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):

```
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:

```
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())

file_path= "https://cf-courses-data.s3.us.cloud-object-
storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DA0101EN-Coursera/
laptop_pricing_dataset_mod1.csv"
```

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

```
await download(file_path, "laptops.csv")
file_name="laptops.csv"
```

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

```
df = pd.read_csv(file_name, header=0, on_bad_lines='skip')
```

> 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.

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

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

```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 239 entries, 0 to 238
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Unnamed: 0      239 non-null    int64  
 1   Manufacturer    239 non-null    object 
 2   Category        239 non-null    int64  
 3   Screen          239 non-null    object 
 4   GPU             239 non-null    int64  
 5   OS              239 non-null    object 
 6   CPU_core        239 non-null    int64  
 7   Screen_Size_cm  235 non-null    float64
 8   CPU_frequency   238 non-null    float64
 9   RAM_GB          238 non-null    float64
 10  Storage_GB_SSD  238 non-null    float64
```

```
 11  Weight_kg       233 non-null    float64
 12  Price           238 non-null    float64
dtypes: float64(6), int64(4), object(3)
memory usage: 21.5+ KB
None
```

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

```
df.head()
```

```
   Unnamed: 0 Manufacturer  Category     Screen  GPU OS  CPU_core  \
0           0         Acer         4  IPS Panel    2  1         5
1           1         Dell         3    Full HD    1  1         3
2           2         Dell         3    Full HD    1  1         7
3           3         Dell         4  IPS Panel    2  1         5
4           4           HP         4    Full HD    2  1         7

   Screen_Size_cm  CPU_frequency  RAM_GB  Storage_GB_SSD  Weight_kg
Price
0          35.560            1.6     8.0           256.0       1.60
978.0
1          39.624            2.0     4.0           256.0       2.20
634.0
2          39.624            2.7     8.0           256.0       2.20
946.0
3          33.782            1.6     8.0           128.0       1.22
1244.0
4          39.624            1.8     8.0           256.0       1.91
837.0
```

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()`

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

```
   Unnamed: 0 Manufacturer  Category     Screen  GPU OS  CPU_core  \
0           0         Acer         4  IPS Panel    2  1         5
1           1         Dell         3    Full HD    1  1         3
2           2         Dell         3    Full HD    1  1         7
3           3         Dell         4  IPS Panel    2  1         5
4           4           HP         4    Full HD    2  1         7

   Screen_Size_cm  CPU_frequency  RAM_GB  Storage_GB_SSD  Weight_kg
Price
0           35.56            1.6     8.0           256.0       1.60
978.0
1           39.62            2.0     4.0           256.0       2.20
634.0
```

```
2           39.62            2.7     8.0           256.0       2.20
946.0
3           33.78            1.6     8.0           128.0       1.22
1244.0
4           39.62            1.8     8.0           256.0       1.91
837.0
```

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.

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

```
   Unnamed: 0  Manufacturer  Category  Screen    GPU     OS  CPU_core
\
0       False         False     False   False  False  False     False

1       False         False     False   False  False  False     False

2       False         False     False   False  False  False     False

3       False         False     False   False  False  False     False

4       False         False     False   False  False  False     False


   Screen_Size_cm  CPU_frequency  RAM_GB  Storage_GB_SSD  Weight_kg
Price
0           False          False   False           False      False
False
1           False          False   False           False      False
False
2           False          False   False           False      False
False
3           False          False   False           False      False
False
4           False          False   False           False      False
False
```

```
Unnamed: 0
False    239
Name: Unnamed: 0, dtype: int64

Manufacturer
False    239
Name: Manufacturer, dtype: int64

Category
False    239
Name: Category, dtype: int64

Screen
False    239
Name: Screen, dtype: int64

GPU
False    239
Name: GPU, dtype: int64

OS
False    239
Name: OS, dtype: int64

CPU_core
False    239
Name: CPU_core, dtype: int64

Screen_Size_cm
False    235
True       4
Name: Screen_Size_cm, dtype: int64

CPU_frequency
False    238
True       1
Name: CPU_frequency, dtype: int64

RAM_GB
False    238
True       1
Name: RAM_GB, dtype: int64

Storage_GB_SSD
False    238
True       1
Name: Storage_GB_SSD, dtype: int64

Weight_kg
False    233
```

```
True       6
Name: Weight_kg, dtype: int64

Price
False    238
True       1
Name: Price, dtype: int64
```

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.

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

# astype() function converts the values to the desired data type.
# axis=0 indicates that the mean value is to calculated across all
column elements in a row.
```

```
Average of Weight_kg: 1.8622317596566522
```

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.

```
# Write your code below and press Shift+Enter to execute
# replacing missing data with mode
common_screen_size = df['Screen_Size_cm'].value_counts().idxmax()
df["Screen_Size_cm"].replace(np.nan, common_screen_size, inplace=True)
```

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.

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

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.

```
1 inch = 2.54 cm
1 kg   = 2.205 pounds

# Write your code below and press Shift+Enter to execute
# Data standardization: convert weight from kg to pounds
df["Weight_kg"] = df["Weight_kg"]*2.205
df.rename(columns={'Weight_kg':'Weight_pounds'}, inplace=True)

# Data standardization: convert screen size from cm to inch
df["Screen_Size_cm"] = df["Screen_Size_cm"]/2.54
df.rename(columns={'Screen_Size_cm':'Screen_Size_inch'}, inplace=True)
```

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.

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

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".

```
# 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)
```

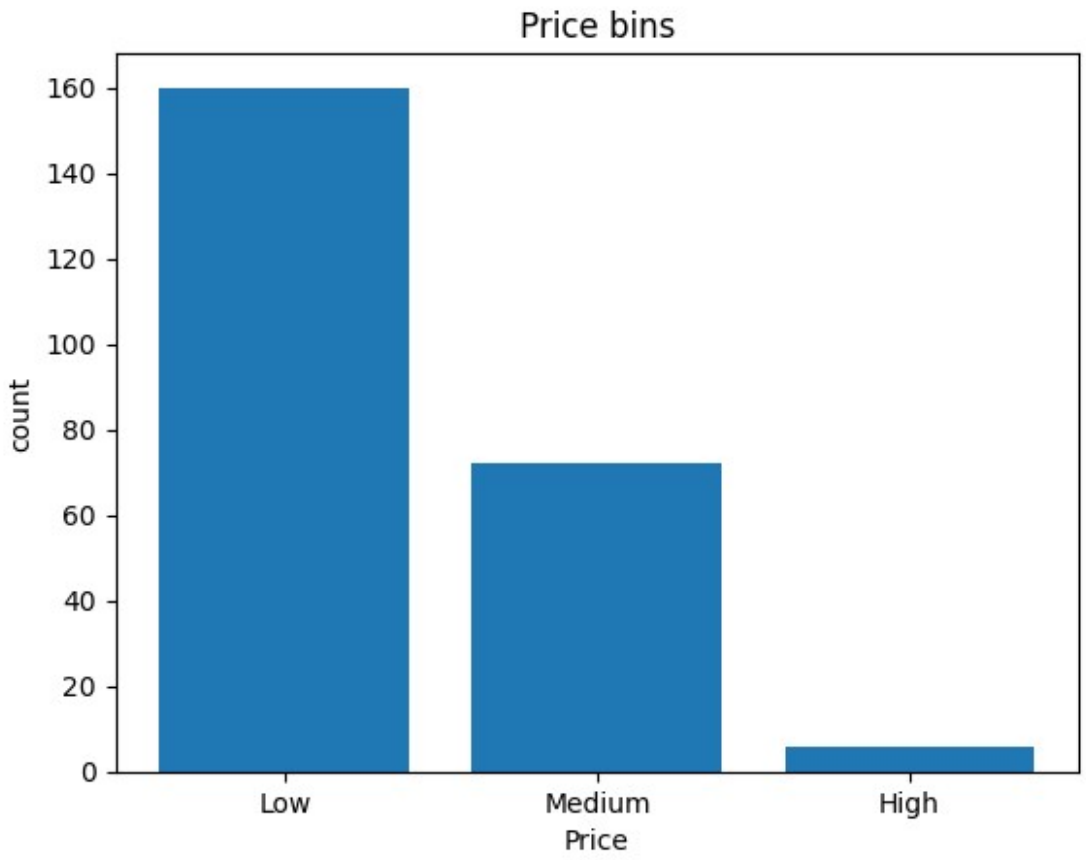
Also, plot the bar graph of these bins.

```
# Write your code below and press Shift+Enter to execute
df["Price-binned"].value_counts()

%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")
```

```
Text(0.5, 1.0, 'Price bins')
```



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.

```
# Write your code below and press Shift+Enter to execute
#Indicator Variable: Screen
dummy_variable_1 = pd.get_dummies(df["Screen"])
dummy_variable_1.rename(columns={'IPS Panel':'Screen-IPS_panel', 'Full
HD':'Screen-Full_HD'}, inplace=True)
df = pd.concat([df, dummy_variable_1], axis=1)

# drop original column "Screen" from "df"
df.drop("Screen", axis = 1, inplace=True)
```

```
---------------------------------------------------------------------
-----
KeyError                                  Traceback (most recent call
last)
```

```
File /lib/python3.11/site-packages/pandas/core/indexes/base.py:3802,
in Index.get_loc(self, key, method, tolerance)
   3801 try:
-> 3802     return self._engine.get_loc(casted_key)
   3803 except KeyError as err:

File /lib/python3.11/site-packages/pandas/_libs/index.pyx:138, in
pandas._libs.index.IndexEngine.get_loc()

File /lib/python3.11/site-packages/pandas/_libs/index.pyx:162, in
pandas._libs.index.IndexEngine.get_loc()

File /lib/python3.11/site-packages/pandas/_libs/index.pyx:203, in
pandas._libs.index.IndexEngine._get_loc_duplicates()

File /lib/python3.11/site-packages/pandas/_libs/index.pyx:211, in
pandas._libs.index.IndexEngine._maybe_get_bool_indexer()

File /lib/python3.11/site-packages/pandas/_libs/index.pyx:107, in
pandas._libs.index._unpack_bool_indexer()

KeyError: 'Screen'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call
last)
Cell In[53], line 3
      1 # Write your code below and press Shift+Enter to execute
      2 #Indicator Variable: Screen
----> 3 dummy_variable_1 = pd.get_dummies(df["Screen"])
      4 dummy_variable_1.rename(columns={'IPS Panel':'Screen-
IPS_panel', 'Full HD':'Screen-Full_HD'}, inplace=True)
      5 df = pd.concat([df, dummy_variable_1], axis=1)

File /lib/python3.11/site-packages/pandas/core/frame.py:3807, in
DataFrame.__getitem__(self, key)
   3805 if self.columns.nlevels > 1:
   3806     return self._getitem_multilevel(key)
-> 3807 indexer = self.columns.get_loc(key)
   3808 if is_integer(indexer):
   3809     indexer = [indexer]

File /lib/python3.11/site-packages/pandas/core/indexes/base.py:3804,
in Index.get_loc(self, key, method, tolerance)
   3802     return self._engine.get_loc(casted_key)
   3803 except KeyError as err:
-> 3804     raise KeyError(key) from err
   3805 except TypeError:
   3806     # If we have a listlike key, _check_indexing_error will
```

```
raise
   3807     #  InvalidIndexError. Otherwise we fall through and re-
raise
   3808     #  the TypeError.
   3809     self._check_indexing_error(key)

KeyError: 'Screen'
```

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.

```
print(df.head())
```

```
   Unnamed: 0 Manufacturer  Category  GPU OS  CPU_core  Screen_Size_cm
\
0           0         Acer         4    2  1         5           35.56

1           1         Dell         3    1  1         3           39.62

2           2         Dell         3    1  1         7           39.62

3           3         Dell         4    2  1         5           33.78

4           4           HP         4    2  1         7           39.62


   CPU_frequency  RAM_GB  Storage_GB_SSD  Weight_kg   Price
Weight_pounds  \
0       0.551724     8.0           256.0       1.60   978.0
3.52800
1       0.689655     4.0           256.0       2.20   634.0
4.85100
2       0.931034     8.0           256.0       2.20   946.0
4.85100
3       0.551724     8.0           128.0       1.22  1244.0
2.69010
4       0.620690     8.0           256.0       1.91   837.0
4.21155

   Screen_Size_inch Price-binned  4.25565  Screen-Full_HD  Screen-
IPS_Panel
0         14.000000          Low        0               0
1
1         15.598425          Low        0               1
0
2         15.598425          Low        0               1
0
3         13.299213          Low        0               0
1
```

```
4           15.598425           Low         0                  1
0
```

Congratulations! You have completed the lab

Authors

Abhishek Gagneja

Vicky Kuo

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2023-09-15	0.1	Abhishek Gagneja	Initial Version Created
2023-09-19	0.2	Vicky Kuo	Reviewed and Revised