Hands-on Practice Lab: Data Wrangling

Estimated time needed: **30** minutes

In this lab, you will use the skills acquired in the module and address the issues of handling missing data, correct the data type of the dataframe attribute and execute the processes of data standardization and data normalization on specific attributes of the dataset.

Objectives

After completing this lab you will be able to:

- Handle missing data in different ways
- Correct the data type of different data values as per requirement
- Standardize and normalize the appropriate data attributes
- Visualize the data as grouped bar graph using Binning
- Cnverting a categorical data into numerical indicator variables

Setup

For this lab, we will be using the following libraries:

- `skillsnetwork` to download the dataset
- `pandas` for managing the data.
- `numpy` for mathematical operations.
- `matplotlib` for additional plotting tools.

Importing Required Libraries

We recommend you import all required libraries in one place (here):

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

Download the updated dataset by running the cell below.

The functions below will download the dataset into your browser:

In [2]:
```
from pyodide.http import pyfetch

async def download(url, filename):
    response = await pyfetch(url)
    if response.status == 200:
        with open(filename, "wb") as f:
            f.write(await response.bytes())
```

In [3]:
```
file_path= "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DA0101EN-Coursera/laptop_pricing_dataset_mod1.cs
```

To obtain the dataset, utilize the download() function as defined above:

In [4]:
```
await download(file_path, "laptops.csv")
file_name="laptops.csv"
```

First we load data into a `pandas.DataFrame` :

In [5]:
```
df = pd.read_csv(file_name, header=0)
```

> Note: This version of the lab is working on JupyterLite, which requires the dataset to be downloaded to the interface.While working on the downloaded version of this notebook on their local machines(Jupyter Anaconda), the learners can simply **skip the steps above,** and simply use the URL directly in the `pandas.read_csv()` function. You can uncomment and run the statements in the cell below.

In [ ]:
```
#filepath = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DA0101EN-Coursera/laptop_pricing_dataset_mod1.c
#df = pd.read_csv(filepath, header=None)
```

Verify loading by displaying the dataframe summary using `dataframe.info()`

In [6]:
```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 238 entries, 0 to 237
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Unnamed: 0      238 non-null    int64
 1   Manufacturer    238 non-null    object
 2   Category        238 non-null    int64
 3   Screen          238 non-null    object
 4   GPU             238 non-null    int64
 5   OS              238 non-null    int64
 6   CPU_core        238 non-null    int64
 7   Screen_Size_cm  234 non-null    float64
 8   CPU_frequency   238 non-null    float64
 9   RAM_GB          238 non-null    int64
 10  Storage_GB_SSD  238 non-null    int64
 11  Weight_kg       233 non-null    float64
 12  Price           238 non-null    int64
dtypes: float64(3), int64(8), object(2)
memory usage: 22.4+ KB
None
```

View the first 5 values of the updated dataframe using `dataframe.head()`

In [7]:
```
df.head()
```

Out[7]:

	Unnamed: 0	Manufacturer	Category	Screen	GPU	OS	CPU_core	Screen_Size_cm	CPU_frequency	RAM_GB	Storage_GB_SSD	Weight_kg	Price
0	0	Acer	4	IPS Panel	2	1	5	35.560	1.6	8	256	1.60	978
1	1	Dell	3	Full HD	1	1	3	39.624	2.0	4	256	2.20	634
2	2	Dell	3	Full HD	1	1	7	39.624	2.7	8	256	2.20	946
3	3	Dell	4	IPS Panel	2	1	5	33.782	1.6	8	128	1.22	1244
4	4	HP	4	Full HD	2	1	7	39.624	1.8	8	256	1.91	837

Note that we can update the `Screen_Size_cm` column such that all values are rounded to nearest 2 decimal places by using `numpy.round()`

In [8]:
```
df[['Screen_Size_cm']] = np.round(df[['Screen_Size_cm']],2)
df.head()
```

Out[8]:

	Unnamed: 0	Manufacturer	Category	Screen	GPU	OS	CPU_core	Screen_Size_cm	CPU_frequency	RAM_GB	Storage_GB_SSD	Weight_kg	Price
0	0	Acer	4	IPS Panel	2	1	5	35.56	1.6	8	256	1.60	978
1	1	Dell	3	Full HD	1	1	3	39.62	2.0	4	256	2.20	634
2	2	Dell	3	Full HD	1	1	7	39.62	2.7	8	256	2.20	946
3	3	Dell	4	IPS Panel	2	1	5	33.78	1.6	8	128	1.22	1244
4	4	HP	4	Full HD	2	1	7	39.62	1.8	8	256	1.91	837

Task - 1

Evaluate the dataset for missing data

Missing data was last converted from '?' to numpy.NaN. Pandas uses NaN and Null values interchangeably. This means, you can just identify the entries having Null values. Write a code that identifies which columns have missing data.

In [10]:
```
# Write your code below and press Shift+Enter to execute
missing_data = df.isnull()
for column in missing_data.columns.values.tolist():
    print(missing_data[column].value_counts())
    print("")
```

```
Unnamed: 0
False    238
Name: count, dtype: int64

Manufacturer
False    238
Name: count, dtype: int64

Category
False    238
Name: count, dtype: int64

Screen
False    238
Name: count, dtype: int64

GPU
False    238
Name: count, dtype: int64

OS
False    238
Name: count, dtype: int64

CPU_core
False    238
Name: count, dtype: int64

Screen_Size_cm
False    234
True       4
Name: count, dtype: int64

CPU_frequency
False    238
Name: count, dtype: int64

RAM_GB
False    238
Name: count, dtype: int64

Storage_GB_SSD
False    238
Name: count, dtype: int64

Weight_kg
False    233
True       5
Name: count, dtype: int64

Price
False    238
Name: count, dtype: int64
```

▸ Click here for the solution

Task - 2

Replace with mean

Missing values in attributes that have continuous data are best replaced using Mean value. We note that values in "Weight_kg" attribute are continuous in nature, and some values are missing. Therefore, write a code to replace the missing values of weight with the average value of the attribute.

In [ ]:
```
# Write your code below and press Shift+Enter to execute
df["Weight_kg"].replace(np.nan,df["Weight_kg"].astype('float').mean(axis=0),inplace=True)
```

▸ Click here for the solution

Replace with the most frequent value

Missing values in attributes that have categorical data are best replaced using the most frequent value. We note that values in "Screen_Size_cm" attribute are categorical in nature, and some values are missing. Therefore, write a code to replace the missing values of Screen Size with the most frequent value of the attribute.

In [ ]:
```
# Write your code below and press Shift+Enter to execute
df["Screen_Size_cm"].replace(np.nan,df["Screen_Size_cm"].value_counts().idxmax(),inplace=True)
```

▸ Click here for the solution

Task - 3

Fixing the data types

Both "Weight_kg" and "Screen_Size_cm" are seen to have the data type "Object", while both of them should be having a data type of "float". Write a code to fix the data type of these two columns.

In [13]:
```
# Write your code below and press Shift+Enter to execute
df[["Weight_kg","Screen_Size_cm"]] = df[["Weight_kg","Screen_Size_cm"]].astype('float')
```

▸ Click here for Solution

Task - 4

Data Standardization

The value of Screen_size usually has a standard unit of inches. Similarly, weight of the laptop is needed to be in pounds. Use the below mentioned units of conversion and write a code to modify the columns of the dataframe accordingly. Update their names as well.

```
{math}
1 inch = 2.54 cm
1 kg   = 2.205 pounds
```

In [15]:
```
# Write your code below and press Shift+Enter to execute
df["Weight_kg"] = 2.205 * df["Weight_kg"]
df["Screen_Size_cm"] = df["Screen_Size_cm"] / 2.54
df.rename(columns={"Weight_kg":"Weight_pound","Screen_Size_cm":"Screen_Size_inch"},inplace=True)
df[["Weight_pound","Screen_Size_inch"]].head()
```

Out[15]:

	Weight_pound	Screen_Size_inch
0	3.52800	14.000000
1	4.85100	15.598425
2	4.85100	15.598425
3	2.69010	13.299213
4	4.21155	15.598425

▸ Click here for Solution

Data Normalization

Often it is required to normalize a continuous data attribute. Write a code to normalize the "CPU_frequency" attribute with respect to the maximum value available in the dataset.

In [17]:
```
# Write your code below and press Shift+Enter to execute
df["CPU_frequency"] = df["CPU_frequency"] / df["CPU_frequency"].max()
```

▸ Click here for Solution

Task - 5

Binning

Binning is a process of creating a categorical attribute which splits the values of a continuous data into a specified number of groups. In this case, write a code to create 3 bins for the attribute "Price". These bins would be named "Low", "Medium" and "High". The new attribute will be named "Price-binned".

In [18]:
```
# Write your code below and press Shift+Enter to execute
bins = np.linspace(min(df["Price"]),max(df["Price"]),4)
group_names = ['Low', 'Medium', 'High']
df['Price-binned'] = pd.cut(df['Price'], bins, labels=group_names, include_lowest=True )
df[['Price','Price-binned']].head(5)
df['Price-binned'].value_counts()
```

Out[18]:
```
Price-binned
Low       160
Medium     72
High        6
Name: count, dtype: int64
```

▸ Click here for Solution

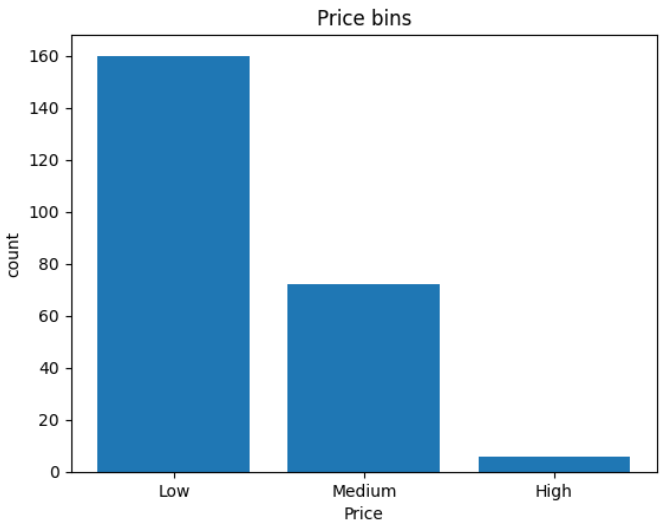
Also, plot the bar graph of these bins.

In [20]:
```
# Write your code below and press Shift+Enter to execute
%matplotlib inline

import matplotlib as plt
from matplotlib import pyplot
pyplot.bar(group_names, df["Price-binned"].value_counts())

plt.pyplot.xlabel("Price")
plt.pyplot.ylabel("count")
plt.pyplot.title("Price bins")
```

Out[20]:
```
Text(0.5, 1.0, 'Price bins')
```



▸ Click here for Solution

Task - 6

Indicator variables

Convert the "Screen" attribute of the dataset into 2 indicator variables, "Screen-IPS_panel" and "Screen-Full_HD". Then drop the "Screen" attribute from the dataset.

In [26]:
```
# Write your code below and press Shift+Enter to execute
df_dummies = pd.get_dummies(df["Screen"])
df_dummies.rename(columns={"Full HD":"Screen-Full_HD","IPS Panel":"Screen-IPS_panel"}, inplace=True)
df = pd.concat([df,df_dummies],axis=1)
df.drop("Screen",axis=1,inplace=True)
```

▸ Click here for Solution

This version of the dataset, now finalized, is the one you'll be using in all subsequent modules.

Print the content of dataframe.head() to verify the changes that were made to the dataset.

In [27]:
```
print(df.head())
```

```
   Unnamed: 0 Manufacturer  Category  GPU   OS  CPU_core  Screen_Size_inch  \
0         0.0         Acer       4.0  2.0  1.0       5.0         14.000000
1         1.0         Dell       3.0  1.0  1.0       3.0         15.598425
2         2.0         Dell       3.0  1.0  1.0       7.0         15.598425
3         3.0         Dell       4.0  2.0  1.0       5.0         13.299213
4         4.0           HP       4.0  2.0  1.0       7.0         15.598425

   CPU_frequency  RAM_GB  Storage_GB_SSD  Weight_pound   Price Price-binned  \
0       0.551724     8.0           256.0       3.52800   978.0          Low
1       0.689655     4.0           256.0       4.85100   634.0          Low
2       0.931034     8.0           256.0       4.85100   946.0          Low
3       0.551724     8.0           128.0       2.69010  1244.0          Low
4       0.620690     8.0           256.0       4.21155   837.0          Low

   Screen-Full_HD  Screen-IPS_panel  Screen-Full_HD  Screen-IPS_panel
0             NaN               NaN           False              True
1             NaN               NaN            True             False
2             NaN               NaN            True             False
3             NaN               NaN           False              True
4             NaN               NaN            True             False
```

Congratulations! You have completed the lab

Authors

Abhishek Gagneja

Vicky Kuo