928.8206345060865,
936.8205426959101,
1798.606858865751,
1609.7013973224807,
1752.5993655220827,
1611.4723038898567,
517.4187619538426,
343.09704104828575,
1683.8582018526176,
1612.4761262957531,
526.7355639448966,
1377.4237492231698,
1351.1104063404491,

634.7901409175652,
589.7386535649189,
1519.4725016194564,
667.7057502566104,
1412.061486342157,
1851.1791893367788,
409.91161521279054,
1075.420079900405,
614.536882172197,
2181.260984443421,
1261.059210743423,
1044.9635927046597,
1220.9014203224874,
1703.1426114112196,
441.496982882666,
343.74334817478734,
1221.109188063598,
607.2511855586563,
1968.8232798833944,
1371.9627341281407,
1267.0187001766658,
384.0459400713623,
1537.0984893077498,
1826.5349736958208,
641.5656041080595,
599.1195278445322,
740.1324420957371,
1423.492439927415,
450.3236347021175,
1262.9956666450855,
1026.5851910708138,
525.9741730331499,
842.7798303581878,
1130.3230626246113,
1796.3881678481682,
782.0565470304149,
1749.0876774978308,
1603.4189399033162,
1054.0674934591748,
997.6601066725199,
458.06290745570874,
1090.9269583393368,
677.6881112287831,
1600.2130052845105,
646.249158123318,
545.65508766954,
452.7792332375697,

774.808112005622,
1039.0924122817958,
1093.3063796913686,
371.6344540722165,
520.6586756941958,
794.2568145746009,
1497.8887345556511,
1641.466769939266,
1672.2560463086782,
821.5573273056117,
1162.2900203326433,
1380.0996917549835,
1151.2851080672513,
652.1164446799756,
1575.6342286755764,
1040.8404403962847,
782.4060792904386,
1215.4080379943534,
1198.1993302179628,
496.8726828335108,
839.521801052076,
373.8985200043724,
525.7250436194101,
1054.2799020132368,
1818.823692639784,
1384.058327005106,
1770.6846043451653,
1756.410684624088,
967.9098376244727,
1817.8612686640513,
1259.9163855598422,
587.6884364858131,
2001.4087930105661,
1271.1290581233072,
1070.1579688057877,
1275.5477371391619,
821.2384773834406,
983.7741613416447,
345.792740815261,
800.1770011589765,
1339.4803006430566,
378.3060970315643,
352.6928763579766,
1129.1559737635996,
1055.4195977543563,
685.403217252059,
1616.0925008914387,

1163.8782969838958,
1281.9109211033858,
736.710465389839,
779.6506530643888,
1175.1776252940538,
1101.413358926733,
331.86409435274123,
624.6719697637568,
1749.432846786322,
964.6281283865837,
1812.5627669306161,
1462.535452453275,
1520.3701120489916,
1050.8812312438736,
558.6979580461339,
798.0671792842351,
1030.9662486477023,
731.5142979456449,
555.0180779379173,
676.7341334557801,
997.1264914638941,
2010.742299778255,
875.7114190073528,
1199.7881874777856,
377.2921680690913,
1096.0775171330017,
1092.217346332676,
932.7626514305545,
510.07226460444224,
1257.8295927723534,
1127.8769366054573,
1607.7300646312974,
612.0523729287268,
1571.7581062241409,
888.9708108169411,
489.0191652702038,
567.2083977523175,
714.1539773370516,
559.8304317486877,
528.6490038223988,
1187.5171992907965,
555.0843655958374,
997.1264914638941,
2161.789924871148,
561.6090973127348,
928.1748087440711,
363.5285231656052,

371.64831790559646,
739.5522850465026,
629.4368067357482,
504.6606785251323,
440.8308071066902,
555.5824687515976,
1359.3338452228631,
1264.865811341621,
902.0956045981249,
1329.8500552629084,
641.5475298124076,
663.2787632144086,
398.1263842252672,
662.5119248557362,
1014.7123224894904,
502.3865309269695,
653.8111920251591,
964.5188554574521,
687.657846585351,
359.1319511067351,
1753.3816542642921,
2121.301905167154,
2126.5859062545355,
911.8721996201601,
380.7865042635477,
701.5551580839353,
618.4888476409212,
1290.5678120495995,
356.2025935047999,
1967.8290774535537,
1971.9773295276705,
1591.7421999452554,
760.5603573984203,
1146.3040890935379,
780.4244353914984,
933.0204070302665,
525.065909840694,
1201.171748756487,
490.967563976189,
545.1335046188067,
1617.4553931730284,
1055.1401979717493,
1431.5852365648327,
1577.3621661516315,
1104.2222549840492,
498.19215846736677,
377.59365348310075,

1944.427758162127,
470.19352409224706,
1811.3584409421242,
1112.4577642992606,
663.8253743498715,
1333.1765110823328,
356.3795199743008,
1107.819389587974,
1323.0923233949666,
457.2902011720008,
457.7836754120466,
541.8779159585251,
1394.6233020799068,
616.9945379863008,
1467.387881531813,
1803.5374156755277,
409.982598377794,
882.044484731899,
1054.0674934591748,
665.1299527616976,
802.4867852609054,
635.8061482844093,
893.084365190365,
340.5516514912063,
606.7330158236962,
779.1073298356761,
1089.0000852060293,
1738.186248502714,
1649.1659746763105,
804.4852957818309,
544.4531955320956,
480.31672861596786,
811.9333751373969,
1463.1246323976661,
327.1402627188275,
1527.407733691698,
1656.113876629992,
1242.751536183673,
610.9064452915444,
1751.242921051717,
1039.5965393397692,
491.45824605843296,
344.36300621044865,
1207.0400060452,
1746.236608252559,
1219.6137212877213,
1556.8956718172474,

2189.0520574900797,
888.9708108169411,
1373.2023115429727,
1274.9579100212334,
454.06512761620877,
875.7114190073528,
1042.9151306534152,
1175.1776252940538,
1397.1845511403808,
2331.4412243886636,
1054.7849201665117,
373.29710979203116,
677.0255622101357,
532.5073453370401,
1322.7405474628765,
554.6830628603769,
787.1541925287477,
883.3260824355834,
1036.1856455173208,
485.2739398169177,
1600.2130052845105,
887.4752688501577,
1368.81751577565,
529.3650433722163,
723.4107427359669,
1678.4830196762896,
1352.3164479692284,
618.3347773170941,
1562.504545153279,
1262.4971761014885,
1130.3230626246113,
417.4822432940424,
1420.9768932252975,
614.2182622629184,
1617.227868889046,
1998.9441270254042,
617.3883210125359,
504.6606785251323,
600.7805636009379,
373.0668239055341,
346.8697526435359,
861.2921558981303,
1415.754914557012,
1594.0816472253857,
1098.7728362752746,
356.0416025546285,
1722.653669346629,

440.35458563310846,
1757.6333021906332,
732.8107046802652,
992.3814622299666,
608.8690145925561,
583.0151146979043,
662.1706835368067,
450.3492743346416,
961.2315717486566,
1179.368341713957,
1825.1433092456498,
1053.208072117475,
602.1636104661278,
1242.751536183673,
888.6718294835773,
691.7053464630093,
373.78148017525115,
1128.9804242743528,
1022.2442406413155,
997.1264914638941,
489.7592251530199,
379.367033892427,
1091.1400982501343,
1676.4089340447651,
1113.894018238849,
1371.6762858857928,
2340.2954833192052,
1322.7405474628765,
520.4597364787306,
878.8024804307931,
641.3651415127578,
617.2554730618169,
554.5538773718356,
1958.0521280974162,
669.4929345164765,
613.8407889267488,
622.6613905149034,
1125.352393722941,
713.3953278325437,
1762.2960063551736,
941.7553315071212,
1692.9007261327952,
491.45824605843296,
1034.9608675087647,
635.336731185023,
1253.6421342246404,
1179.7930365553943,

1998.0746275827642,
341.7288978384569,
1676.7064444725831,
733.7257713915096,
911.0526429707792,
1275.965459613301,
474.1075921773153,
1015.6691919510063,
545.6787866827113,
1015.1796663081142,
714.4120397428416,
1435.5917853031144,
923.3276054447732,
1601.4630198880757,
2193.270438320927,
697.4030772468179,
1242.751536183673,
1697.5640385241256,
1247.0275493415068,
1417.6164552790758,
552.6017340150463,
775.0037130090534,
427.85812892723305,
1586.3520882432374,
1559.8414146887599,
364.4659874352877,
1204.1433085766844,
1575.8166529076166,
794.5976780032707,
1132.0290758477013,
1101.5400815577455,
1116.8967693491754,
1516.729983570944,
605.1248669111386,
1830.0605520933486,
655.6665643264744,
2328.0531578453333,
2075.5967775088993,
1736.4732136469017,
464.0233246736555,
1699.7199899638451,
1720.6821900618527,
1108.3121442457812,
909.2383353494305,
2094.3727178877684,
1117.919933213126,
1256.7380453172568,

359.76113612376923,
1617.4553931730284,
1104.4620709685182,
1000.695183577775,
1414.8560055658402,
355.85131833987634,
789.560855151633,
1513.9699430972366,
724.5309381776909,
1714.976907336131,
956.301969200098,
1373.2023115429727,
1153.940952398967,
481.9355858077356,
1186.5396068903694,
641.4473657113691,
594.2925079109976,
1862.5423116650359,
668.7166182823146,
1428.8967758802157,
1879.282437257671,
1368.3965202965226,
913.7594734074908,
932.7626514305545,
1542.3698745518027,
1825.2876287763297,
348.5499013227717,
2010.742299778255,
691.7507225941022,
487.3741024388781,
984.1812326878769,
1634.3235595910194,
1130.8193857672702,
1112.0977345457961,
459.33915276181085,
1355.0178586973223,
534.8124661327537,
362.0694323216603,
746.8810431270499,
2237.7389794505884,
1801.2159962850233,
582.4285869282645,
1054.0674934591748,
1185.148156422376,
1888.4376764628826,
1796.7965969248598,
509.8757126836992,

350.80900018815015,
796.7417479668616,
1786.053010658967,
2305.9454050484187,
715.7011232787763,
771.9590833373584,
1570.4569253413483,
808.6142301177367,
418.442121647332,
1331.4768416937848,
1383.895380622047,
1570.4569253413483,
406.1961899665824,
1530.279839830722,
827.8573491802742,
1403.3154442722082,
1079.8272024488904,
949.696499031779,
373.9878114294641,
1533.792042845099,
832.393414115147,
475.4391567644422,
806.1743868633134,
977.8075222276883,
937.321565121307,
898.097365858809,
1799.3245537476723,
699.543450896335,
1251.0812935154543,
1527.407733691698,
1059.024626628351,
1244.23445093839,
613.3979205043645,
488.43880174124945,
1123.8338673348358,
646.4975424245926,
1662.3460533937832,
1470.4536410405801,
365.03317224017223,
1314.327439193329,
741.7058431643056,
796.9532638625603,
1093.7550869397817,
713.0839466670583,
357.5107048729183,
913.4879191811599,
387.49941882856206,

1093.7550869397817,
470.19352409224706,
1150.9636555667194,
358.03584941415863,
509.48834435336266,
811.264888249847,
1707.4252710946507,
1522.5699382407815,
1371.6762858857928,
1873.5698138206676,
1220.32771195805,
1028.8170003170455,
914.9039764842652,
504.6606785251323,
824.5602272545929,
1659.9026858073928,
2139.2373167556343,
1651.0410085809335,
1218.8934191230446,
1076.6526215151393,
1690.6393428011274,
1530.5332725806186,
394.77096667873116,
917.3619430181238,
1637.4698103444562,
1051.3752570960316,
1214.7724711866701,
828.3044293829623,
1086.6214064410149,
516.4046697897141,
849.7930815394858,
1078.0999844176642,
1726.800722625201,
1506.237432335292,
1108.3121442457812,
607.9955153966926,
1352.5218352149957,
1592.5475743189425,
1307.6093356018741,
1438.088537393669,
641.5475298124076,
1556.8956718172474,
971.8062261968869,
615.6088262304598,
682.4386574194526,
376.1781179133787,
949.3755816401831,

365.3468385554536,
1822.9721207388613,
901.675036707041,
1323.8629047509137,
496.8726828335108,
1274.9579100212334,
1363.792648659642,
2219.286739234808,
1150.649145622701,
431.55696360105225,
622.0450467590182,
835.1016317216834,
1115.86861657772,
1348.8494917794953,
1455.0872170014936,
810.4505185401498,
514.3985930577261,
353.915298164422,
1349.4087077054076,
1526.1234042164647,
1219.6137212877213,
644.1234619350057,
1428.4054618027503,
2245.9409123726064,
1372.7828885572012,
626.2820928388049,
1579.2881016202684,
409.982598377794,
1219.6137212877213,
1423.492439927415,
917.7099180549405,
1667.1247665142494,
699.216218412697,
470.7009226574938,
641.5475298124076,
528.3825607477888,
2238.2017142941213,
744.8487312562253,
1777.444422969859,
1553.7472944717238,
2216.372839750507,
1071.5721300810862,
1732.105288608707,
930.8760935446044,
663.7998223834715,
1748.450919109969,
646.870973572549,

```
 1808.1113239806375,
 1457.4812773141118,
 940.1253549175115,
 612.7213725124775,
 698.063943201263,
 716.889731131416,
 616.5794315711494,
 792.575309144761,
 1242.751536183673,
 748.1776358027325,
 993.4279834177255,
 1054.5474056434407,
 571.184692000761,
 404.53968155780206,
 964.5188554574521,
 1188.2494806152295,
 1201.2238086226723,
 1547.078141011211,
 376.9194909004185,
 1648.886854112317,
 1601.5802909400284,
 …]
```

[197]:
```
X
```

[197]:

	Company	TypeName	Ram_GB	OpSys	Weight_KG	TouchScreen	\
0	Apple	Ultrabook	8	MaC	1.37	0	
1	Apple	Ultrabook	8	MaC	1.34	0	
2	HP	Notebook	8	Others	1.86	0	
3	Apple	Ultrabook	16	MaC	1.83	0	
4	Apple	Ultrabook	8	MaC	1.37	0	
…	…	…	…	…	…	…	
1298	Lenovo	2 in 1 Convertible	4	Windows	1.80	1	
1299	Lenovo	2 in 1 Convertible	16	Windows	1.30	1	
1300	Lenovo	Notebook	2	Windows	1.50	0	
1301	HP	Notebook	6	Windows	2.19	0	
1302	Asus	Notebook	4	Windows	2.20	0	

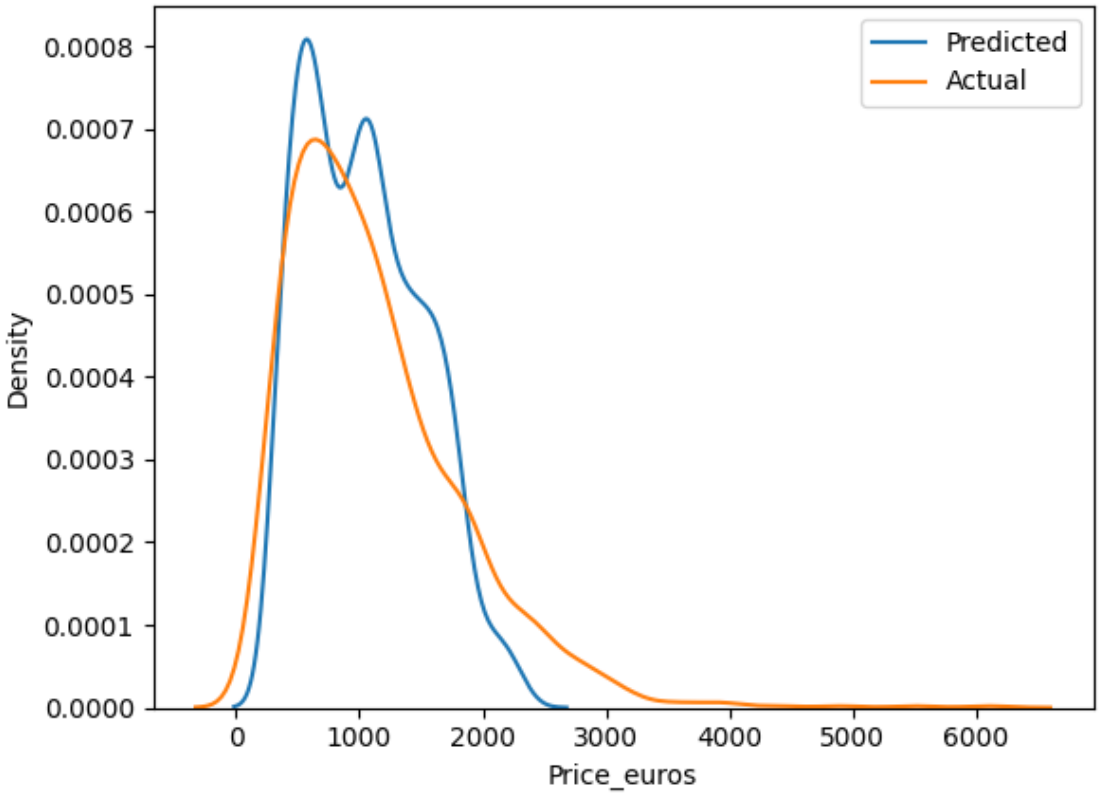
	IPS	PPI	Cpu_name	HDD	SSD	Gpu_brand
0	1	226.992481	Intel Core i5	0	128	Intel
1	0	127.669173	Intel Core i5	0	0	Intel
2	0	141.217949	Intel Core i5	0	256	Intel
3	1	220.519481	Intel Core i7	0	512	AMD
4	1	226.992481	Intel Core i5	0	256	Intel
…	…	…	…	…	…	…
1298	1	157.357143	Intel Core i7	0	128	Intel
1299	1	276.090226	Intel Core i7	0	512	Intel

```
1300    0  111.928571  Other Intel Processor     0    0     Intel
1301    0  100.448718          Intel Core i7  1000    0       AMD
1302    0  100.448718  Other Intel Processor   500    0     Intel

[1302 rows x 12 columns]
```

[198]:
```
X['predicted'] = np.array(predicted_price_euros)
X.head()
```

[198]:
```
  Company   TypeName  Ram_GB   OpSys  Weight_KG  TouchScreen  IPS         PPI  \
0   Apple  Ultrabook       8     MaC       1.37            0    1  226.992481
1   Apple  Ultrabook       8     MaC       1.34            0    0  127.669173
2      HP   Notebook       8  Others       1.86            0    0  141.217949
3   Apple  Ultrabook      16     MaC       1.83            0    1  220.519481
4   Apple  Ultrabook       8     MaC       1.37            0    1  226.992481

        Cpu_name  HDD  SSD Gpu_brand    predicted
0  Intel Core i5    0  128     Intel  1313.447554
1  Intel Core i5    0    0     Intel  1080.976002
2  Intel Core i5    0  256     Intel   815.816445
3  Intel Core i7    0  512       AMD  1704.959704
4  Intel Core i5    0  256     Intel  1384.920418
```

[199]:
```
# Let's compare the predicted price and Actual Price

sns.distplot(X['predicted'],hist = False, label = 'Predicted')
sns.distplot(df['Price_euros'], hist = False, label = 'Actual')
plt.legend()
plt.show()
```



[200]:
```
# Actually in RFRegV2 is not thinking is good model as its price is truncating␣
 ↪near about 3000 euros
# and maximum price it is not fitting good to actual data, we will not be using␣
 ↪it.

RFRegV3 = RandomForestRegressor(n_estimators = 100,
                                max_depth = 15,
                                max_features = 0.75,
                                max_samples = 0.5)

RFRegV3.fit(X_train_new,y_train)
```

[200]:
```
RandomForestRegressor(max_depth=15, max_features=0.75, max_samples=0.5)
```

[201]:
```
y_pred_RFRegV3 = RFRegV3.predict(X_test_new)
y_pred_RFRegV3
```

[201]:
```
array([6.75890169, 5.77457981, 5.87286816, 6.20932772, 7.19073002,
       7.49041674, 6.15451611, 6.64222505, 7.10201766, 5.7647322 ,
       7.09151745, 6.84697628, 7.54578918, 6.94632266, 7.51948084,
       7.06887054, 7.05428943, 5.79231086, 7.52492245, 7.53123969,
```

```
7.44294303, 7.32341507, 6.61800832, 7.05108015, 6.36651123,
7.53648515, 6.97462539, 6.10084065, 6.99324267, 6.53194493,
7.18586694, 6.24140099, 6.8089036 , 6.91604154, 7.01924731,
6.72663836, 7.11448366, 7.23573475, 7.52982978, 7.44371395,
7.55103715, 7.21133751, 5.90951207, 5.79053278, 6.15824277,
5.92651513, 6.89917944, 7.58460268, 7.31940754, 7.09418429,
7.71543831, 7.54257532, 5.97501496, 5.46725534, 6.45522192,
6.84722195, 6.09949742, 7.20089461, 6.11121723, 6.46241169,
7.24174108, 7.01062637, 5.48771125, 7.32220734, 7.03545802,
6.6203688 , 6.0479565 , 7.25683316, 6.70022363, 5.81449335,
7.01064557, 6.59539841, 6.67485096, 6.4996227 , 7.16133641,
6.93218826, 7.30406962, 7.97804656, 7.31131826, 6.53372428,
6.84003889, 7.08483648, 7.44576757, 6.10496862, 7.27239807,
6.45707616, 6.99661407, 7.39912697, 6.12419323, 6.66817928,
6.47041438, 6.30858725, 6.61252419, 7.64637565, 6.29393117,
5.72172493, 6.50146909, 6.2902127 , 5.85850982, 7.75088096,
6.5641951 , 7.26346513, 6.06756438, 7.58332444, 6.81743384,
7.15593704, 7.32372022, 6.3266494 , 6.90532832, 5.90064466,
6.74414028, 6.92292986, 6.54294243, 7.52610371, 6.15245666,
6.75890169, 5.76981914, 7.68448991, 6.59313919, 6.78650884,
6.46323346, 7.50161111, 6.96667989, 6.504053  , 6.53169343,
7.21379394, 6.53657081, 5.96423874, 7.58586663, 7.45330351,
6.43919835, 7.28383092, 6.31837928, 7.19685532, 7.22013276,
6.54968677, 7.09347439, 7.15593704, 7.17600298, 7.52610371,
5.78242092, 5.50785784, 6.8443615 , 6.42032512, 6.68801071,
7.3997449 , 6.69169041, 5.76796614, 7.33740115, 6.33466629,
7.79841929, 7.2344553 , 6.86649895, 7.09614458, 6.99592573,
6.6203688 , 7.07062931, 7.75647612, 7.13397343, 7.37690912,
5.47951702, 6.17795175, 6.7056099 , 6.53082109, 7.04595033,
7.19251514, 7.5070631 , 6.23015447, 7.40476411, 7.11308005,
6.18269527, 6.9645497 , 6.0253667 , 7.17942333, 6.92387533,
7.06808842, 7.54101279, 7.5999463 , 7.06834904, 6.1571939 ,
6.7948869 , 7.11850868, 7.12251285, 6.81458557, 6.09688455,
6.77902955, 6.57694772, 7.36311548, 6.7893782 , 6.05359484,
6.15190734, 7.24230915, 5.87084706, 7.04512962, 7.38487197,
7.47695917])
```

[202]:
```
print('R2_sore: ', metrics.r2_score(y_test,y_pred_RFRegV2))
print('MAE: ', metrics.mean_absolute_error(y_test,y_pred_RFRegV2))
print('MSE: ',metrics.mean_squared_error(y_test,y_pred_RFRegV2))

# Calculate R-squared scores
train_r2 = metrics.r2_score(y_train, RFRegV2.predict(X_train_new))
test_r2 = metrics.r2_score(y_test, y_pred_RFRegV2)

# Print the results
print(f'Training R-squared: {train_r2}')
```

```
print(f'Test R-squared: {test_r2}')
```

```
R2_sore:  0.83572357883684
MAE:  0.19721247060802555
MSE:  0.06284412886011013
Training R-squared: 0.8312606138533705
Test R-squared: 0.83572357883684
```

[203]:
```
predicted = []

npArrar_df = np.array(train_new)
for i in range(len(npArrar_df)):
    predicted.append(RFRegV3.predict([npArrar_df[i]]))

predicted
```

[203]:
```
[array([7.18885712]),
 array([6.90532832]),
 array([6.5196534]),
 array([7.82666139]),
 array([7.33740115]),
 array([5.95444151]),
 array([7.39652949]),
 array([6.90532832]),
 array([7.4468491]),
 array([6.81047553]),
 array([6.22539101]),
 array([6.00659289]),
 array([7.75141659]),
 array([6.28600667]),
 array([7.1830591]),
 array([7.33740115]),
 array([6.75611314]),
 array([7.82666139]),
 array([6.33385492]),
 array([7.03711986]),
 array([5.48771125]),
 array([6.92110147]),
 array([5.74847883]),
 array([6.79336726]),
 array([6.97404482]),
 array([6.14057916]),
 array([6.94182691]),
 array([6.70559269]),
 array([7.23753251]),
 array([6.80934357]),
 array([5.62758212]),
```

```
array([5.93528965]),
array([6.18833338]),
array([7.67837238]),
array([6.94182691]),
array([5.66648023]),
array([5.96977058]),
array([6.92387533]),
array([6.41716412]),
array([6.81402916]),
array([6.07510481]),
array([7.32630353]),
array([6.38276765]),
array([6.57663961]),
array([6.98219116]),
array([7.33740115]),
array([6.0479565]),
array([7.16133641]),
array([6.51070638]),
array([6.10606393]),
array([6.05359484]),
array([6.85130714]),
array([6.10631764]),
array([7.09195383]),
array([6.07380093]),
array([6.80245613]),
array([6.23826307]),
array([6.53617562]),
array([7.5070631]),
array([6.14717874]),
array([7.2094635]),
array([6.99661407]),
array([6.55592003]),
array([7.01168631]),
array([6.9635101]),
array([7.52186431]),
array([6.09688455]),
array([5.78242092]),
array([6.54968677]),
array([7.01057577]),
array([7.06834904]),
array([7.07062931]),
array([6.75611314]),
array([7.04640153]),
array([6.23236659]),
array([6.91442241]),
array([6.36747474]),
array([6.89930277]),
```

```
array([6.26822522]),
array([6.88201218]),
array([7.01953582]),
array([7.35569469]),
array([6.916266]),
array([6.17795175]),
array([6.00181602]),
array([7.23484849]),
array([6.88243823]),
array([6.78066547]),
array([7.34925433]),
array([7.32830509]),
array([6.68511445]),
array([6.65703652]),
array([7.25384207]),
array([6.95699289]),
array([7.06712489]),
array([6.92992241]),
array([6.5283281]),
array([6.18821764]),
array([6.34191263]),
array([7.13397343]),
array([6.87220413]),
array([6.15190734]),
array([6.46323346]),
array([7.08966412]),
array([7.33615553]),
array([6.68414103]),
array([6.15451611]),
array([7.06013071]),
array([7.34901135]),
array([7.10295191]),
array([7.31243085]),
array([7.31014063]),
array([7.3029191]),
array([6.66508185]),
array([7.28737093]),
array([7.28697892]),
array([7.04814726]),
array([6.96992413]),
array([6.29135227]),
array([7.02809901]),
array([6.25256198]),
array([7.54553916]),
array([6.49420147]),
array([7.30437809]),
array([6.07814576]),
```

```
array([5.70083954]),
array([6.74088433]),
array([6.24140099]),
array([7.2094635]),
array([6.95284615]),
array([6.47553278]),
array([6.22387669]),
array([6.93858555]),
array([6.30926305]),
array([6.45141361]),
array([6.9754266]),
array([5.67757116]),
array([7.19251514]),
array([6.24133001]),
array([6.90725387]),
array([6.44201275]),
array([7.03545802]),
array([6.89620809]),
array([6.52210733]),
array([5.98889488]),
array([7.06712489]),
array([7.32341507]),
array([5.79607818]),
array([7.5858177]),
array([6.14682181]),
array([7.25339109]),
array([6.93666804]),
array([6.44018209]),
array([7.57976454]),
array([7.06440098]),
array([6.61521751]),
array([6.47162467]),
array([7.08483648]),
array([7.04250993]),
array([6.1972043]),
array([6.00237652]),
array([6.6167361]),
array([6.72132858]),
array([7.14648347]),
array([5.84519916]),
array([7.41193268]),
array([5.84519916]),
array([7.01610105]),
array([6.99686669]),
array([6.75890169]),
array([7.27594269]),
array([6.55731621]),
```

```
array([5.87084706]),
array([6.26569488]),
array([6.88604868]),
array([6.21108469]),
array([6.15421851]),
array([7.97265921]),
array([6.77999663]),
array([7.3441186]),
array([7.08377585]),
array([7.32372022]),
array([7.53648515]),
array([6.64632902]),
array([6.7769958]),
array([7.41357998]),
array([7.65367325]),
array([6.77280597]),
array([7.24174108]),
array([7.01266809]),
array([7.78983282]),
array([5.52812775]),
array([7.02205405]),
array([6.55207427]),
array([6.78826184]),
array([6.81766259]),
array([8.38909358]),
array([7.01720757]),
array([7.40801466]),
array([7.01064557]),
array([7.55212143]),
array([6.38248871]),
array([6.6536423]),
array([7.41304751]),
array([7.77064705]),
array([7.27356187]),
array([6.14170643]),
array([7.38141561]),
array([7.31252815]),
array([6.51386855]),
array([6.61252419]),
array([7.54492399]),
array([6.10084065]),
array([6.41989307]),
array([7.45711514]),
array([7.07062931]),
array([6.45374664]),
array([6.98587653]),
array([6.94818661]),
```

```
array([7.33642146]),
array([6.54991729]),
array([6.94366512]),
array([7.29098794]),
array([6.69817549]),
array([7.85152691]),
array([6.8019382]),
array([6.92292986]),
array([6.86235892]),
array([6.91281464]),
array([6.90639984]),
array([6.11121723]),
array([5.95959839]),
array([7.37690912]),
array([6.98353377]),
array([6.7612284]),
array([7.06887054]),
array([5.96977058]),
array([6.2181456]),
array([8.06646101]),
array([6.89917944]),
array([6.30858725]),
array([6.94394998]),
array([6.84770179]),
array([6.88399123]),
array([7.25410731]),
array([6.82526783]),
array([6.40766805]),
array([7.79455082]),
array([6.75890169]),
array([7.44294303]),
array([6.80519214]),
array([7.65341869]),
array([6.16670052]),
array([7.50713578]),
array([6.45919313]),
array([7.11283023]),
array([6.2954458]),
array([7.08062206]),
array([7.82570924]),
array([7.2344553]),
array([7.01266809]),
array([6.12514344]),
array([6.63411147]),
array([6.53082109]),
array([6.48587469]),
array([6.9645497]),
```

```
array([7.2415896]),
array([6.20065169]),
array([6.91638964]),
array([6.92804109]),
array([7.44294303]),
array([7.46387917]),
array([6.67485096]),
array([7.5999463]),
array([6.58052176]),
array([6.53308101]),
array([6.88077848]),
array([6.69006228]),
array([6.20526969]),
array([6.78650884]),
array([6.70022363]),
array([6.12944352]),
array([6.86396065]),
array([6.54294243]),
array([7.17942333]),
array([6.77902955]),
array([7.05453718]),
array([6.97462539]),
array([7.15314543]),
array([6.5385122]),
array([5.48842606]),
array([7.0776975]),
array([6.26072244]),
array([7.11448366]),
array([6.83074978]),
array([7.12439986]),
array([6.8089036]),
array([7.93594882]),
array([6.23030834]),
array([7.40741376]),
array([6.45211776]),
array([7.54492399]),
array([6.49011466]),
array([6.37369201]),
array([7.10425337]),
array([5.80904797]),
array([7.47148019]),
array([6.93542991]),
array([6.79172866]),
array([6.14743733]),
array([7.13653228]),
array([7.73588353]),
array([7.55963074]),
```

```
array([6.54993451]),
array([5.63891344]),
array([6.76400358]),
array([7.67517528]),
array([5.87922636]),
array([6.96667989]),
array([5.85143299]),
array([7.22622715]),
array([6.45296217]),
array([7.04512962]),
array([6.47041438]),
array([6.44571023]),
array([5.66648023]),
array([6.37688109]),
array([6.962203]),
array([7.32709047]),
array([7.89283928]),
array([7.21379394]),
array([7.0143894]),
array([7.10040444]),
array([6.15451611]),
array([6.96247073]),
array([7.11308005]),
array([6.94088518]),
array([7.1702719]),
array([5.78520031]),
array([6.94742768]),
array([5.79107888]),
array([6.66896212]),
array([6.57489702]),
array([7.53648515]),
array([7.31252815]),
array([7.17006451]),
array([5.50785784]),
array([7.50161111]),
array([5.98673957]),
array([6.67442053]),
array([7.65367325]),
array([7.09347439]),
array([7.15593704]),
array([7.53708143]),
array([7.34541779]),
array([6.89678058]),
array([5.80904797]),
array([6.8762039]),
array([6.4043589]),
array([7.27584831]),
```

```
array([6.11916869]),
array([6.96342856]),
array([6.5079555]),
array([6.70322205]),
array([6.37133724]),
array([6.0668733]),
array([6.92041714]),
array([6.82785832]),
array([6.81288974]),
array([7.3596645]),
array([7.19685532]),
array([6.03640461]),
array([7.55413584]),
array([6.16128319]),
array([6.7904178]),
array([6.1571939]),
array([6.22596621]),
array([5.77926043]),
array([5.70634382]),
array([6.61252419]),
array([7.26680919]),
array([6.06754346]),
array([7.19108698]),
array([7.00147446]),
array([7.52671964]),
array([7.57894454]),
array([6.47029703]),
array([7.21146815]),
array([7.58332444]),
array([7.64023095]),
array([7.0776975]),
array([7.52610371]),
array([6.55958007]),
array([7.21721309]),
array([6.6861742]),
array([6.35668048]),
array([6.95102423]),
array([7.00673963]),
array([7.71614675]),
array([7.75647612]),
array([5.85850982]),
array([7.54492399]),
array([6.84722195]),
array([6.8089036]),
array([7.57976454]),
array([7.21133751]),
array([7.55592496]),
```

```
array([7.45330351]),
array([6.1366289]),
array([5.57301235]),
array([7.54884262]),
array([7.46986101]),
array([6.21453823]),
array([6.8258545]),
array([6.92648972]),
array([6.36011656]),
array([6.51477546]),
array([7.35629695]),
array([6.42083626]),
array([7.12251285]),
array([7.57560106]),
array([6.05928963]),
array([7.08281566]),
array([6.40798669]),
array([7.87934451]),
array([7.0352243]),
array([6.94276112]),
array([6.94961554]),
array([7.44845574]),
array([5.88055461]),
array([5.67366674]),
array([7.0796908]),
array([6.41276216]),
array([7.79783124]),
array([7.4363768]),
array([7.16133641]),
array([5.81834858]),
array([7.68652357]),
array([7.72062654]),
array([6.36651123]),
array([6.30917139]),
array([6.53169343]),
array([7.34195136]),
array([6.01984991]),
array([7.06913555]),
array([7.01506168]),
array([6.23543967]),
array([7.19852828]),
array([6.95719221]),
array([7.73342703]),
array([6.44050566]),
array([7.53365278]),
array([7.58460268]),
array([7.09418429]),
```

```
array([6.83837009]),
array([6.0253667]),
array([7.2894594]),
array([6.3266494]),
array([7.43819701]),
array([6.41020872]),
array([6.27994027]),
array([6.14308894]),
array([6.62520579]),
array([6.91807335]),
array([7.18849217]),
array([5.79170369]),
array([6.19583554]),
array([6.66783695]),
array([7.4121647]),
array([7.51767253]),
array([7.47695917]),
array([6.89085583]),
array([7.16032676]),
array([7.28045447]),
array([7.21756893]),
array([6.54110665]),
array([7.56893538]),
array([7.05428943]),
array([6.53220735]),
array([7.27476256]),
array([7.08013698]),
array([6.28749491]),
array([6.78146441]),
array([5.62758212]),
array([6.10496862]),
array([7.07146259]),
array([7.54101279]),
array([7.39787576]),
array([7.56338228]),
array([7.56183077]),
array([6.84681595]),
array([7.82651425]),
array([7.15962641]),
array([6.31030858]),
array([7.55705365]),
array([7.0520437]),
array([6.91221409]),
array([7.31227769]),
array([6.88086776]),
array([6.93064532]),
array([5.74215144]),
```

```
array([6.4930681]),
array([7.21357258]),
array([5.75126169]),
array([5.69021509]),
array([7.02711019]),
array([7.03914718]),
array([6.49925369]),
array([7.35300771]),
array([6.94578151]),
array([6.98790374]),
array([6.56100384]),
array([6.59539841]),
array([7.10752452]),
array([7.27267922]),
array([5.47157944]),
array([6.44489989]),
array([7.66760688]),
array([6.83673024]),
array([7.52610371]),
array([7.27312268]),
array([7.28117775]),
array([6.99324267]),
array([6.42366536]),
array([6.75047946]),
array([7.01323008]),
array([6.62768893]),
array([6.19309899]),
array([6.5087985]),
array([6.99592573]),
array([7.79841929]),
array([6.84835392]),
array([7.28383092]),
array([5.7352217]),
array([7.15256024]),
array([7.08191844]),
array([6.70559269]),
array([6.09094674]),
array([7.26765268]),
array([6.98050413]),
array([7.30140366]),
array([6.56954896]),
array([7.18625583]),
array([6.65867583]),
array([5.93979211]),
array([6.36183801]),
array([6.72664774]),
array([6.20932772]),
```

```
array([6.16528327]),
array([7.18586694]),
array([6.23015447]),
array([6.99592573]),
array([7.85152691]),
array([6.28034485]),
array([7.02195969]),
array([5.77457981]),
array([5.6687499]),
array([6.70022363]),
array([6.31952075]),
array([6.36077265]),
array([5.9561068]),
array([6.30354747]),
array([7.60514648]),
array([7.58428169]),
array([6.80641806]),
array([7.2696396]),
array([6.73748773]),
array([6.49223126]),
array([5.8613689]),
array([6.73988168]),
array([7.3089655]),
array([6.14514392]),
array([6.37952145]),
array([6.92434617]),
array([6.69169041]),
array([5.76981914]),
array([7.56617883]),
array([7.97804656]),
array([7.82570924]),
array([6.65972324]),
array([5.79886748]),
array([6.53194493]),
array([6.58983742]),
array([7.33274592]),
array([5.73735687]),
array([7.76525107]),
array([7.50312286]),
array([7.39315538]),
array([6.42745928]),
array([7.24398552]),
array([6.7056099]),
array([6.92876457]),
array([6.2902127]),
array([7.30126144]),
array([6.14141845]),
```

```
array([6.14845376]),
array([7.52982978]),
array([7.11716845]),
array([7.41220696]),
array([7.49855168]),
array([7.09347439]),
array([6.34561733]),
array([5.7667944]),
array([7.70923748]),
array([6.12274257]),
array([7.67784067]),
array([7.0779639]),
array([6.41848266]),
array([7.06824953]),
array([5.78870297]),
array([8.0477164]),
array([7.00526633]),
array([6.06916996]),
array([6.00590229]),
array([6.45085065]),
array([7.22013276]),
array([6.32714306]),
array([7.21185959]),
array([7.54257532]),
array([5.97501496]),
array([6.81600656]),
array([7.09418429]),
array([6.55351684]),
array([6.59363035]),
array([6.6203688]),
array([6.70586839]),
array([5.49777071]),
array([6.23150963]),
array([6.61156285]),
array([7.11448366]),
array([7.53262459]),
array([7.30859092]),
array([6.70296947]),
array([6.25999624]),
array([6.05276381]),
array([6.75043514]),
array([7.38692156]),
array([5.52434276]),
array([7.45108407]),
array([7.76507789]),
array([7.01062637]),
array([6.3541128]),
```

```
array([7.67340068]),
array([6.90610828]),
array([6.11819224]),
array([5.65934671]),
array([6.92701154]),
array([7.75489549]),
array([7.25391574]),
array([7.391713]),
array([7.7715491]),
array([6.65867583]),
array([7.31362707]),
array([7.10201766]),
array([5.92651513]),
array([6.84835392]),
array([6.75611314]),
array([7.10752452]),
array([7.26346513]),
array([8.04323537]),
array([7.01650309]),
array([5.90301704]),
array([6.5641951]),
array([6.2948999]),
array([7.15593704]),
array([6.18269527]),
array([6.73104925]),
array([6.70482715]),
array([7.0143894]),
array([6.03548004]),
array([7.43688716]),
array([6.77999663]),
array([7.1751419]),
array([5.93299634]),
array([6.57412569]),
array([7.34518102]),
array([7.03995832]),
array([6.63029529]),
array([7.51766099]),
array([7.26670073]),
array([6.95719221]),
array([6.02398901]),
array([7.55478118]),
array([6.44426643]),
array([7.42463945]),
array([7.78831519]),
array([6.51451062]),
array([6.28575335]),
array([6.38232029]),
```

```
array([5.78499567]),
array([5.82575794]),
array([6.71403716]),
array([7.54929218]),
array([7.63271614]),
array([6.96087781]),
array([5.76467367]),
array([7.50136401]),
array([6.06268711]),
array([7.53214848]),
array([6.56175443]),
array([6.79690974]),
array([6.15552855]),
array([6.19885179]),
array([6.45139848]),
array([5.97717861]),
array([6.7948869]),
array([6.7612284]),
array([7.51948084]),
array([7.10502311]),
array([6.40084184]),
array([7.01062637]),
array([6.84012528]),
array([6.55378234]),
array([5.72876937]),
array([7.25928486]),
array([6.94632266]),
array([6.99592573]),
array([5.81793714]),
array([5.67433724]),
array([7.07801241]),
array([7.51330797]),
array([6.84540536]),
array([7.31969521]),
array([8.13178093]),
array([7.15593704]),
array([6.20949676]),
array([6.70896931]),
array([6.46241169]),
array([6.43406368]),
array([6.12419323]),
array([7.67138894]),
array([6.47479272]),
array([6.29393117]),
array([6.64222505]),
array([6.97657545]),
array([6.60785193]),
```

```
array([7.5691464]),
array([6.66079997]),
array([7.52492245]),
array([6.11819224]),
array([7.04595033]),
array([6.46370202]),
array([7.29532591]),
array([6.95174815]),
array([7.93078424]),
array([5.46725534]),
array([7.50974956]),
array([6.56328859]),
array([6.92000411]),
array([7.83757977]),
array([6.19544558]),
array([7.1702719]),
array([6.29922673]),
array([6.85746516]),
array([6.59968837]),
array([7.28569037]),
array([6.86244279]),
array([7.54578918]),
array([7.88628897]),
array([6.47828752]),
array([7.01062637]),
array([7.49041674]),
array([7.22110729]),
array([7.06808842]),
array([6.57334061]),
array([6.6437037]),
array([6.10606393]),
array([7.29055904]),
array([7.36347063]),
array([5.7647322]),
array([7.15332206]),
array([7.43725342]),
array([6.80407891]),
array([6.99590245]),
array([7.15335851]),
array([6.91963318]),
array([7.75088096]),
array([6.43545802]),
array([7.78121349]),
array([6.61557084]),
array([8.15294304]),
array([7.70625192]),
array([7.51686483]),
```

```
array([5.98427602]),
array([7.64268328]),
array([7.52397166]),
array([7.00163405]),
array([6.66817928]),
array([7.7651602]),
array([6.93322176]),
array([7.15397051]),
array([5.44656741]),
array([7.52982978]),
array([6.88591216]),
array([7.05276638]),
array([7.485194]),
array([5.75265001]),
array([6.57639738]),
array([7.25410731]),
array([6.62362181]),
array([7.6644412]),
array([6.92569068]),
array([7.2989563]),
array([7.31152784]),
array([6.31639736]),
array([7.23039128]),
array([6.52210733]),
array([6.34653658]),
array([7.57618036]),
array([6.45006241]),
array([7.41197302]),
array([7.6475191]),
array([7.31131826]),
array([6.79638557]),
array([6.70964162]),
array([7.43979111]),
array([7.63062888]),
array([5.80401028]),
array([7.79357836]),
array([6.71410005]),
array([6.13820336]),
array([7.01728966]),
array([7.59578751]),
array([6.95447819]),
array([7.09151745]),
array([6.13230598]),
array([7.08483648]),
array([6.36401323]),
array([5.80853547]),
array([6.66795557]),
```

```
array([8.17882351]),
array([7.676753]),
array([6.26893921]),
array([7.1110461]),
array([7.26937199]),
array([7.67577348]),
array([7.54492399]),
array([6.26569488]),
array([5.75442565]),
array([6.61319722]),
array([7.53123969]),
array([8.03490276]),
array([6.66348625]),
array([6.55581902]),
array([7.36400731]),
array([6.61005115]),
array([6.0911785]),
array([7.05108015]),
array([7.50417647]),
array([7.36400731]),
array([5.94597522]),
array([7.39912697]),
array([6.78458328]),
array([7.09614458]),
array([6.83798679]),
array([6.88090861]),
array([5.81449335]),
array([7.57661702]),
array([6.64773906]),
array([6.10608317]),
array([6.6819128]),
array([7.10391361]),
array([7.07032686]),
array([6.79084761]),
array([7.64492452]),
array([6.76544887]),
array([7.13600924]),
array([7.45108407]),
array([6.94092349]),
array([7.31940754]),
array([6.65114496]),
array([6.13264297]),
array([7.09024366]),
array([6.504053]),
array([7.40476411]),
array([7.17515197]),
array([5.78520031]),
```

```
array([7.11850868]),
array([6.62459532]),
array([6.81743384]),
array([7.19073002]),
array([6.68663081]),
array([5.77556836]),
array([6.72663836]),
array([5.90064466]),
array([7.19073002]),
array([6.12274257]),
array([7.17503518]),
array([5.75987634]),
array([6.09949742]),
array([6.96742594]),
array([7.4075835]),
array([7.35493416]),
array([7.31969521]),
array([7.6918228]),
array([7.29386183]),
array([6.88000934]),
array([6.79016904]),
array([6.36077265]),
array([6.85004922]),
array([7.47505806]),
array([7.61145348]),
array([7.40958423]),
array([7.28678097]),
array([7.2592219]),
array([7.52758085]),
array([7.41950384]),
array([5.83286527]),
array([6.78943995]),
array([7.50603843]),
array([6.98353377]),
array([7.27239807]),
array([6.65125396]),
array([6.93466642]),
array([6.15824277]),
array([6.85029438]),
array([6.96802833]),
array([7.44371395]),
array([7.20805999]),
array([7.00163405]),
array([6.51998646]),
array([6.93218826]),
array([7.23188914]),
array([7.36311548]),
```

```
array([7.38563558]),
array([6.7046391]),
array([7.391713]),
array([6.99072201]),
array([6.49699629]),
array([6.86649895]),
array([5.85850982]),
array([7.19765951]),
array([5.71317423]),
array([7.58374167]),
array([6.91394573]),
array([7.37712036]),
array([6.28749491]),
array([7.10201766]),
array([7.3256991]),
array([7.87596068]),
array([7.1981019]),
array([6.11272339]),
array([6.48990044]),
array([6.69845496]),
array([7.32220734]),
array([7.10128422]),
array([7.34146443]),
array([6.5385122]),
array([6.34229305]),
array([5.79071995]),
array([7.56743648]),
array([7.40723583]),
array([7.25391574]),
array([6.7893782]),
array([7.18437162]),
array([7.92421925]),
array([7.3885407]),
array([6.41804834]),
array([7.22557547]),
array([5.97501496]),
array([7.25391574]),
array([7.34195136]),
array([6.88008549]),
array([7.47567411]),
array([6.56499124]),
array([6.06756438]),
array([6.68778002]),
array([6.3496886]),
array([8.00382123]),
array([6.56826382]),
array([7.58139782]),
```

```
        array([7.49178243]),
        array([7.9790055]),
        array([7.02369699]),
        array([7.50881878]),
        array([6.89376707]),
        array([6.75107874]),
        array([7.64637565]),
        array([6.51475118]),
        array([7.52270333]),
        array([7.17600298]),
        array([7.04809074]),
        array([6.29996578]),
        array([6.48801235]),
        array([6.76855291]),
        array([6.4996227]),
        array([6.74414028]),
        array([7.01062637]),
        array([6.54414879]),
        array([6.93388525]),
        array([7.35306822]),
        array([6.25596538]),
        array([5.90951207]),
        array([6.92434617]),
        array([6.84697628]),
        array([7.12021594]),
        array([7.38487197]),
        array([5.87863951]),
        array([7.40848106]),
        array([7.5004086]),
        …]
```

[204]:
```
predicted_price_eruro = [np.exp(predicted[i][0]) for i in range(len(predicted))]
```

[205]:
```
predicted_price_eruro
```

[205]:
```
[1324.5884951345706,
 997.5759796670696,
 678.3432339014422,
 2506.547005886473,
 1536.7132265255912,
 385.4615758980789,
 1630.316564865233,
 997.5759796670696,
 1714.4525550630858,
 907.3021551125644,
 505.42062939000476,
 406.09734249006226,
```

2324.863469864135,
537.0046074115158,
1316.9307285242626,
1536.7132265255912,
859.2957328394078,
2506.547005886473,
563.3239802078745,
1138.1049782588866,
241.7033738540302,
1013.435649468648,
313.71308536154527,
891.9118140346482,
1068.5360597938645,
464.32240955841763,
1034.658719205057,
816.962086413135,
1390.65829833147,
906.2757084955103,
277.989158633539,
378.14951109940233,
487.033727757612,
2161.0994673626687,
1034.658719205057,
289.0154735280634,
391.4158616948721,
1016.250679710212,
612.2643444035256,
910.5321084379292,
434.8950793131376,
1519.7536497288813,
591.562685035749,
718.1221001904499,
1077.2762672750564,
1536.7132265255912,
423.2472380117788,
1288.631917699457,
672.3011526356194,
448.5696333612637,
425.6403929527667,
945.1154971801697,
448.6834551495873,
1202.2545093909805,
434.32840195310575,
900.0552345773663,
511.96848261242536,
689.64406422195,
1820.858011388308,

467.3968762097932,
1352.1666371128777,
1092.9263181078463,
703.3959962499338,
1109.5239260199235,
1057.3384136889315,
1848.0093435783597,
444.4708858030099,
324.5439339263809,
699.0251806917267,
1108.2924413021062,
1174.207858353366,
1176.8884309767636,
859.2957328394078,
1148.7176760424106,
508.958557505979,
1006.6894075575649,
582.5847925392861,
991.5831087588618,
527.5402807730861,
974.5854252469883,
1118.2674172010807,
1565.0838594274555,
1008.5470390884968,
482.00367817290334,
404.1620926949323,
1386.9307416115319,
975.0007385913879,
880.6545768376087,
1555.0365587769668,
1522.798577736828,
800.4022818697048,
778.2412115277499,
1413.5252934897683,
1050.4699193433921,
1172.7713398853525,
1022.4146469852293,
684.2532545149412,
486.97736542589024,
567.8814228266765,
1253.8491664790097,
965.0733750617945,
469.6122430712983,
641.1307813012182,
1199.5048466534827,
1534.8002598543346,
799.6235288514902,

470.83895371932834,
1164.5973792909601,
1554.6587529273427,
1215.5499833385952,
1498.8161544750035,
1495.3874676326006,
1484.6273830066466,
784.5276777831748,
1461.722667327901,
1461.1497659021338,
1150.7247780143432,
1064.142006312431,
539.8829013240668,
1127.884474612347,
519.3416637747002,
1892.282696287707,
661.2959499368321,
1486.7950176265433,
436.2195867805411,
299.1184164204199,
846.3088236496776,
513.577522486759,
1352.1666371128777,
1046.1229162217542,
649.0649448069928,
504.65583665702366,
1031.3104480128327,
549.6397417010345,
633.5973201135155,
1070.0135624949132,
292.2387653636322,
1329.4427377795914,
513.541072834992,
999.4987165076434,
627.668869544527,
1136.2151975327579,
988.5192251552589,
680.0098849438172,
398.97345237739864,
1172.7713398853525,
1515.3702356119556,
329.0067208294736,
1970.0568909072203,
467.23007881967203,
1412.8879621122012,
1029.3347920283638,
626.5208757167871,

1958.1678426328344,
1169.5811559061121,
746.3670546634457,
646.5332743005072,
1193.728024792671,
1144.2560090677528,
491.3733887909018,
404.3886902539231,
747.5013383030242,
829.9193929930362,
1269.6333863670725,
345.5713597113903,
1655.6230465397407,
345.5713597113903,
1114.433015007024,
1093.2024418964083,
861.6952648384587,
1445.1128489337907,
704.3787494256529,
354.54917801953314,
526.2071121514582,
978.5272943985783,
498.24139753319963,
470.6988542905581,
2900.560320832659,
880.0657601942823,
1547.070771448853,
1192.4625942189919,
1515.8327226386937,
1875.2272848731209,
769.9526497370072,
877.4287881973803,
1658.3526032723537,
2108.3759719574423,
873.7602045155483,
1396.523318739777,
1110.6137688590657,
2415.9136561437767,
251.6722752915024,
1121.0870287281864,
700.6960985560218,
887.3698266021344,
913.846474445452,
4398.828704611039,
1115.666844997159,
1649.1489762540027,
1108.3698090097555,

1904.7793038256825,
591.3976949603958,
775.6041717217629,
1657.4698219215372,
2370.004313418395,
1441.6763787701711,
464.8461211311008,
1605.861440597504,
1498.9619995124156,
674.4304441976757,
744.3595527162133,
1891.118992329737,
446.23273988473363,
613.9374631933922,
1732.1438513546823,
1176.8884309767636,
635.0772471965793,
1081.2537534630044,
1041.2598037091782,
1535.2084713976242,
699.1863401538566,
1036.5623887784686,
1467.0193069083439,
810.9249361238228,
2569.6549600061976,
899.5891882127423,
1015.2902982794542,
955.6186372928664,
1005.0721766559147,
998.6454757237906,
450.88721510651385,
387.45448829934816,
1598.6409233027662,
1078.7236009533794,
863.7025187447738,
1174.8203668814956,
391.4158616948721,
501.7718845676359,
3185.8073307625177,
991.4608238255976,
549.2684217191174,
1036.8577059884183,
941.714160584745,
976.5160971410572,
1413.90026304572,
920.8229986794697,
606.4777526963071,

2427.3388724189977,
861.6952648384587,
1707.7688400596762,
902.5211636617445,
2107.8393308330174,
476.6109407726334,
1820.9903459002664,
638.5456245914357,
1227.6170770962567,
542.0974600882571,
1188.7077320688238,
2504.1615368130983,
1386.3855258989731,
1110.6137688590657,
457.21029207349346,
760.6029467636561,
685.9612148210189,
655.8123452373577,
1058.4381963446697,
1396.3117841018682,
493.070263727982,
1008.6717442372124,
1020.4929615602309,
1707.7688400596762,
1743.899833154992,
792.2293682176563,
1998.0885884068953,
720.9153727647348,
687.5131894239116,
973.3838288523189,
804.3723475447578,
495.3525273310765,
885.815630207741,
812.5875211145687,
459.1805653877647,
957.1505121291074,
694.3265830067251,
1312.1513594762946,
879.2150776240643,
1158.1013502502906,
1069.1565992740477,
1278.1198751621396,
691.2573608306977,
241.87620851386617,
1185.2363665375783,
523.5970724884941,
1229.64853233459,

925.8847606790641,
1241.9026264948147,
905.8770623459701,
2796.0103850265136,
507.9120702429588,
1648.1582991173136,
634.0436225845501,
1891.118992329737,
658.5988763812666,
586.2181616368493,
1217.1330023738929,
333.3016607440553,
1757.205760761824,
1028.061127689809,
890.451521724569,
467.5177550179686,
1257.061679726819,
2289.0302424618367,
1919.1367188279623,
699.198378988243,
281.1570582814995,
866.1027755347576,
2154.2012429441766,
357.5325340034277,
1060.695271106541,
347.7323191805249,
1375.0249468558022,
634.5792422775463,
1147.2575328446858,
645.751259835164,
629.993961156085,
289.0154735280634,
588.0906408327236,
1055.9572717384356,
1520.9500797412359,
2678.0368217660252,
1358.0348151510693,
1112.5271282556978,
1212.4573465738933,
470.83895371932834,
1056.240023488852,
1227.923797581021,
1033.6848020754526,
1300.1980769200052,
325.44722500802686,
1040.46986313055,
327.3660221685515,

787.5777718105636,
716.8717957778291,
1875.2272848731209,
1498.9619995124156,
1299.9284526080658,
246.62225694722767,
1810.9577144882176,
398.1144688416095,
791.8884452468221,
2108.3759719574423,
1204.0839976817874,
1281.692873241522,
1876.345773853028,
1549.082017152361,
989.0853076552326,
333.3016607440553,
968.9411727273939,
604.4741448186886,
1444.9764660939884,
454.48671907881584,
1057.2522015999045,
670.454272339137,
815.0276557525101,
584.8393757393145,
431.32992974065803,
1012.7423576239494,
923.2114686906604,
909.4952185521971,
1571.3092936428077,
1335.225299998239,
418.3860655751012,
1908.6201858448815,
474.03596511993817,
889.2850298176719,
472.10144843645946,
505.71142907166956,
323.51983551144986,
300.76938969320855,
744.3595527162133,
1431.9739930853577,
431.61908860804084,
1327.5454305630683,
1098.251290659492,
1857.003857022438,
1956.5628091309313,
645.675486345364,
1354.8799695468545,

1965.1511486383306,
2080.224189087948,
1185.2363665375783,
1855.860431335929,
705.9751700260002,
1362.6860797331997,
801.2509584259823,
576.3300410795845,
1044.2186974620238,
1104.0490188323577,
2244.2950726915224,
2336.6559954426384,
350.2018915495339,
1891.118992329737,
941.262399950194,
905.8770623459701,
1958.1678426328344,
1354.702988025751,
1912.0379877083312,
1725.554120049198,
462.4918323911604,
263.2258346497289,
1898.54412436639,
1754.3628346386224,
499.9650670891465,
921.363375970034,
1018.9110311352545,
578.3137585410021,
675.0423715608554,
1566.0267442526128,
614.5167940790396,
1239.5613576757619,
1950.0320058273321,
428.0712395672395,
1191.318149662369,
606.6710333996674,
2642.140105227776,
1135.9496719525728,
1035.6257594834283,
1042.7487515643782,
1717.209277669252,
358.0077399927364,
291.0999682573211,
1187.6012606243169,
609.5751008744897,
2435.3146430701377,
1696.5919853419707,

1288.631917699457,
336.4160304187888,
2178.786982559669,
2254.3715915210423,
582.0237362775239,
549.5893635250253,
686.5598711785669,
1543.7215325543227,
411.51682678248494,
1175.1317523522687,
1113.2753111015468,
510.52503058362896,
1337.4609534850858,
1050.679325061138,
2283.4141318149586,
626.7236265976244,
1869.9234706060538,
1967.664694240409,
1204.939084551458,
932.9672414340619,
413.7933528070396,
1464.778630897011,
559.2795302681642,
1699.6829398586824,
608.0205737230891,
533.7567836757175,
465.4892226449894,
753.8593323637538,
1010.3714889499391,
1324.1051685225991,
327.5706286711989,
490.70127506029974,
786.6921053017463,
1656.007234000088,
1840.2791184158616,
1766.8598728413224,
983.242545176882,
1287.331507114793,
1451.6476013177924,
1363.1710673843418,
693.0531220815641,
1937.0769366632553,
1157.8144721123986,
686.9127934162292,
1443.408431626942,
1188.1312527285227,
537.8043902643426,

881.3584482537957,
277.989158633539,
448.0785827494966,
1177.8695117327647,
1883.7368859057653,
1632.5128874073748,
1926.3499704708743,
1923.3635370531688,
940.8803182988421,
2506.178212479923,
1286.4302462308476,
550.2147056753814,
1914.1973029833528,
1155.217250649656,
1004.468761286547,
1498.5866175004146,
973.4707279122276,
1023.1540291029613,
311.73436725859887,
660.5468801337717,
1357.734226543324,
314.5873209896762,
295.9572723267685,
1126.769745630481,
1140.4146282599074,
664.645415646544,
1560.8841617981445,
1038.7584812633208,
1083.4479094863195,
706.9810341872646,
731.7203542515116,
1221.1209466661871,
1440.4044495204464,
237.8355432525894,
629.4836558404912,
2137.958924245749,
931.4385725219784,
1855.860431335929,
1441.0433452710156,
1452.6979389526693,
1089.2478274686064,
616.2577850827462,
854.4683484763104,
1111.2381026902299,
755.7336000162555,
489.3602832414994,
671.0197000404106,

1092.174270142468,
2436.74714335704,
942.3284850920983,
1456.5572997022612,
309.58159816680325,
1277.3721504028538,
1190.249751066054,
816.962086413135,
441.83952090774295,
1433.1823573383992,
1075.46040534034,
1482.3792181898218,
713.0481591257159,
1321.1473384620424,
779.518037680624,
379.85595452989645,
579.3101557344876,
834.34563209182,
497.36676955015304,
475.9359439717281,
1320.6336544332871,
507.8339242715184,
1092.174270142468,
2569.6549600061976,
533.9727751481109,
1120.9812399149919,
322.0091019824211,
289.67218832676826,
812.5875211145687,
555.3067991715524,
578.6933102479965,
386.1040147405051,
546.5071949328983,
2008.5060813616062,
1967.0331794285983,
903.6282586166845,
1436.0328202690955,
843.4391215902345,
659.9943387582709,
351.2045795903974,
845.4606943772486,
1493.6312219945355,
466.44677473250533,
589.645463096489,
1016.7292806511595,
805.6830396959077,
320.47976615080654,

1931.7446414188464,
2916.228810472802,
2504.1615368130983,
780.3349423007694,
329.92570230483665,
686.7325616533079,
727.6625587446458,
1529.5761106938144,
310.2433139737481,
2357.250252855667,
1813.6975018434946,
1624.8249791293258,
618.6002605819264,
1399.6612506532847,
816.976146206924,
1021.2315423264907,
539.2680198718854,
1482.1684107895899,
464.71227588177925,
467.99319624682573,
1862.7883875743119,
1232.95431602385,
1656.0772188766682,
1805.4256943040268,
1204.0839976817874,
569.9891512935479,
319.5118617517487,
2228.842078872771,
456.1139052006885,
2159.9506904710233,
1185.552154306059,
613.0721665168535,
1174.0910275861568,
326.5891529760772,
3126.646788768731,
1102.4236233633842,
432.32168844243915,
405.8169864750892,
633.2407304184534,
1366.6704845540833,
559.5556897838123,
1355.4104330015884,
1886.6825892069116,
393.4739865623015,
912.3343751376098,
1204.939084551458,
701.7076334181979,

730.4277699689331,
750.221725730808,
817.1873544031471,
244.14705173827673,
508.5225868958546,
743.6443135443318,
1229.64853233459,
1868.0018072480002,
1493.0718465981286,
814.8218292400038,
523.2169742663683,
425.2868210773383,
854.4304800195911,
1614.7276155691618,
250.7215002629189,
1721.7286103051713,
2356.8420640801646,
1108.3485238744568,
574.8521038623268,
2150.381778636337,
998.354358761599,
454.0431526026928,
286.96111187330666,
1019.4428604535336,
2332.9655166386706,
1413.6294256541632,
1622.483047603988,
2372.143130872212,
779.518037680624,
1500.6101456900226,
1214.4148856873232,
374.8459454965957,
942.3284850920983,
859.2957328394078,
1221.1209466661871,
1427.1933830438938,
3112.6675467331847,
1114.8811593777195,
366.1404631718196,
709.2407959008378,
541.8016096849709,
1281.692873241522,
484.2955041682813,
838.0261016024026,
816.3369100095118,
1112.5271282556978,
417.99941933162677,

1697.458081651752,
880.0657601942823,
1306.545483987454,
377.28328983299855,
716.3190667737701,
1548.715290360932,
1141.340029326224,
757.705881086105,
1840.2578717457106,
1431.8186959275652,
1050.679325061138,
413.22366753497494,
1909.8522856502366,
629.0850301082376,
1676.7949001184243,
2412.249970189544,
674.8636173408486,
536.8685873643064,
591.2981030414443,
325.3806319583877,
338.91791410460326,
823.890109636202,
1899.3978211735896,
2064.6502880963776,
1054.5588486186043,
318.8349823371805,
1810.510278806821,
429.5280767124288,
1867.112657096808,
707.5118853361989,
895.0769916863034,
471.31589112873064,
492.18358538084715,
633.5877346648643,
394.3262477535319,
893.2682297052072,
863.7025187447738,
1843.6099148467242,
1218.0702333115707,
602.351909692025,
1108.3485238744568,
934.6062114343748,
701.8939625500402,
307.59050504439165,
1421.2397914444552,
1039.3207577178664,
1092.174270142468,

336.2776447905818,
291.2952154595297,
1185.6096706120945,
1832.2646105856481,
939.5540585663907,
1509.743737559331,
3400.850835980718,
1281.692873241522,
497.4508494118447,
819.7253243967283,
640.6041340424896,
622.6992641757104,
456.7760526222201,
2146.060118204473,
648.584774048108,
541.2770064667038,
766.7992631766368,
1071.24355540759,
740.8898275730479,
1937.4857346302226,
781.1756061802066,
1853.6694678453514,
454.0431526026928,
1148.19948086819,
641.4312579286402,
1473.39701369637,
1044.97490257999,
2781.607410147614,
236.8093382153303,
1825.7562395697266,
708.5981572730989,
1012.3241503523649,
2534.064385022313,
490.50995586755636,
1300.1980769200052,
544.1509735667182,
950.9534993728251,
734.8661447202413,
1459.2682252057623,
955.6987907809304,
1892.7558709297728,
2660.5522015005763,
650.8554122125245,
1108.3485238744568,
1790.798230772991,
1368.003003396792,
1173.9018794723193,

715.7569168899989,
767.9339329342736,
448.5696333612637,
1466.390238581356,
1577.3013088714283,
318.85364371500884,
1278.3456473509555,
1698.0798985541305,
901.5170108263793,
1092.1488402882267,
1278.3922457518931,
1011.948726448726,
2323.618520869477,
623.56812187812,
2395.1795889787927,
746.6308180447327,
3473.586948822207,
2222.1976528580194,
1838.7933212025234,
397.13489848177215,
2085.3318413066763,
1851.9078566657627,
1098.4265768734506,
786.9614589455504,
2357.0360616028647,
1025.7935197040706,
1279.1748561461459,
231.9605724764767,
1862.7883875743119,
978.3937143009085,
1156.0524041464203,
1781.4697491243671,
315.0243723013468,
717.9481671533568,
1413.90026304572,
752.6661837226254,
2131.201534212071,
1018.0972070450192,
1478.7557461456095,
1497.4633250797403,
553.5750641151724,
1380.7626650400769,
680.0098849438172,
570.5133543236495,
1951.1619816399923,
632.7417785197571,
1655.6898430993647,

2095.4405667490314,
1497.149520586426,
894.6079444756461,
820.2766154258262,
1702.3945664481325,
2060.345327409709,
331.6268119865543,
2424.979519570181,
823.9419303275116,
463.2205809133463,
1115.7584257368073,
1989.7962204910446,
1047.8316305383662,
1201.7299795055217,
460.4968317725008,
1193.728024792671,
580.5716549530688,
333.1308895873255,
786.7854297804881,
3564.6584248524046,
2157.6026545628974,
527.9170700836997,
1225.4287979073558,
1435.6485624189297,
2155.4902608662333,
1891.118992329737,
526.2071121514582,
315.58424074470014,
744.8606978578945,
1865.416615069174,
3086.8386626441975,
783.2768819970058,
703.3249490725286,
1578.1480307631757,
742.5209990375346,
441.9419308755802,
1154.1046790947985,
1815.6094341913922,
1578.1480307631757,
382.2119198099955,
1634.5567903388744,
884.1115814126076,
1207.303424983611,
932.6097075728985,
973.5104985075441,
335.12156565854474,
1952.0141514771742,

771.0390789626057,
448.5782648230335,
797.8437700322421,
1216.7195389724295,
1176.5325288759632,
889.6673366030252,
2090.0108144733917,
867.3554512245862,
1256.4043675572361,
1721.7286103051713,
1033.724405375887,
1509.3094958076101,
773.6696363509726,
460.65204169223915,
1200.2002115958385,
667.8429229566838,
1643.7970367561027,
1306.5586392612456,
325.44722500802686,
1234.607866126571,
753.3992657629778,
913.6374576706403,
1327.071644260557,
801.6168974049006,
322.3275802489051,
834.3378016749651,
365.2728695818902,
1327.071644260557,
456.1139052006885,
1306.4060535557905,
317.3090884813926,
445.63374692210834,
1061.4868984037682,
1648.4380742779447,
1563.894026238578,
1509.743737559331,
2190.3635290189745,
1471.2414309532016,
972.6354454573615,
889.0638323042766,
578.6933102479965,
943.9273698063399,
1763.5040755801363,
2021.213761357109,
1651.7394685535903,
1460.8605600087985,
1421.1503090594565,

1858.6038122297325,
1668.2056114444995,
341.33529689779095,
888.4158688346671,
1818.9931796805765,
1078.7236009533794,
1439.9995391587545,
773.7539714368942,
1027.276520768967,
472.59688527537986,
944.1588059909624,
1062.1265196924994,
1709.085902917057,
1350.2701881439762,
1098.4265768734506,
678.5691987844749,
1024.7339106586714,
1382.832395890294,
1576.74121788013,
1612.6524311627397,
816.1834117826409,
1622.483047603988,
1086.5056588637153,
663.1467350473007,
959.5831319821574,
350.2018915495339,
1336.2995115141623,
302.8307996241544,
1965.971241724081,
1006.2096486712511,
1598.9786493251672,
537.8043902643426,
1214.4148856873232,
1518.8353490510979,
2633.2146599063412,
1336.8908047376738,
451.5668360610901,
658.457804764397,
811.1515959811667,
1513.5411863120946,
1213.5245080736868,
1542.9700248498884,
691.2573608306977,
568.0974967280614,
327.2485407544689,
1934.1756144455358,
1647.8650711557186,

1413.6294256541632,
888.3610109092716,
1318.660362483005,
2763.4059798121325,
1617.3442018670132,
612.8059593648555,
1374.1291568884396,
393.4739865623015,
1413.6294256541632,
1543.7215325543227,
972.7095151569478,
1764.5908174608867,
709.8056784063197,
431.62811990401667,
802.5386557985954,
572.3144609383454,
2992.3706998000757,
712.1323811554892,
1961.3686916223862,
1793.2455820759487,
2919.0266436071374,
1122.9304164115376,
1824.0576500470406,
986.1091730855399,
854.980565073946,
2093.0459051001653,
675.0259817165587,
1849.5605071400398,
1307.6710136567579,
1150.6597317330425,
544.5532731560471,
657.2157477271912,
870.0519364026371,
664.8907254001914,
849.0688532808972,
1108.3485238744568,
695.1646979877888,
1026.4743491435474,
1560.9786143006545,
521.1122034576968,
368.5262967739381,
1016.7292806511595,
941.0311895019528,
1236.7174616496777,
1611.4214638880146,
357.3227783144322,
1649.9183088979275,

```
1808.7813284184233,
…]
```

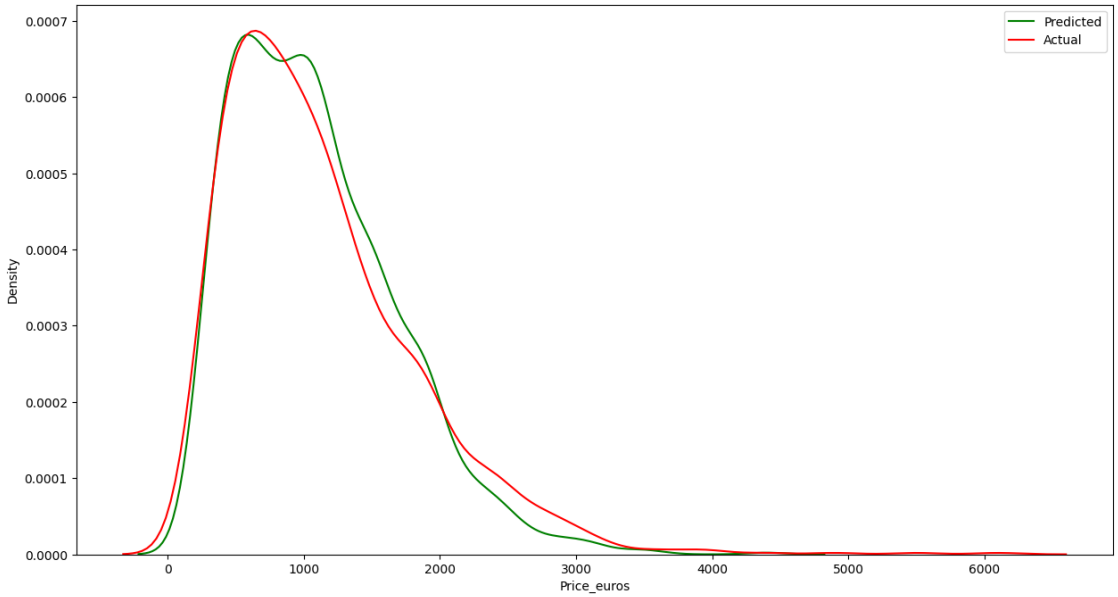
[206]:
```
X['predicted'] = np.array(predicted_price_eruro)
X.head()
```

[206]:

	Company	TypeName	Ram_GB	OpSys	Weight_KG	TouchScreen	IPS	PPI	\
0	Apple	Ultrabook	8	MaC	1.37	0	1	226.992481	
1	Apple	Ultrabook	8	MaC	1.34	0	0	127.669173	
2	HP	Notebook	8	Others	1.86	0	0	141.217949	
3	Apple	Ultrabook	16	MaC	1.83	0	1	220.519481	
4	Apple	Ultrabook	8	MaC	1.37	0	1	226.992481	

	Cpu_name	HDD	SSD	Gpu_brand	predicted
0	Intel Core i5	0	128	Intel	1324.588495
1	Intel Core i5	0	0	Intel	997.575980
2	Intel Core i5	0	256	Intel	678.343234
3	Intel Core i7	0	512	AMD	2506.547006
4	Intel Core i5	0	256	Intel	1536.713227

[207]:
```
# Let's compare the predicted price and Actual Price

plt.figure(figsize = (15,8))
sns.distplot(X['predicted'],hist = False, label = 'Predicted', color = 'green')
sns.distplot(df['Price_euros'], hist = False, label = 'Actual', color = 'red')
plt.legend()
plt.show()
```



[208]:
```
# The pickle module allows you to serialize and save Python objects, including␣
 ↪machine learning models, to a file.

import pickle

file = open('laptopPricePredictor.pkl','wb')
pickle.dump(RFRegV3,file)
file.close()
```