**Advanced Machine Learning**

**CIS550**

**Spring ‘24**

Lab Homework 3

Submitted by :

Name : Manas Vishal (01971464)

Email : mvishal@umassd.edu

Title : Encoding of ordinal and non-ordinal categorical data

Encoding the data

When data researchers take data, it can often be non-ordinal which means it might not have any numerical value associated with it. For those type of data, machine learning engineers and data scientists have to encode them properly and assign numerical values to it. This is called encoding of data.

About the data

In this exercise I am working with automobile dataset which I got from here : [Automobile - UCI Machine Learning Repository](https://archive.ics.uci.edu/dataset/10/automobile)

The data has following attributes :

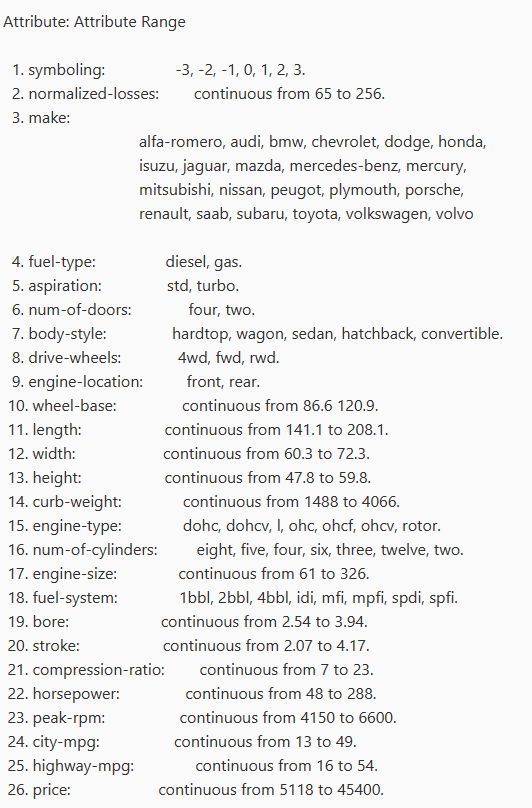


Fig. 1 Data and its attributes



Fig. 2 Importing necessary modules and setting parameters for pandas

The dataset is downloaded by using the .get function from requests module.

 Fig. 2 Downloading and extracting the data locally

The extracted data is unstructured in .data and .names files so we need to pack it in dataframe using the column names given.



Fig. 3 Generating a dataframe using the data and column names

We examine the data by checking its shape and head

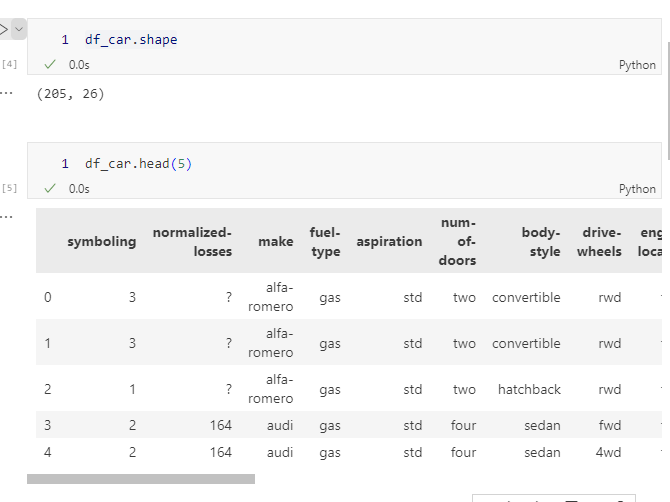


Fig. 4 Printing off the data and its values on screen

To get more information about the dataframe we use the info function

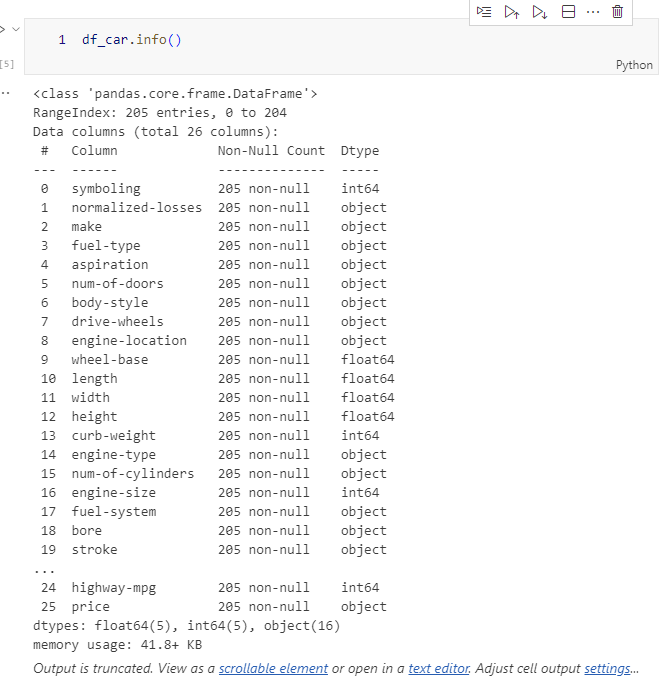


Fig. 5 Data Info

I trim the data that I will not be using by reassigning the dataframe into a new variable and using the column names that I want to work on.



Fig. 6 Trimming the data

Now we can work on this subset of data for our lab. We can encode the following data.

We can see the num-of-cylinders field is **text but does have an ordinal value** so we will have to make it ordered by assigning numerical values to it.

While for aspiration and drive-wheels, the data is not ordinal. They have to be mapped to a numerical value somehow.

Encoding the ordinal data

We start with first getting the info for the new data.

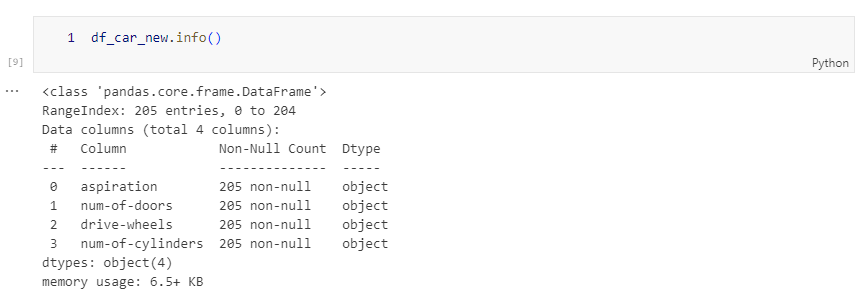


Fig. 7 Subset of data

We can see from the screenshot above that all the columns have 205 values. Lets work on the num-of-doors field for now.

We print off the value counts for the num-of-doors field/feature.

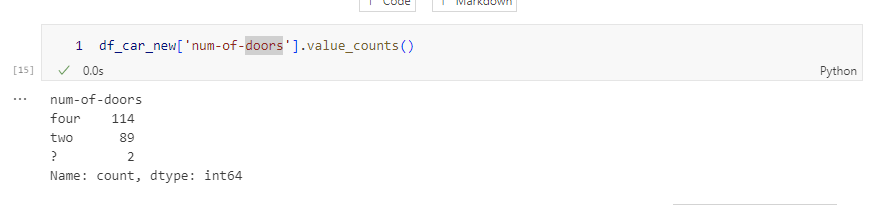


Fig. 8 Value counts of the num-of-doors feature

This feature has three values : four, two and unknown. We can create a simple mapper and we can assign numerics for all. I choose unknown to be 0.



Fig. 9 Remapping to encode the dataframe using replace function and a remapper dictionary

We repeat the same process for another field **num-of-cylinders**

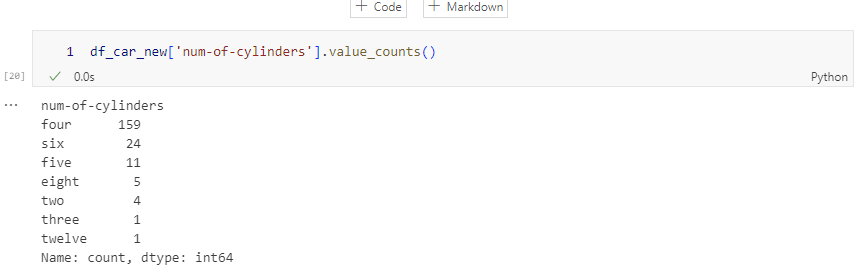
****

Fig. 10 Getting the values

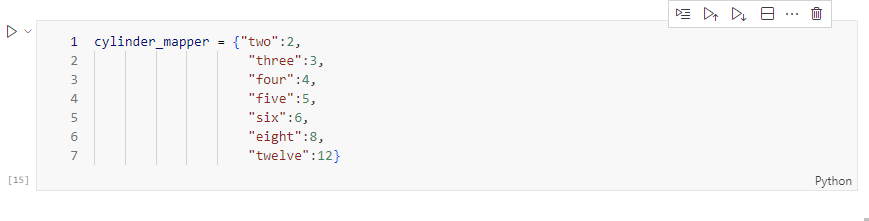


Fig. 11 Remapping dictionary

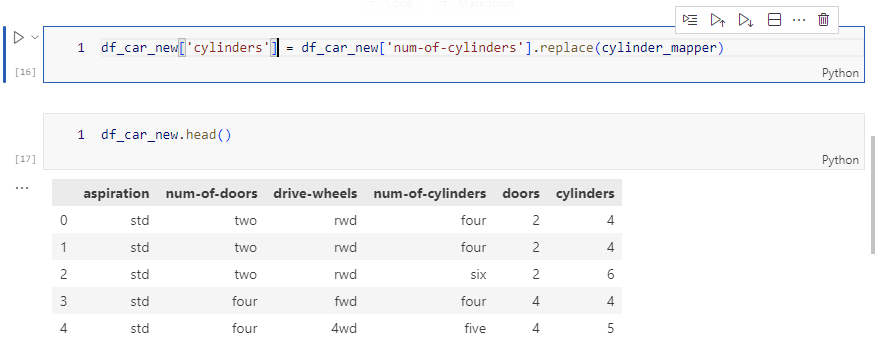


Fig. 12 Remapped-encoded dataframe

Encoding the non-ordinal data

In this dataframe, we have a few columns that are non-ordinal. We can use the get\_dummies function of pandas to encode these features and add new binary features to dataframe.

For e.g., the drive-wheels feature can take 4wd, fwd,rwd as values. As mentioned in the lab, one can think of assigning some number to these features but by doing that we add extra order to the system which is not present at all. That’s why we have to rely on binary for these type of data. They are represented well as binary features. This process is called one-hot encoding or dummying statistics.

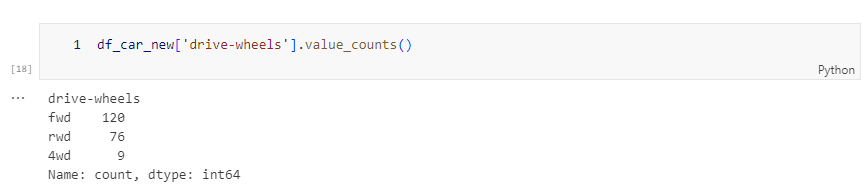


Fig. 13 Value counts of drive-wheels feature

As expected from the attribute table, we have three possible values of drive-wheels



Fig. 14 Using the get\_dummies function to convert the feature to binary

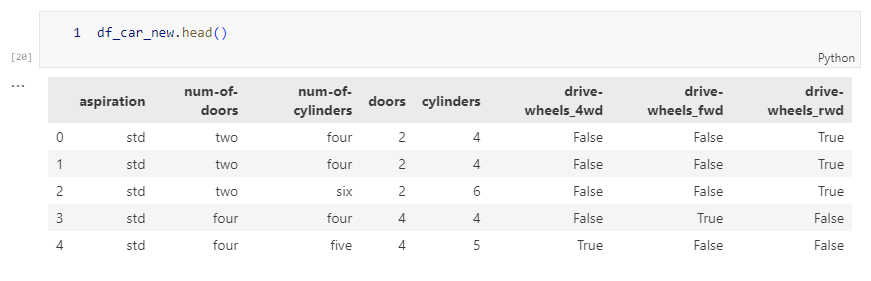


Fig. 15 Encoded dataframe of non-ordinal data

We can see there are 3 new columns corresponding the 3 values of the feature we expected. These new columns are either True or False which means they are binary. In short we are unpacking the num-of-wheels column into three separate binary columns.

We can repeat the same thing for aspiration column. We could do the same thing as we did for above feature. But here, the data is either std or turbo, so we get to choose which feature gets to be mapped with True or False. Here, I will use the drop\_first function and just choose turbo as true or false which means if the engine is turbo, it will get mapped to 1 or True.

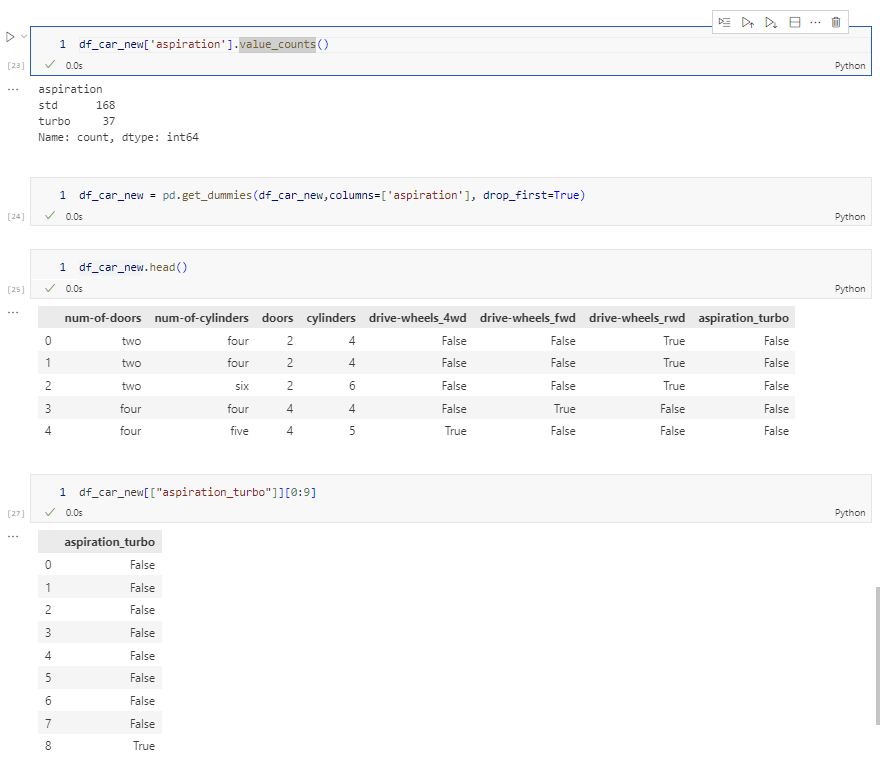
****

Fig. 16 Encoded data for aspiration feature

Challenge Task

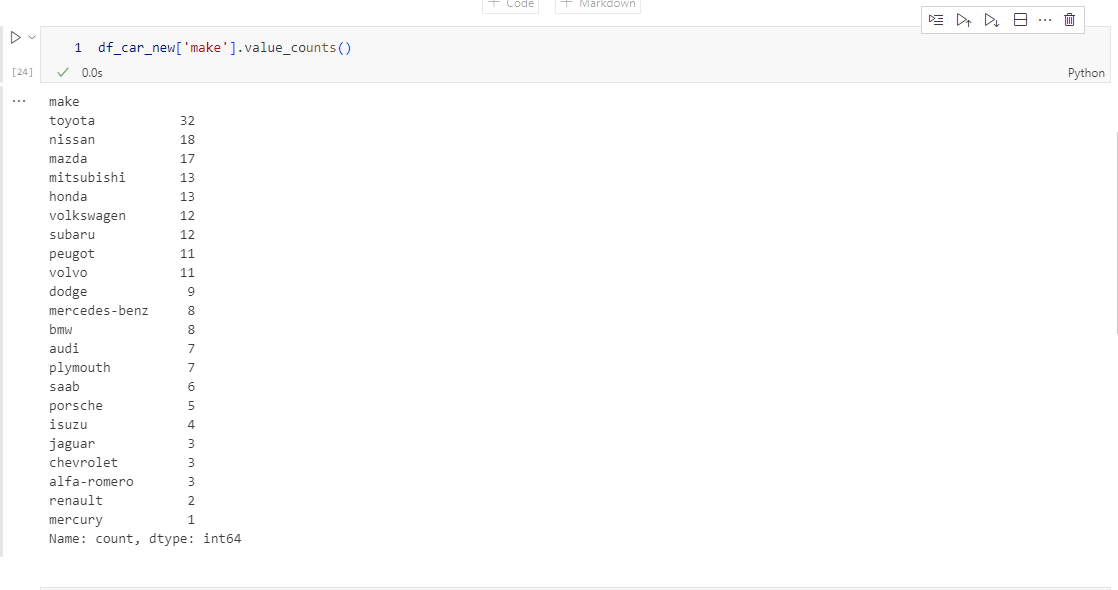
For the challenge task, I am adding three new columns : 'normalized-losses','make', 'fuel-type'

First field of normalized-losses does not need any encoding as it is a continuous attribute. However, it has missing values so we need to remap those values to either zero or some other constant.



Fig. 17 Adding three new columns

The make field has 22 available options.



Here, we will use the get\_dummies function to assign binary feature to the new dataframe.

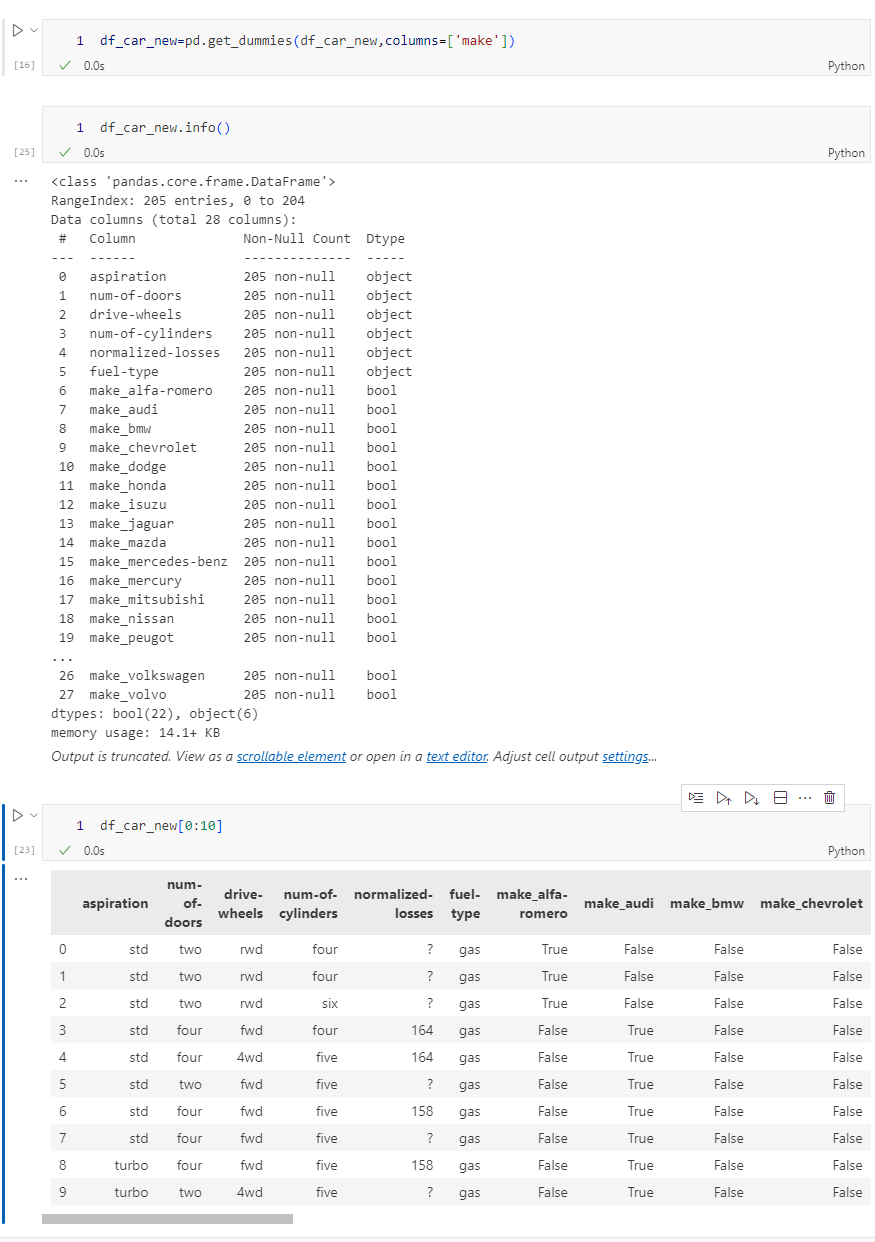


Fig. 18 New dataframe with extra Boolean/binary features associated to the make column

For the fuel-type column, I use drop\_first function to directly the column into fule-type\_gas column which is a Boolean/binary feature..

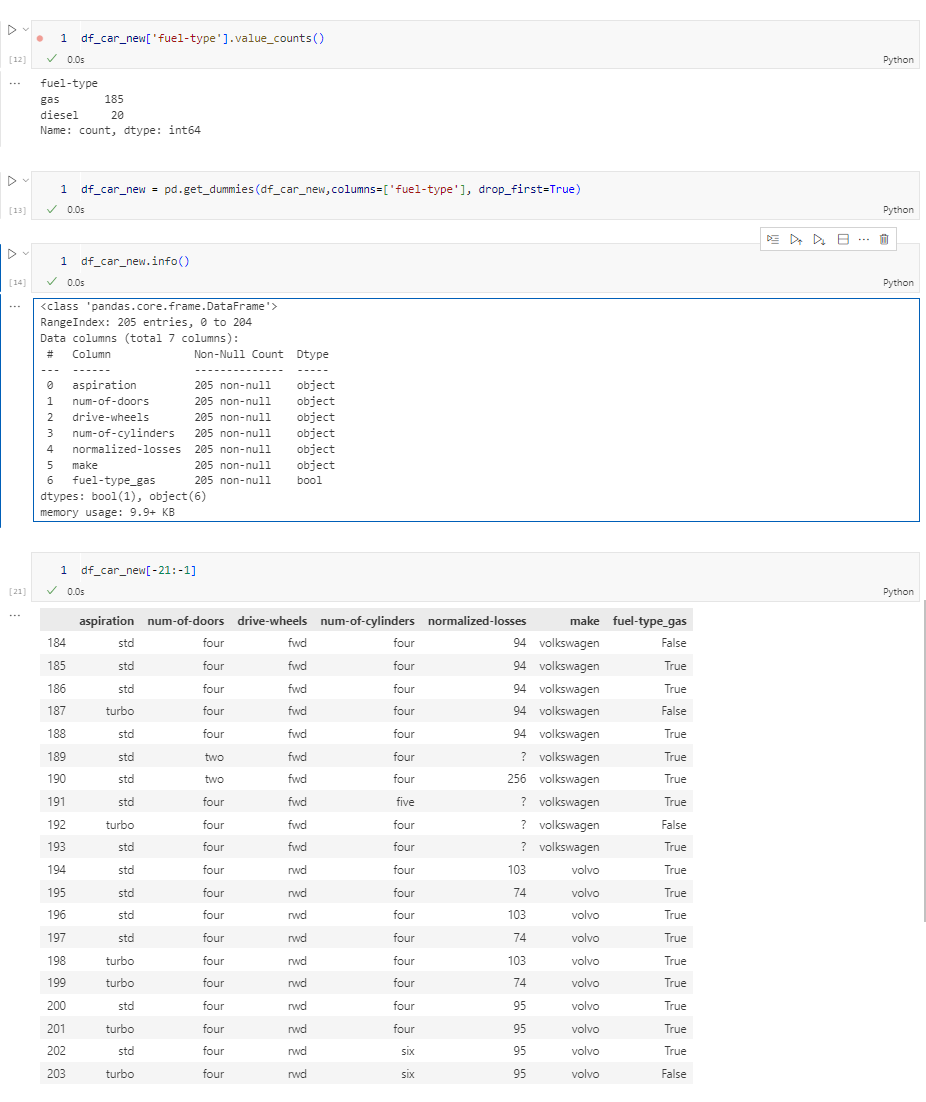


Fig. 19 Encoding the non-ordinal data

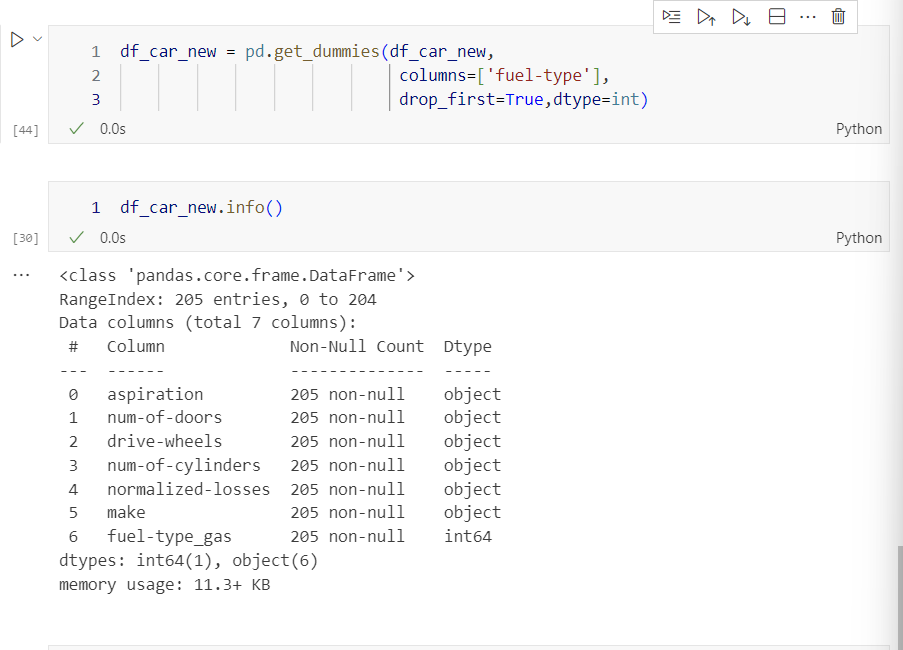
Conclusion

In this lab, I learnt a lot about data types in data mining. Preprocessing of data is very important as if not done correctly, it can lead to incorrect or even irrelevant data analysis/prediction from machine learning. The data is acquired from different sources and can be packed in different ways. In our case, the dataset we worked on today was also packed non-ordinally for some features. It is very important to encode it properly to be passed in the machine learning architecture. We learnt ways to encode the non-ordinal as well as ordinal data.

I look forward to using these methods in future to encode raw dataset to feed into machine learning algorithms.

Edit : New update to the homework for returning 0 and 1’s instead of Boolean in the challenge task

Passing an extra argument to the get\_dummies function will return 0 and 1’s instead of Boolean



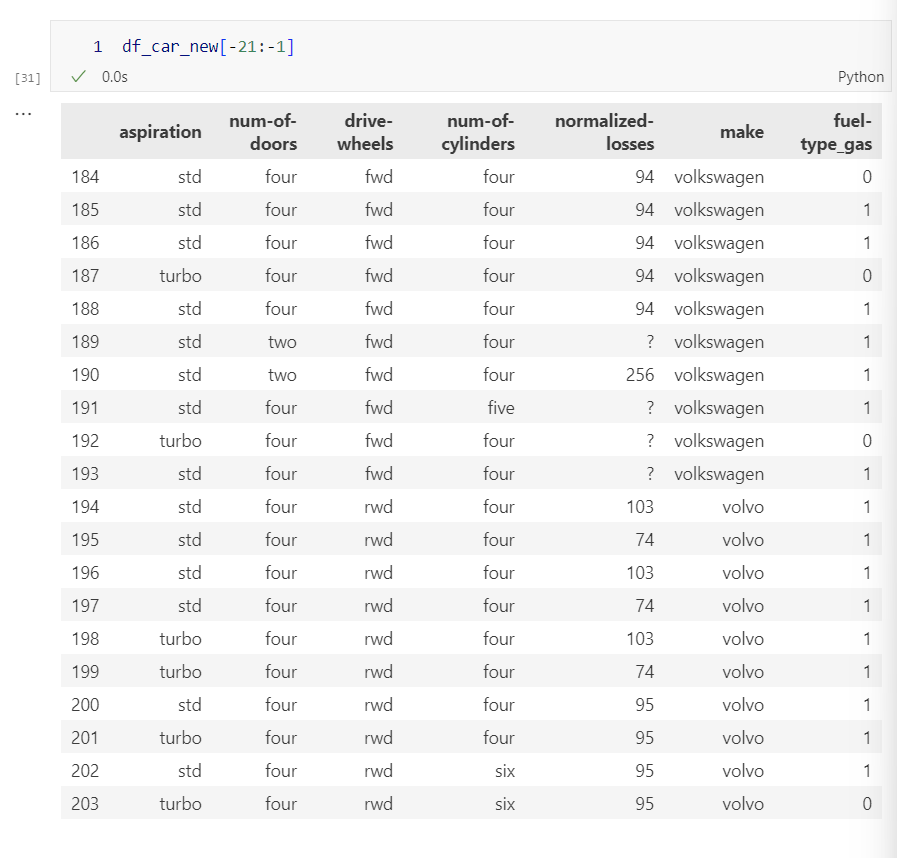


Fig. Updated challenge task