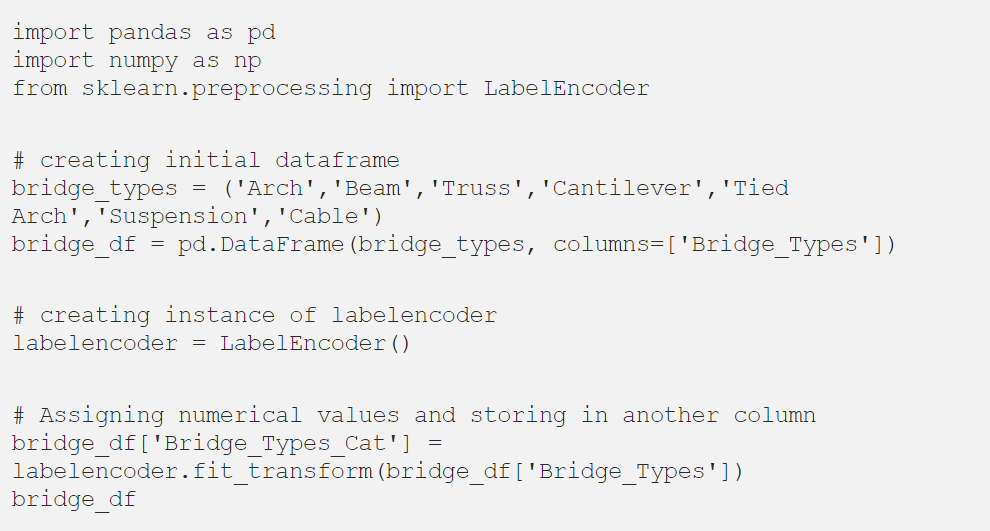
**1.08 Encoding Model**

Label Encoding in Python



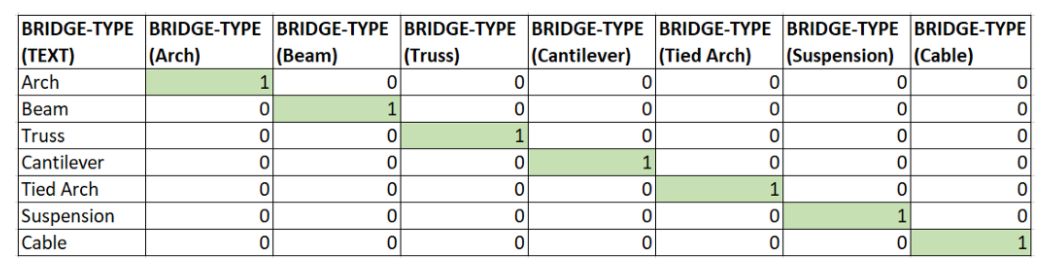
Using sci-kit learn library approach:

Another common approach which many data analyst perform label-encoding is by using SciKit learn library.



**One-Hot Encoder**

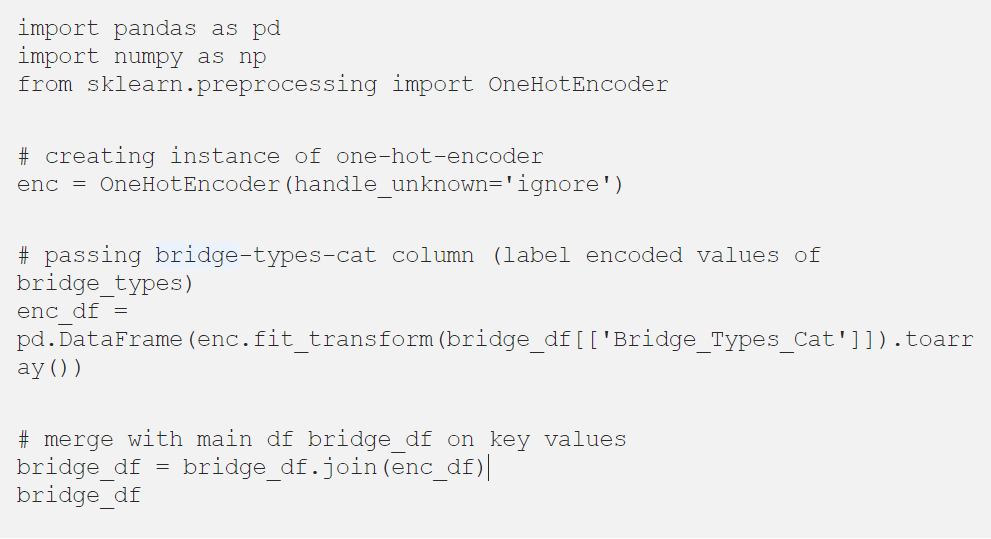
Though label encoding is straight but it has the disadvantage that the numeric values can be misinterpreted by algorithms as having some sort of hierarchy/order in them. This ordering issue is addressed in another common alternative approach called ‘One-Hot Encoding’. In this strategy, each category value is converted into a new column and assigned a 1 or 0 (notation for true/false) value to the column. Let’s consider the previous example of bridge type and safety levels with one-hot encoding.

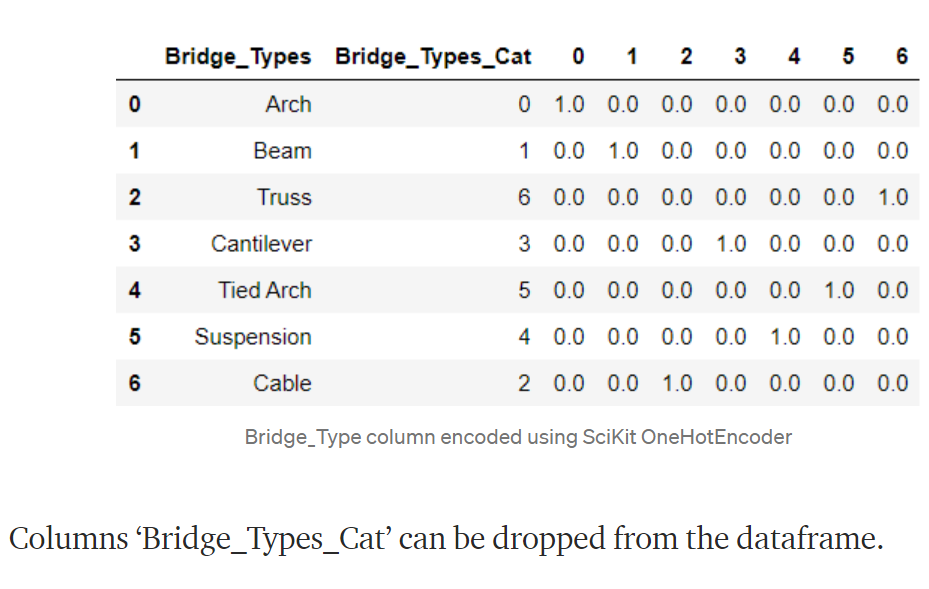


One-Hot Encoding in Python

Using sci-kit learn library approach:

OneHotEncoder from SciKit library only takes numerical categorical values, hence any value of string type should be label encoded before one hot encoded. So, taking the dataframe from the previous example, we will apply OneHotEncoder on column Bridge\_Types\_Cat.





**Conclusion**

It is important to understand various option for encoding categorical variables because each approach has its own pros and cons. In data science, it is an important step, so I really encourage you to keep these ideas in mind when dealing with categorical variables. For any suggestion or for more details on the code used in this article, feel free to comment.

**From data pre-processing to Optimizing a Regression linear model**

**1. What is Data pre-processing and why it is needed?**

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data, Noisy: containing errors or outliers. Inconsistent: containing discrepancies in codes or names. Data preprocessing is a proven method of resolving such issues.

In Real-world data are generally incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data. Noisy: containing errors or outliers. Inconsistent: containing discrepancies in codes or names.

**1.1) Steps in Data Preprocessing**

Step 1: Import the libraries  
Step 2: Import the data-set  
Step 3: Check out the missing values  
Step 4: Encode the Categorical data  
Step 5: Splitting the dataset into Training and Test set  
Step 6: Feature scaling

Let’s discuss all these steps in details.

**Step 1: Import the libraries**

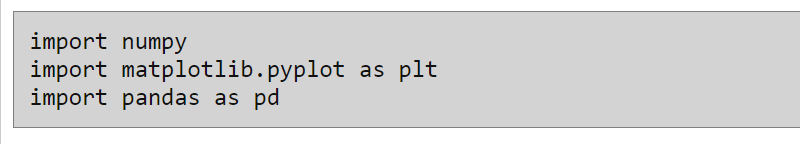
A **library** is also a collection of implementations of behavior, written in terms of a language, that has a well-defined interface by which the behavior is invoked. For instance, people who want to write a higher-level program can use a library to make system calls instead of implementing those system calls over and over again. — [Wikipedia](https://en.wikipedia.org/wiki/Library_(computing)?source=post_page---------------------------)

We need to import 3 essential python libraries.

1. **[Numpy](https://www.numpy.org/?source=post_page---------------------------" \t "_blank)** is the fundamental package for scientific computing with Python.

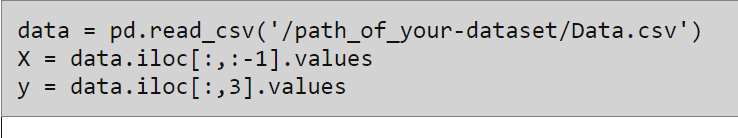
2. [**Pandas**](https://pandas.pydata.org/?source=post_page---------------------------) *is for*data manipulation and analysis*.*

3. [**Matplotlib**](https://matplotlib.org/?source=post_page---------------------------) is a Python 2D plotting library which produces publication quality figures in a variety of hard copy formats and interactive environments across platforms.



**Step 2: Import the data-set**

Data is imported using the pandas library.

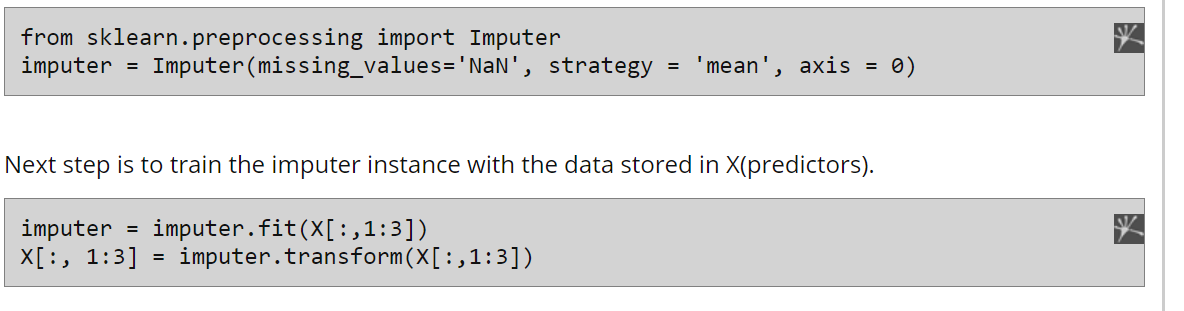


**Step 3: Check out the missing values**

There are two ways by which we can handle missing values in our dataset. The first method commonly used to handle null values. Here, we either delete a particular row if it has a null value for a particular feature and a particular column if it has more than 75% of missing values. This method is advised only when there are enough samples in the data set. One has to make sure that after we have deleted the data, there is no addition of bias.

In the second method, we replace all the NaN values with either mean, median or most frequent value. This is an approximation which can add variance to the data set. But the loss of the data can be negated by this method which yields better results compared to removal of rows and columns. Replacing with the above three approximations are a statistical approach to handling the missing values. This method is also called as **leaking the data** while training.

For dealing with missing data, we will use [Imputer](https://scikit-learn.org/stable/modules/impute.html?source=post_page---------------------------) library from sklearn.preprocessing package. Instead of providing mean you can also provide median or most frequent value in the strategy parameter.



**Step 4: Encode the Categorical data**

Categorical data are variables that contain label values rather than numeric values. The number of possible values is often limited to a fixed set.  
Some examples include:

A “pet” variable with the values: “dog” and “cat”.

A “color” variable with the values: “red”, “green” and “blue”.

A “place” variable with the values: “first”, “second” and “third”.

Each value represents a different category.

### Note: What is the Problem with Categorical Data?

Some algorithms can work with categorical data directly. But many machine learning algorithms cannot operate on label data directly. They require all input variables and output variables to be numeric.

In general, this is mostly a constraint of the efficient implementation of machine learning algorithms rather than hard limitations on the algorithms themselves. This means that categorical data must be converted to a numerical form. If the categorical variable is an output variable, you may also want to convert predictions by the model back into a categorical form in order to present them or use them in some application.

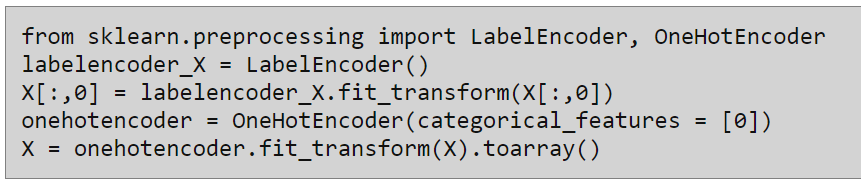
We are going to use a technique called label encoding. Label encoding is simply converting each value in a column to a number. For example, the body\_style column contains 5 different values. We could choose to encode it like this:

* convertible -> 0
* hardtop -> 1
* hatchback -> 2
* sedan -> 3
* wagon -> 4

To implement Label encoding we will import LabelEncoder from sklearn.preprocessing package. But it labels categories as 0,1,2,3…. Now since 0<1<2, the equations in your regression model may thing one category has a higher value than the other, which is of course not true.

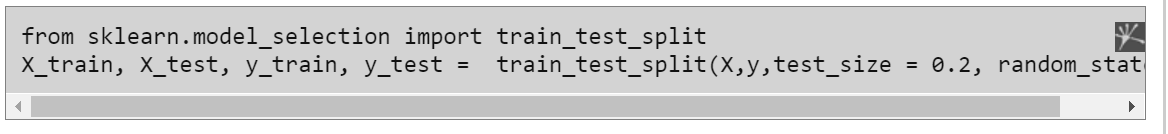
To solve this situation we have a concept called [Dummy variables](https://en.wikipedia.org/wiki/Dummy_variable_(statistics)?source=post_page---------------------------). In regression analysis, a **dummy variable** is one that takes the value 0 or 1 to indicate the absence or presence of some categorical effect that may be expected to shift the outcome. They are used as devices to sort data into mutually exclusive categories (such as smoker/non-smoker, etc.).

To implement the concept of dummy variables we will import [OneHotEncoder](https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html?source=post_page---------------------------" \t "_blank) library from sklearn.preprocessing package. You need to provide the column index which needs to be encoded under categorical\_features. So if a column has 3 categories, 3 columns will be created and likewise for any number of categories.



**Step 5: Splitting the dataset into Training and Test set**

Second last step in our data pre-processing is splitting the data into training and test set.



Regression Accuracy Check in Python (MAE, MSE, RMSE, R-Squared)

**Regression accuracy metrics**  
  
   The MSE, MAE, RMSE, and R-Squared are mainly used metrics to evaluate the prediction error rates and model performance in regression analysis.

* **MAE** (Mean absolute error) represents the difference between the original and predicted values extracted by averaged the absolute difference over the data set.
* **MSE** (Mean Squared Error) represents the difference between the original and predicted values extracted by squared the average difference over the data set.
* **RMSE** (Root Mean Squared Error) is the error rate by the square root of MSE.
* **R-squared** (Coefficient of determination) represents the coefficient of how well the values fit compared to the original values. The value from 0 to 1 interpreted as percentages. The higher the value is, the better the model is.

The above metrics can be expressed,

