17.6.3

Create Custom Encoding

Jill explains that not every machine learning task can be performed by out-of-the-box solutions, meaning libraries written by other programmers. Sometimes you have to roll up your sleeves and write your own custom code!

It's also possible to create custom encoding functions. To understand why this might be useful, let's first look at using the LabelEncoder module. With it, you'll transform the month column into numbers. The goal is to transform each month into its corresponding order: for example, January should be transformed to 1, since it's the first month of the year. Similarly, July should be transformed to 7, since it's the seventh month of the year:

```
label encoder = LabelEncoder()
loans_df["month_le"] = label_encoder.fit_transform(loans_df["month"])
loans df.head()
   amount term
                     month
                                          education gender bad month_le
0
      1000
                                 High School or Below
                                                              0
              30
                      June
                             45
                                                       male
1
      1000
             30
                       July
                             50
                                           Bachelor
                                                     female
                                                              0
                                                                        5
      1000
              30
                     August
                             33
                                           Bachelor
                                                     female
                                                              0
                                                                        1
3
      1000
              15
                 September
                             27
                                             college
                                                      male
                                                              0
                                                                       11
      1000
              30
                    October
                                                                       10
                                             college
                                                     female
```

Note that a new instance of <u>LabelEncoder</u> was created here as <u>label_encoder</u>. The month of August, for example, is converted to 1 instead of 8. July is converted to 5 instead of 7.

Instead, we can create a dictionary of the months of the year and apply a custom function to convert the month names to their corresponding integers:

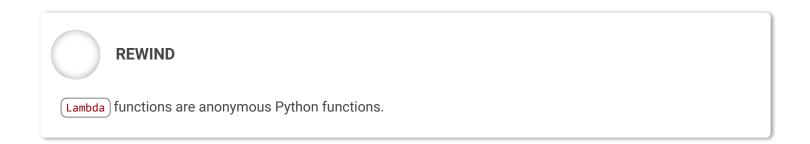
```
months_num = {
    "January": 1,
    "February": 2,
    "March": 3,
    "April": 4,
    "May": 5,
    "June": 6,
    "July": 7,
    "August": 8,
    "September": 9,
    "October": 10,
    "November": 11,
    "December": 12,
}
```

In the next cell, a lambda function is applied to the month column to perform the actual conversion:

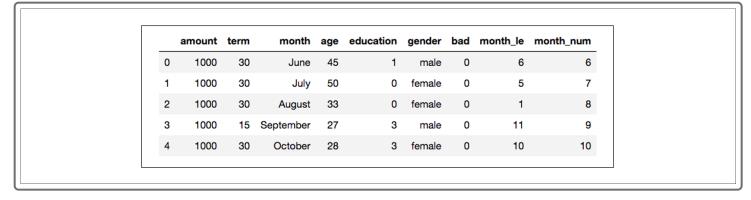
```
loans_df["month_num"] = loans_df["month"].apply(lambda x: months_num[x])
```

The following actions are taking place:

- A transformation is made to the values of the month column, and the transformed values are placed in the month_num column.
- The (apply()) method runs the function inside its parentheses on each element of the (month) column.
- The lambda function takes an argument (x), and returns months_num[x]. For example, if the value in the month column is "June," the function returns months_num["June"], which is 6.



The DataFrame's (month_num) column now displays each month as a number:



The code in the next cell is merely cleanup—it drops the unnecessary columns related to the month:

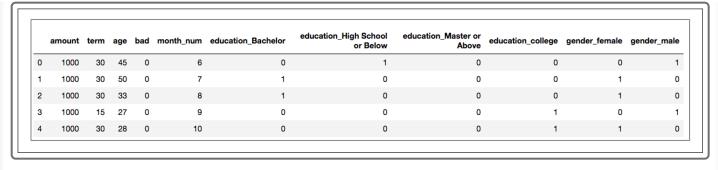
```
loans_df = loans_df.drop(["month", "month_le"], axis=1)
loans_df.head()
```

0 1000 30 45 1 male 0 6 1 1000 30 50 0 female 0 7 2 1000 30 33 0 female 0 8 3 1000 15 27 3 male 0 9 4 1000 30 28 3 female 0 10		amount	term	age	education	gender	bad	$month_num$
2 1000 30 33 0 female 0 8 3 1000 15 27 3 male 0 9	0	1000	30	45	1	male	0	6
3 1000 15 27 3 male 0 9	1	1000	30	50	0	female	0	7
	2	1000	30	33	0	female	0	8
4 1000 30 28 3 female 0 10	3	1000	15	27	3	male	0	9
	4	1000	30	28	3	female	0	10

```
Create a new Jupyter Notebook and open loans_data.csv as a Pandas DataFrame.

Encode the following labels of the dataset: month, education, and gender. Then save your DataFrame as loans_data_encoded.csv.

Your DataFrame should look like this:
```



© 2020 - 2022 Trilogy Education Services, a 2U, Inc. brand. All Rights Reserved.