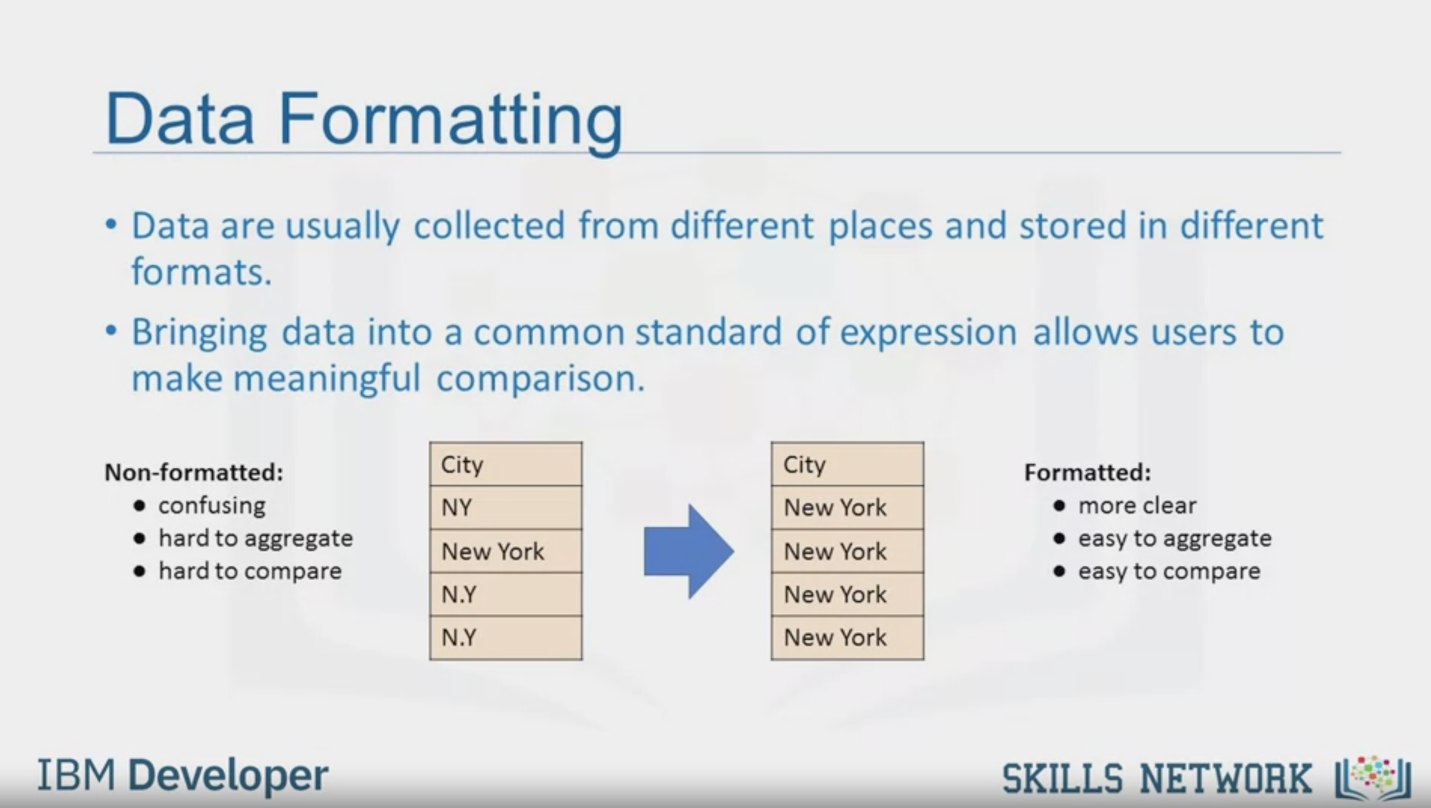


Data Formatting in Python

IBM Developer

SKILLS NETWORK



Data Formatting

• Data are usually collected from different places and stored in different

formats.

• Bringing data into a common standard of expression allows users to

make meaningful comparison.

Non-formatted:

• confusing

• hard to aggregate

• hard to compare

Formatted:

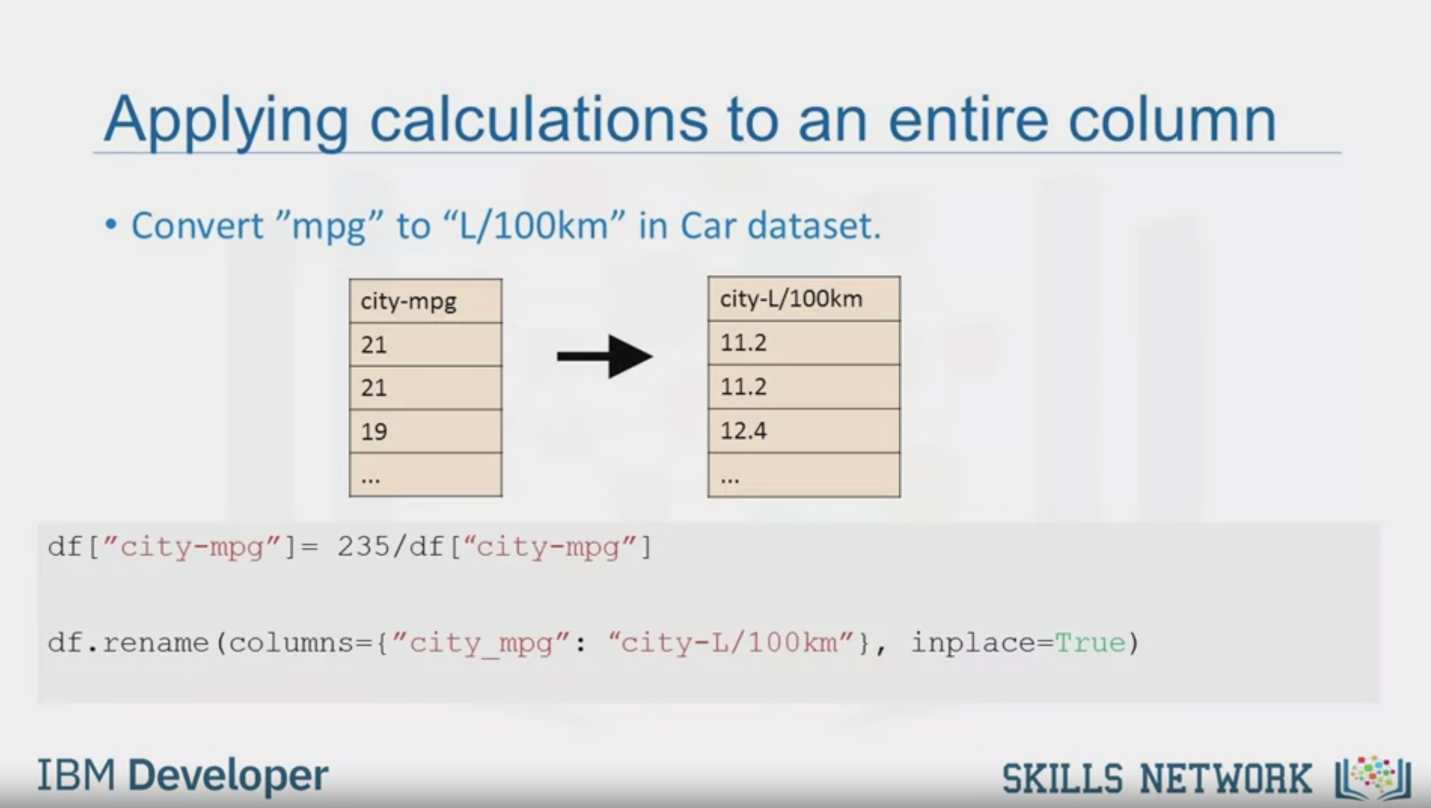
• more clear

• easy to aggregate

• easy to compare

IBM Developer

SKILLS NETWORK



Applying calculations to an entire column

• Convert "mpg" to "L/100km" in Car dataset.

city-mpg

21

21

19

city-L/100km

11.2

11.2

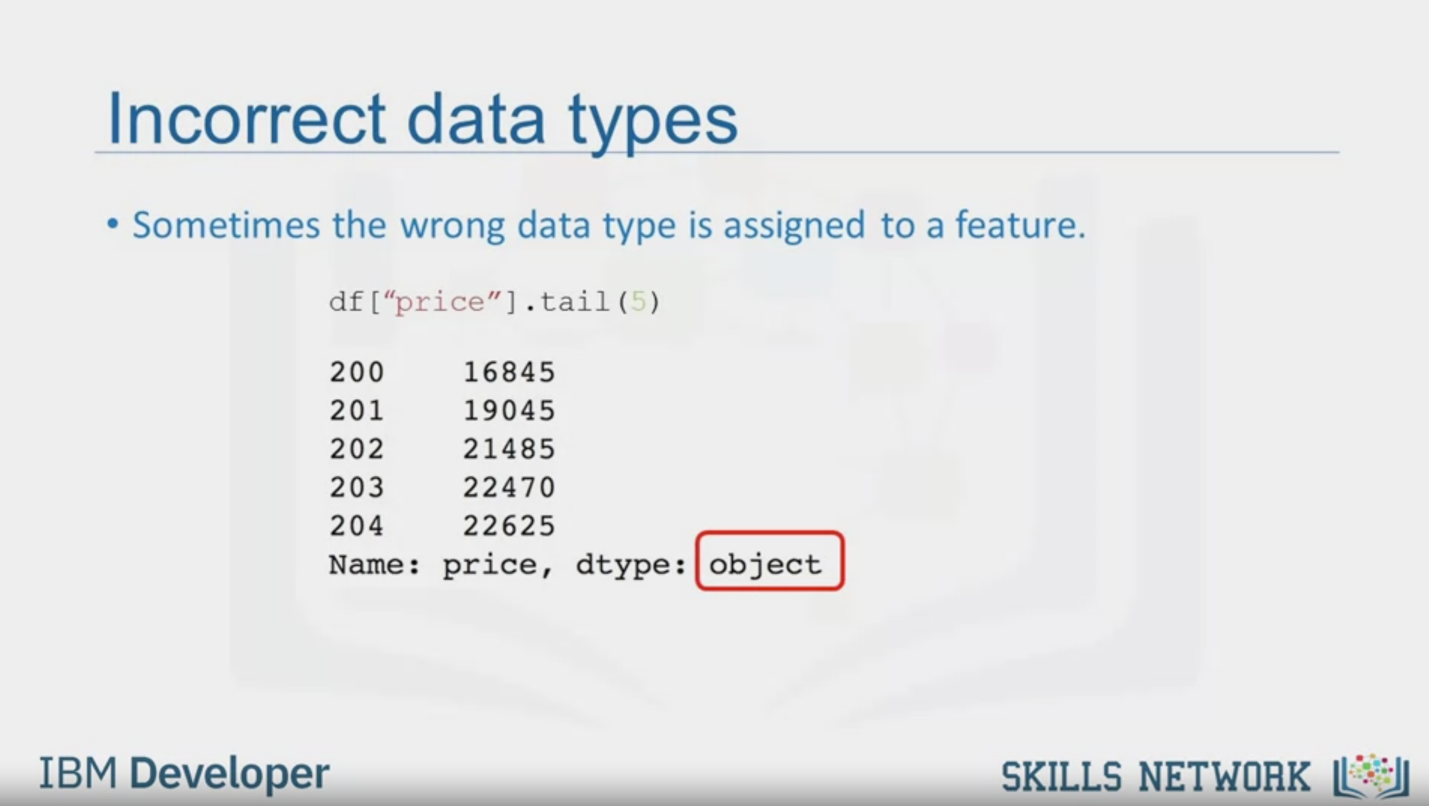
12.4

df["city-mpg"]= 235/df ["city-mpg" ]

df.rename (columns=("city\_mpg": "city-L/100km"}, inplace=True)

IBM Developer

SKILLS NETWORK



Incorrect data types

• Sometimes the wrong data type is assigned to a feature.

df ["price"l.tail (5)

200

16845

201

19045

202

21485

203

22470

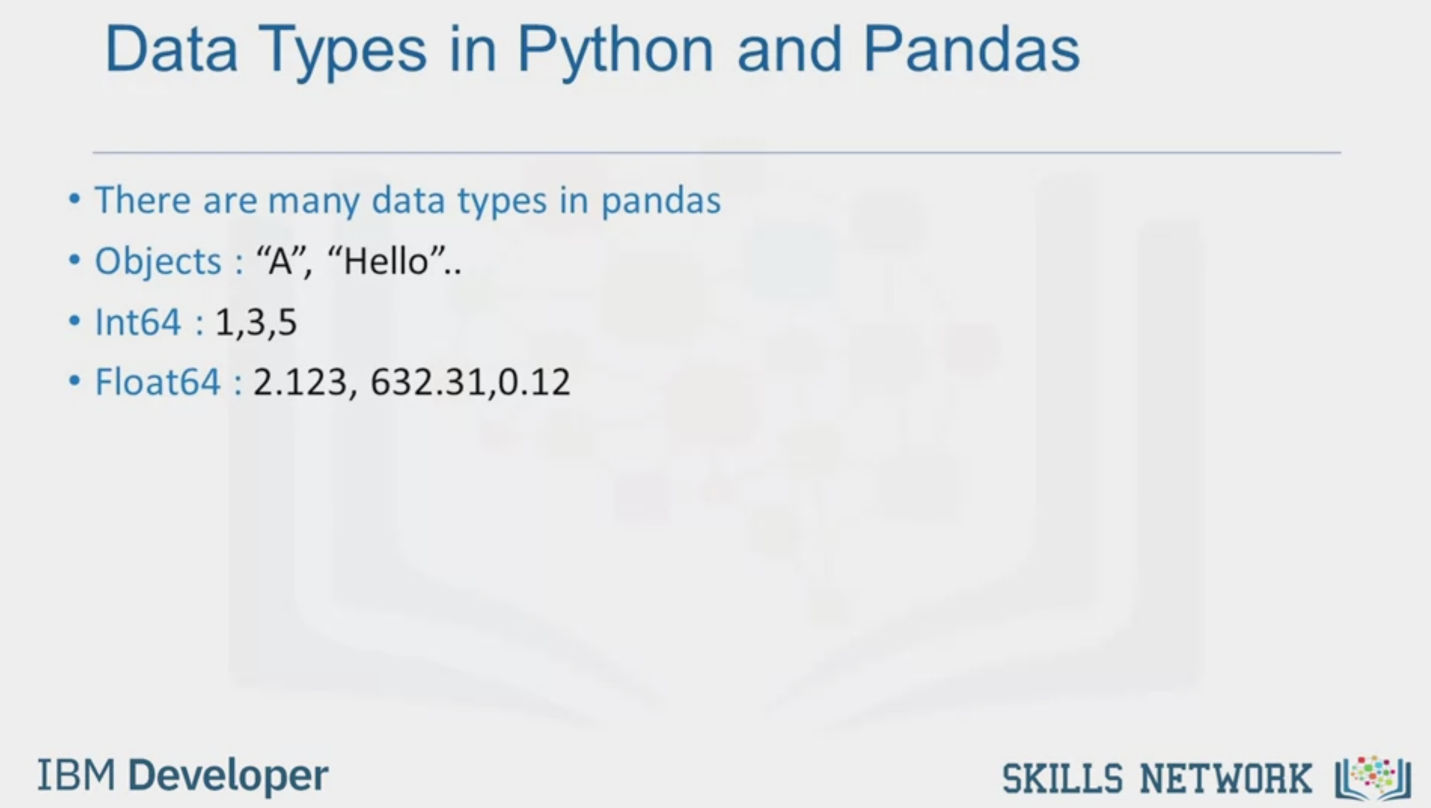
204

22625

Name: price, dtype: object

IBM Developer

SKILLS NETWORK



Data Types in Python and Pandas

• There are many data types in pandas

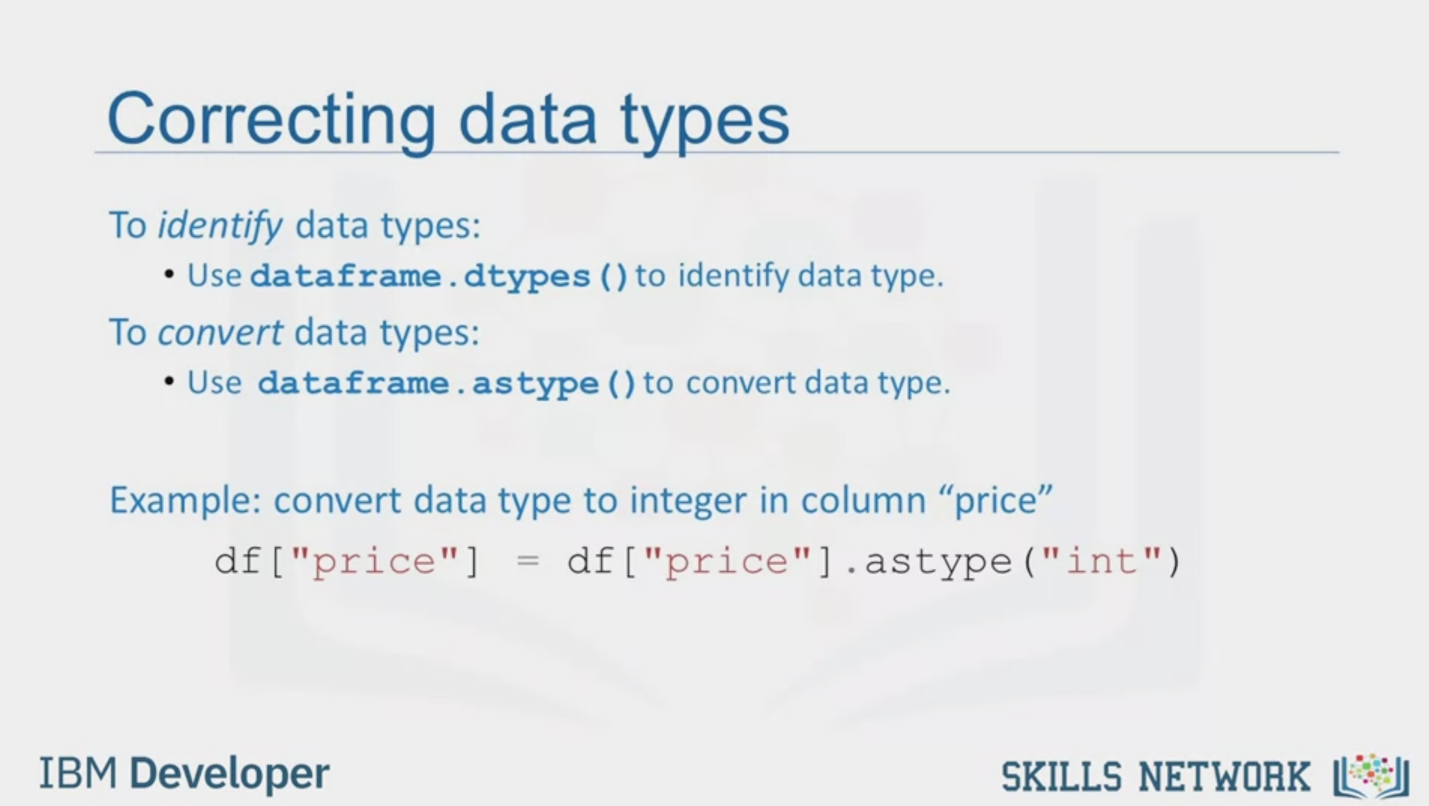
• Objects : "A", "Hello"..

• Int64 : 1,3,5

• Float64 : 2.123, 632.31,0.12

IBM Developer

SKILLS NETWORK



Correcting data types

To identify data types:

• Use dataframe.dtypes () to identify data type.

To convert data types:

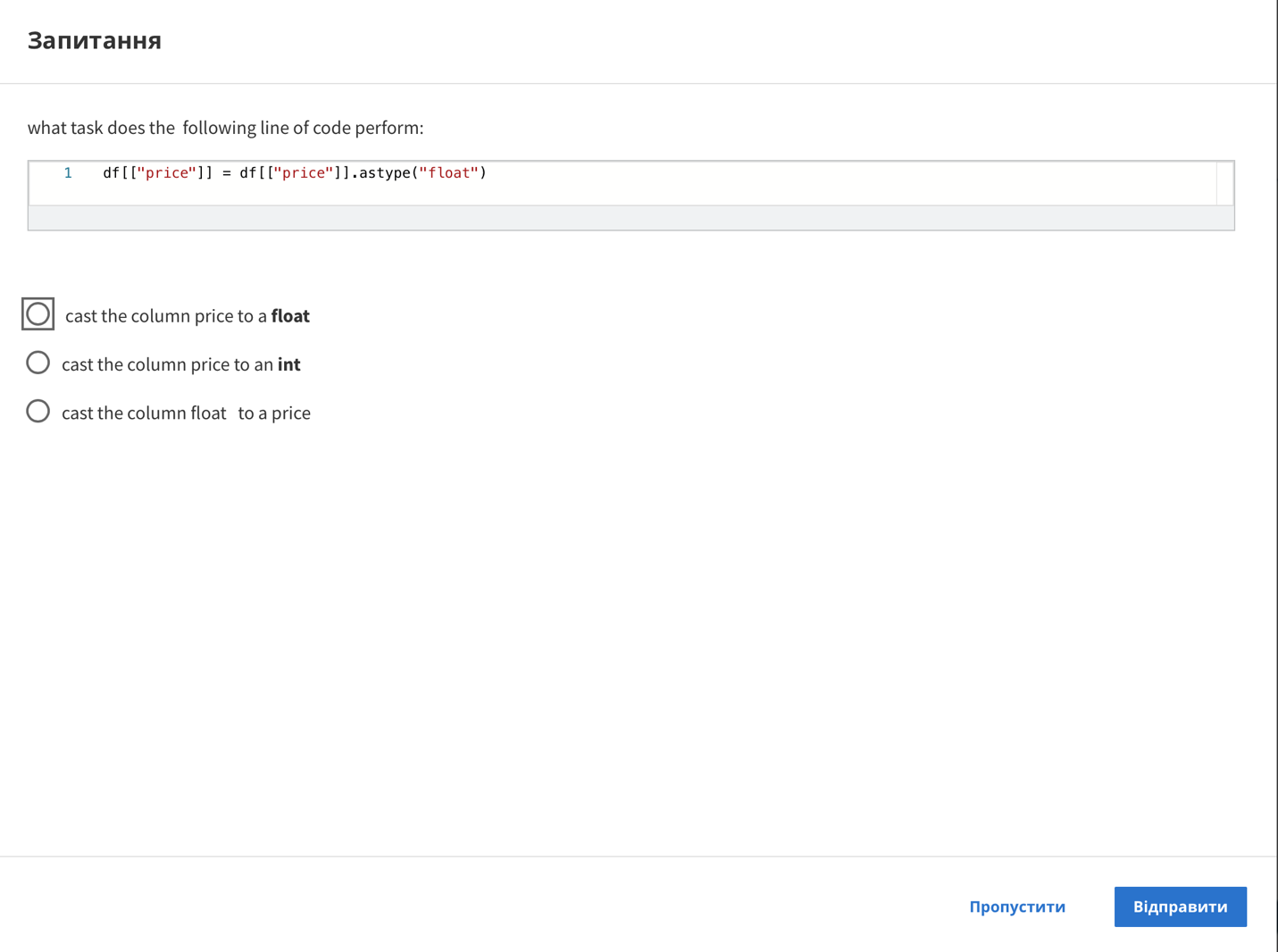
• Use dataframe.astype () to convert data type.

Example: convert data type to integer in column "price"

df["price"] = df["price"].astype ("int")

IBM Developer

SKILLS NETWORK



Question

what task does the following line of code perform:

1 df[["price"]] = df[["price"]].astype("float")

cast the column price to a float

cast the column price to an int

cast the column float to a price

Skip

Send



SKILLS NETWORK

IBM Developer

In this video, we'll look at the problem of data with different formats,

units and conventions and the pandas methods that help us deal with these issues.

Data is usually collected from different places by

different people which may be stored in different formats.

Data formatting means bringing data into a common standard of

expression that allows users to make meaningful comparisons.

As a part of dataset cleaning,

data formatting ensures the data is consistent and easily understandable.

For example, people may use different expressions to represent New York City,

such as uppercase N uppercase Y,

uppercase N lowercase y,

uppercase N uppercase Y and New York.

Sometimes, this unclean data is a good thing to see.

For example, if you're looking at the different ways people tend to write New York,

then this is exactly the data that you want.

Or if you're looking for ways to spot fraud,

perhaps writing N.Y. is more likely

to predict an anomaly than if someone wrote out New York in full.

But perhaps more often than not,

we just simply want to treat them all as the same entity or

format to make statistical analyses easier down the road.

Referring to our used car dataset,

there's a feature named city-miles per gallon in the dataset,

which refers to a car fuel consumption in miles per gallon unit.

However, you may be someone who lives in a country that uses metric units.

So, you would want to convert those values to

liters per 100 kilometers, the metric version.

To transform miles per gallon to liters per 100 kilometers,

we need to divide 235 by each value in the city-miles per gallon column.

In Python, this can easily be done in one line of code.

You take the column and set it to equal to 235,

divide it by the entire column.

In the second line of code,

rename column name from city-miles per gallon to

city-liters per 100 kilometers using the data frame rename method.

For a number of reasons,

including when you import a dataset into Python,

the data type may be incorrectly established.

For example, here we noticed the assigned data type to the price feature is object.

Although the expected data type should really be an integer or float type.

It is important for later analysis to explore

the features data type and convert them to the correct data types.

Otherwise, the developed models later on may behave strangely,

and totally valid data may end up being treated like missing data.

There are many data types in pandas.

Objects can be letters or words.

Int64 are integers and floats are real numbers.

There are many others that we will not discuss.

To identify features data type,

in Python we can use the dataframe.dtypes

method and check the data type of each variable in a data frame.

In the case of wrong data types,

the method dataframe.astype can be

used to convert a data type from one format to another.

For example, using astype int for the price column,

you can convert the object column into an integer type variable.