Simple Methods to deal with Categorical Variables in Predictive Modeling

BUSINESS ANALYTICS INTERMEDIATE MACHINE LEARNING PYTHON TECHNIQUE

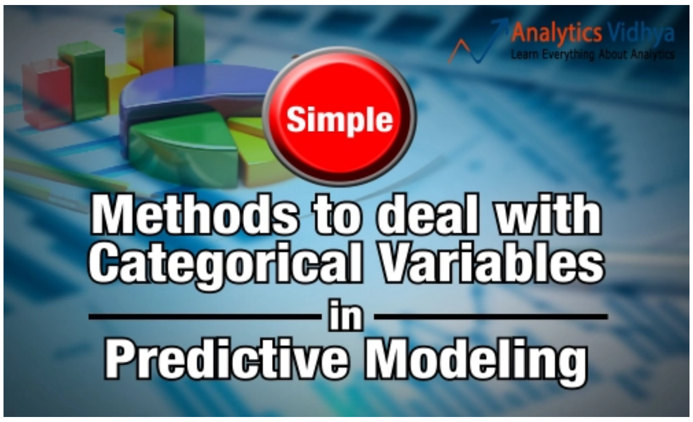
Introduction

Categorical variables are known to hide and mask lots of interesting information in a data set. It's crucial to learn the methods of dealing with such variables. If you won't, many a times, you'd miss out on finding the most important variables in a model. It has happened with me. Initially, I used to focus more on numerical variables. Hence, never actually got an accurate model. But, later I discovered my flaws and learnt the art of dealing with such variables.

If you are a smart data scientist, you'd hunt down the categorical variables in the data set, and dig out as much information as you can. Right? But if you are a beginner, you might not know the smart ways to tackle such situations. Don't worry. I am here to help you out.

After receiving a lot of requests on this topic, I decided to write down a clear approach to help you improve your models using categorical variables.

Note: This article is best written for beginners and newly turned predictive modelers. If you are an expert, you are welcome to share some useful tips of dealing with categorical variables in the comments section below.

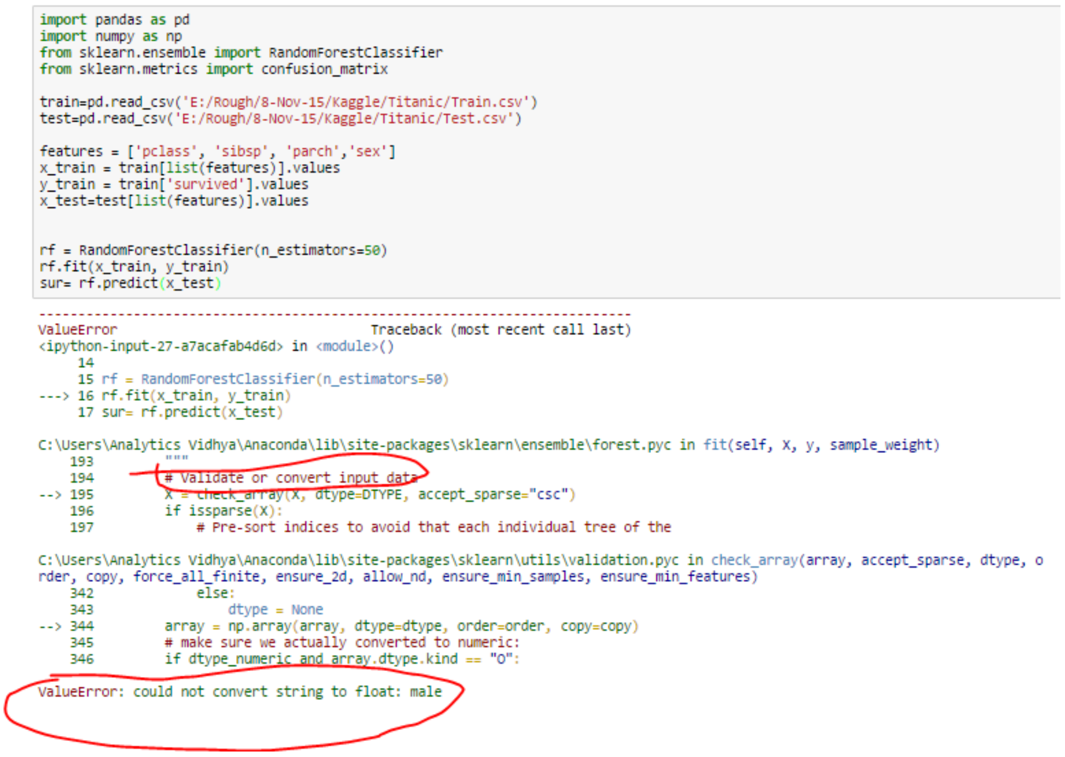


What are the key challenges with categorical variable?

I've had nasty experience dealing with categorical variables. I remember working on a data set, where it took me more than 2 days just to understand the science of categorical variables. I've faced many such instances where error messages didn't let me move forward. Even, my proven methods didn't improve the situation.

But during this process, I learnt how to solve these challenges. I'd like to share all the challenges I faced while dealing with categorical variables. You'd find:

- A categorical variable has too many levels. This pulls down performance level of the model. For example, a cat. variable "zip code" would have numerous levels.
- A categorical variable has levels which rarely occur. Many of these levels have minimal chance of making a real impact on model fit. For example, a variable 'disease' might have some levels which would rarely occur.
- There is one level which always occurs i.e. for most of the observations in data set there is only one level. Variables with such levels fail to make a positive impact on model performance due to very low variation.
- If the categorical variable is masked, it becomes a laborious task to decipher its meaning. Such situations are commonly found in data science competitions.
- You can't fit categorical variables into a regression equation in their raw form. They must be treated.
- Most of the algorithms (or ML libraries) produce better result with numerical variable. In python, library "sklearn" requires features in numerical arrays. Look at the below snapshot. I have applied random forest using sklearn library on titanic data set (only two features sex and pclass are taken as independent variables). It has returned an error because feature "sex" is categorical and has not been converted to numerical form.



```
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix

train=pd.read_csv('E:/Rough/8-Nov-15/Kaggle/Titanic/Train.csv')
test=pd.read_csv('E:/Rough/8-Nov-15/Kaggle/Titanic/Test.csv')

features = ['pclass', 'sibsp', 'parch','sex']
x_train = train[list(features)].values
y_train = train['survived'].values
x_test=test[list(features)].values


rf = RandomForestClassifier(n_estimators=50)
rf.fit(x_train, y_train)
sur= rf.predict(x_test)
```

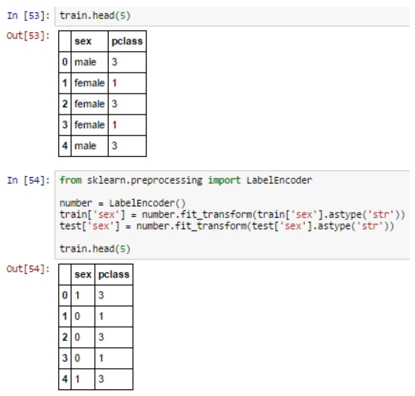
```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-27-a7acafab4d6d> in <module>()
     14
     15 rf = RandomForestClassifier(n_estimators=50)
---> 16 rf.fit(x_train, y_train)
     17 sur= rf.predict(x_test)

C:\Users\Analytics Vidhya\Anaconda\lib\site-packages\sklearn\ensemble\forest.pyc in fit(self, X, y, sample_weight)
    193         """
    194         # Validate or convert input data
--> 195         X = check_array(X, dtype=DTYPE, accept_sparse="csc")
    196         if issparse(X):
    197             # Pre-sort indices to avoid that each individual tree of the

C:\Users\Analytics Vidhya\Anaconda\lib\site-packages\sklearn\utils\validation.pyc in check_array(array, accept_sparse, dtype, o
rder, copy, force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_features)
    342             else:
    343                 dtype = None
--> 344         array = np.array(array, dtype=dtype, order=order, copy=copy)
    345         # make sure we actually converted to numeric:
    346         if dtype_numeric and array.dtype.kind == "O":

ValueError: could not convert string to float: male
```

Proven methods to deal with Categorical Variables

Here are some methods I used to deal with categorical variable(s). A trick to get good result from these methods is 'Iterations'. You must know that all these methods may not improve results in all scenarios, but we should iterate our modeling process with different techniques. Later, evaluate the model performance. Below are the methods:

Convert to Number

- **Convert to number:** As discussed above, some ML libraries do not take categorical variables as input. Thus, we convert them into numerical variables. Below are the methods to convert a categorical (string) input to numerical nature:
  - **Label Encoder:** It is used to transform non-numerical labels to numerical labels (or nominal categorical variables). Numerical labels are always between 0 and n_classes-1.



```
In [53]: train.head(5)
```

Out[53]:

	sex	pclass
0	male	3
1	female	1
2	female	3
3	female	1
4	male	3

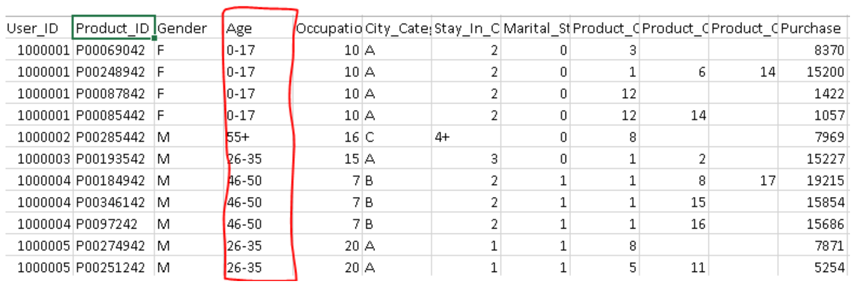
```
In [54]: from sklearn.preprocessing import LabelEncoder

         number = LabelEncoder()
         train['sex'] = number.fit_transform(train['sex'].astype('str'))
         test['sex'] = number.fit_transform(test['sex'].astype('str'))

         train.head(5)
```

Out[54]:

	sex	pclass
0	1	3
1	0	1
2	0	3
3	0	1
4	1	3

A common challenge with nominal categorical variable is that, it may decrease performance of a model. For example: We have two features “age” (range: 0-80) and “city” (81 different levels). Now, when we’ll apply label encoder to ‘city’ variable, it will represent ‘city’ with numeric values range from 0 to 80. The ‘city’ variable is now similar to ‘age’ variable since both will have similar data points, which is certainly not a right approach.

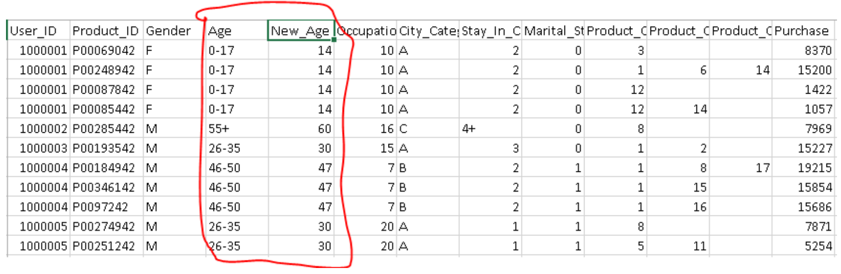
  - **Convert numeric bins to number:** Let’s say, bins of a continuous variable are available in the data set (shown below).



User_ID	Product_ID	Gender	Age	Occupatio	City_Cate	Stay_In_C	Marital_St	Product_C	Product_C	Product_C	Purchase
1000001	P00069042	F	0-17	10	A	2	0	3			8370
1000001	P00248942	F	0-17	10	A	2	0	1	6	14	15200
1000001	P00087842	F	0-17	10	A	2	0	12			1422
1000001	P00085442	F	0-17	10	A	2	0	12	14		1057
1000002	P00285442	M	55+	16	C	4+	0	8			7969
1000003	P00193542	M	26-35	15	A	3	0	1	2		15227
1000004	P00184942	M	46-50	7	B	2	1	1	8	17	19215
1000004	P00346142	M	46-50	7	B	2	1	1	15		15854
1000004	P0097242	M	46-50	7	B	2	1	1	16		15686
1000005	P00274942	M	26-35	20	A	1	1	8			7871
1000005	P00251242	M	26-35	20	A	1	1	5	11		5254

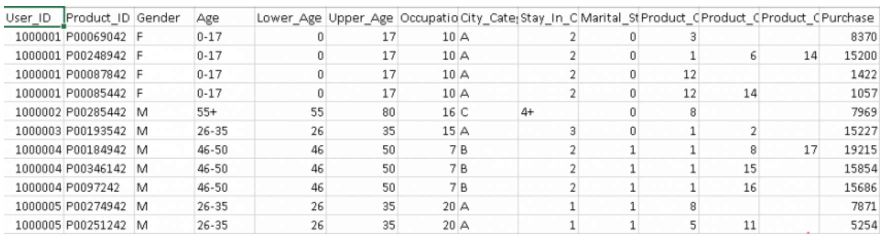
Above, you can see that variable “Age” has bins (0-17, 17-25, 26-35 …). We can convert these bins into definite numbers using the following methods:

    - Using label encoder for conversion. But, these numerical bins will be treated same as multiple levels of non-numeric feature. Hence, wouldn’t provide any additional information
    - Create a new feature using mean or mode (most relevant value) of each age bucket. It would comprise of additional weight for levels.



User_ID	Product_ID	Gender	Age	New_Age	Occupatio	City_Cate	Stay_In_C	Marital_St	Product_C	Product_C	Product_C	Purchase
1000001	P00069042	F	0-17	14	10	A	2	0	3			8370
1000001	P00248942	F	0-17	14	10	A	2	0	1	6	14	15200
1000001	P00087842	F	0-17	14	10	A	2	0	12			1422
1000001	P00085442	F	0-17	14	10	A	2	0	12	14		1057
1000002	P00285442	M	55+	60	16	C	4+	0	8			7969
1000003	P00193542	M	26-35	30	15	A	3	0	1	2		15227
1000004	P00184942	M	46-50	47	7	B	2	1	1	8	17	19215
1000004	P00346142	M	46-50	47	7	B	2	1	1	15		15854
1000004	P0097242	M	46-50	47	7	B	2	1	1	16		15686
1000005	P00274942	M	26-35	30	20	A	1	1	8			7871
1000005	P00251242	M	26-35	30	20	A	1	1	5	11		5254

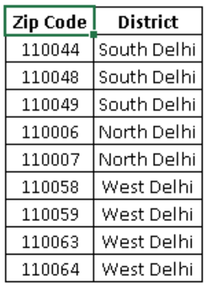
  - Create two new features, one for lower bound of age and another for upper bound. In this method, we'll obtain more information about these numerical bins compare to earlier two methods.



User_ID	Product_ID	Gender	Age	Lower_Age	Upper_Age	Occupatio	City_Cate	Stay_In_C	Marital_St	Product_C	Product_C	Product_C	Purchase
1000001	P00069042	F	0-17	0	17	10	A	2	0	3			8370
1000001	P00248942	F	0-17	0	17	10	A	2	0	1	6	14	15200
1000001	P00087842	F	0-17	0	17	10	A	2	0	12			1422
1000001	P00085442	F	0-17	0	17	10	A	2	0	12	14		1057
1000002	P00285442	M	55+	55	80	16	C	4+	0	8			7969
1000003	P00193542	M	26-35	26	35	15	A	3	0	1	2		15227
1000004	P00184942	M	46-50	46	50	7	B	2	1	1	8	17	19215
1000004	P00346142	M	46-50	46	50	7	B	2	1	1	15		15854
1000004	P0097242	M	46-50	46	50	7	B	2	1	1	16		15686
1000005	P00274942	M	26-35	26	35	20	A	1	1	8			7871
1000005	P00251242	M	26-35	26	35	20	A	1	1	5	11		5254

Combine Levels

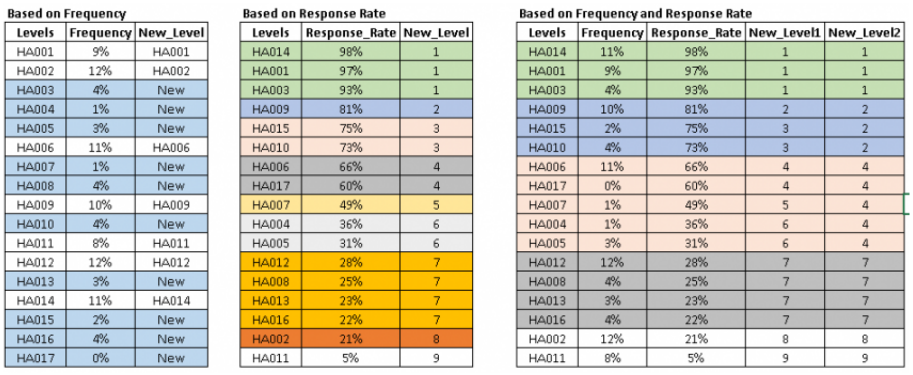
- **Combine levels:** To avoid redundant levels in a categorical variable and to deal with rare levels, we can simply combine the different levels. There are various methods of combining levels. Here are commonly used ones:
  - **Using Business Logic:** It is one of the most effective method of combining levels. It makes sense also to combine similar levels into similar groups based on domain or business experience. For example, we can combine levels of a variable "zip code" at state or district level. This will reduce the number of levels and improve the model performance also.



Zip Code	District
110044	South Delhi
110048	South Delhi
110049	South Delhi
110006	North Delhi
110007	North Delhi
110058	West Delhi
110059	West Delhi
110063	West Delhi
110064	West Delhi

  - **Using frequency or response rate:** Combining levels based on business logic is effective but we may always not have the domain knowledge. Imagine, you are given a data set from Aerospace Department, US Govt. How would you apply business logic here? In such cases, we combine levels by considering the frequency distribution or response rate.
    - To combine levels using their frequency, we first look at the frequency distribution of of each level and combine levels having frequency less than 5% of total observation (5% is standard but you can change it based on distribution). This is an effective method to deal with rare levels.
    - We can also combine levels by considering the response rate of each level. We can simply combine levels having similar response rate into same group.

- Finally, you can also look at both frequency and response rate to combine levels. You first combine levels based on response rate then combine rare levels to relevant group.



Based on Frequency

Levels	Frequency	New_Level
HA001	9%	HA001
HA002	12%	HA002
HA003	4%	New
HA004	1%	New
HA005	3%	New
HA006	11%	HA006
HA007	1%	New
HA008	4%	New
HA009	10%	HA009
HA010	4%	New
HA011	8%	HA011
HA012	12%	HA012
HA013	3%	New
HA014	11%	HA014
HA015	2%	New
HA016	4%	New
HA017	0%	New

Based on Response Rate

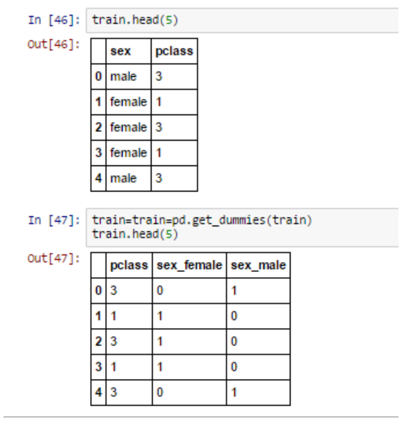
Levels	Response_Rate	New_Level
HA014	98%	1
HA001	97%	1
HA003	93%	1
HA009	81%	2
HA015	75%	3
HA010	73%	3
HA006	66%	4
HA017	60%	4
HA007	49%	5
HA004	36%	6
HA005	31%	6
HA012	28%	7
HA008	25%	7
HA013	23%	7
HA016	22%	7
HA002	21%	8
HA011	5%	9

Based on Frequency and Response Rate

Levels	Frequency	Response_Rate	New_Level1	New_Level2
HA014	11%	98%	1	1
HA001	9%	97%	1	1
HA003	4%	93%	1	1
HA009	10%	81%	2	2
HA015	2%	75%	3	2
HA010	4%	73%	3	2
HA006	11%	66%	4	4
HA017	0%	60%	4	4
HA007	1%	49%	5	4
HA004	1%	36%	6	4
HA005	3%	31%	6	4
HA012	12%	28%	7	7
HA008	4%	25%	7	7
HA013	3%	23%	7	7
HA016	4%	22%	7	7
HA002	12%	21%	8	8
HA011	8%	5%	9	9

Dummy Coding

- **Dummy Coding:** Dummy coding is a commonly used method for converting a categorical input variable into continuous variable. 'Dummy', as the name suggests is a duplicate variable which represents one level of a categorical variable. Presence of a level is represent by 1 and absence is represented by 0. For every level present, one dummy variable will be created. Look at the representation below to convert a categorical variable using dummy variable.



```
In [46]: train.head(5)
```

Out[46]:

	sex	pclass
0	male	3
1	female	1
2	female	3
3	female	1
4	male	3

```
In [47]: train=train=pd.get_dummies(train)
         train.head(5)
```

Out[47]:

	pclass	sex_female	sex_male
0	3	0	1
1	1	1	0
2	3	1	0
3	1	1	0
4	3	0	1

Note: Assume, we have 500 levels in categorical variables. Then, should we create 500 dummy variables? If you can automate it, very well. Or else, I'd suggest you to first, reduce the levels by using combining methods and then use dummy coding. This would save your time.This method is also known as "One Hot Encoding".

End Notes

In this article, we discussed the challenges you might face while dealing with categorical variable in modelling. We also discussed various methods to overcome those challenge and improve model performance. I've used Python for demonstration purpose and kept the focus of article for beginners.

In order to keep article simple and focused towards beginners, I have not described advanced methods like "feature hashing". I will take it up as a separate article in itself in future.

You must understand that these methods are subject to the data sets in question. I've seen even the most powerful methods failing to bring model improvement. Whereas, a basic approach can do wonders. Hence, you must understand the validity of these models in context to your data set. If you still face any trouble, I shall help you out in comments section below.

Did you find this article helpful ? Do you know of other methods which work well with categorical variables? Please share your thoughts in the comments section below. I'd love to hear you.

If you like what you just read & want to continue your analytics learning, subscribe to our emails, follow us on twitter or like our facebook page.

Article Url - https://www.analyticsvidhya.com/blog/2015/11/easy-methods-deal-categorical-variables-predictive-modeling/

Sunil Ray

I am a Business Analytics and Intelligence professional with deep experience in the Indian Insurance industry. I have worked for various multi-national Insurance companies in last 7 years.