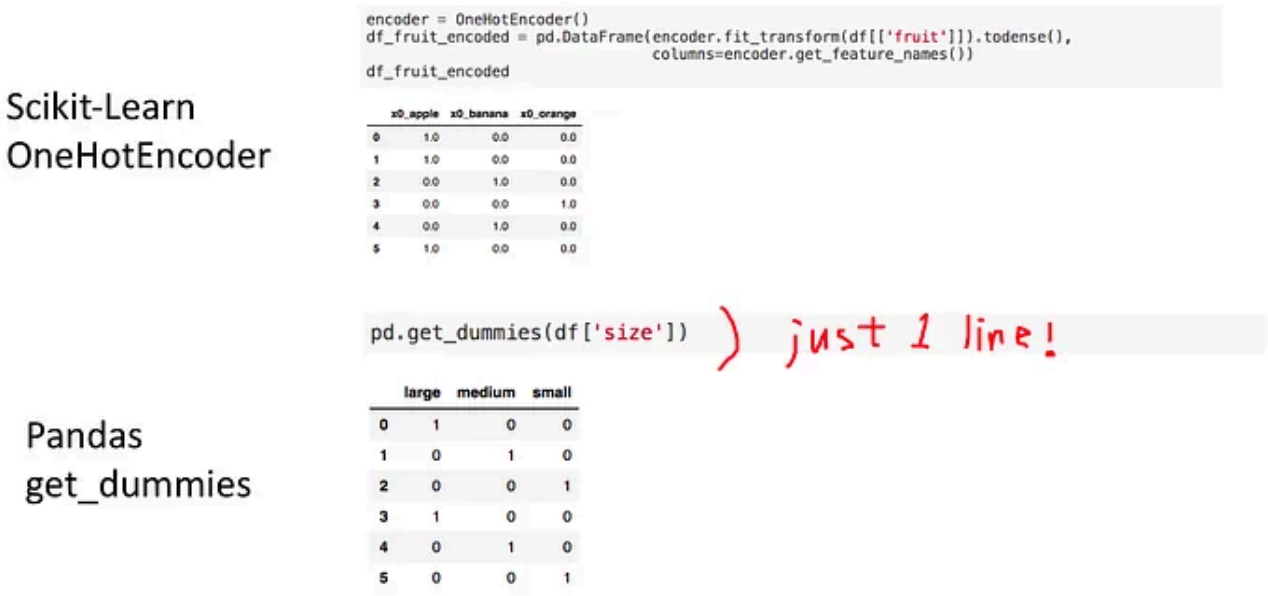
Encoding with Pandas get_dummies

Hasan Ersan YAĞCI · Follow
6 min read · Jan 24, 2021

We previously covered the issue of encoding and its importance. In short, machine learning models are mathematical models that use algorithms that work with numerical data types, and neural networks also work with numerical data types. Therefore, we need encoding methods to convert non-numerical data to meaningful numerical data. We have covered the encoding methods and the options that we can apply these encoding methods at this link.

Label Encoding vs One Hot Encoding

In this story, we will look at the Pandas get_dummies method. Pandas get_dummies is the easiest way to implement one hot encoding method and it has very useful parameters, of which we will mention the most important ones. With get_dummies we can get a hot encoder data frame (dummy variables) in one row.



Two Methods of One Hot Encoding

We will use car_price data for this demo. You can download data and notebook from this link.

The purpose of this data science process is to predict the prices of cars. We will use Linear Regression for this data, but the data is not ready for the machine learning model. Because there are categorical (non-numerical) columns and we need to transform them. For this, we will implement get_dummies.

Before diving into Get_dummies, let's check the data.

```
import pandas as pd
import numpy as np
```

```
df = pd.read_csv('car_price.csv')
```

```
df.head()
```

	make_model	body_type	Body Color	km	hp	Gearing Type	Extras	price
0	Audi A1	Sedans	Black	56013	85	Automatic	Alloy wheels,Catalytic Converter,Voice Control	15770
1	Audi A1	Sedans	Red	80000	85	Automatic	Alloy wheels,Sport seats,Sport suspension,Voic...	14500
2	Audi A1	Sedans	Black	83450	85	Automatic	Alloy wheels,Voice Control	14640
3	Audi A1	Sedans	Brown	73000	85	Automatic	Alloy wheels,Sport seats,Voice Control	14500
4	Audi A1	Sedans	Black	16200	85	Automatic	Alloy wheels,Sport package,Sport suspension,Vo...	16790

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4800 entries, 0 to 4799
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   make_model    4800 non-null   object
 1   body_type     4800 non-null   object
 2   Body Color    4800 non-null   object
 3   km            4800 non-null   int64
 4   hp            4800 non-null   int64
 5   Gearing Type  4800 non-null   object
 6   Extras        4800 non-null   object
 7   price         4800 non-null   int64
dtypes: int64(3), object(5)
memory usage: 300.1+ KB
```

As you can see, the data has no missing values and I acquired this structure after handling outliers. You can check this link to see how to detect and handle outliers with pandas.

We have 5 categorical columns that we need to make numerical before machine learning algorithm. These columns are; “make_model”, “body_type”, “Body Color”, “Gear Type” and “Extras”, all of them are nominal data. For example, for “Body Color” there is no hierarchy between colors. Black has no superiority over red. That’s why we have to use one hot encoder.

Firstly we will just apply get_dummies to the “Body Color” column to see details of get_dummies, then we will use all dataframe;

1 — get_dummies()

```
df['Body Color'].unique()
```

```
array(['Black', 'Red', 'Brown', 'White', 'Grey', 'Blue', 'Silver',
       'Beige', 'Violet', 'Yellow', 'Green', 'Bronze', 'Orange'],
      dtype=object)
```

```
df['Body Color'].nunique()
```

```
13
```

The “Body Color” column has 13 unique values, which means we will get 13 columns after applying get_dummies.

With this syntax we can apply get_dummies to a column of dataframe;

```
pd.get_dummies(df['Body Color'])
```

```
pd.get_dummies(df['Body Color'])
```

	Beige	Black	Blue	Bronze	Brown	Green	Grey	Orange	Red	Silver	Violet	White	Yellow
0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	1	0	0	0	0
2	0	1	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	1	0	0	0	0	0	0	0	0
4	0	1	0	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
4795	0	0	0	0	0	0	0	0	0	0	0	1	0
4796	0	0	0	0	0	0	0	0	0	0	0	1	0
4797	0	0	0	0	0	0	0	0	0	1	0	0	0
4798	0	0	0	0	0	0	0	0	0	1	0	0	0
4799	0	0	0	0	0	0	0	0	0	1	0	0	0

4800 rows × 13 columns

We didn’t use any parameters, get_dummies has default parameters. As you can see we got a 13 column dataframe after get_dummies. There are 4800 rows. We can see the color of the car. For example; the first car color black, the second car color red, etc.

Our original data frame, df, keeps its shape. We must merge these dataframes. Now we can assign “Body Color”s dummy variables (one hot encoder) to a new dataframe to merge with the main dataframe.

```
df_colors = pd.get_dummies(df['Body Color'])
```

```
df.head()
```

	make_model	body_type	Body Color	km	hp	Gearing Type	Extras	price
0	Audi A1	Sedans	Black	56013	85	Automatic	Alloy wheels,Catalytic Converter,Voice Control	15770
1	Audi A1	Sedans	Red	80000	85	Automatic	Alloy wheels,Sport seats,Sport suspension,Voic...	14500
2	Audi A1	Sedans	Black	83450	85	Automatic	Alloy wheels,Voice Control	14640
3	Audi A1	Sedans	Brown	73000	85	Automatic	Alloy wheels,Sport seats,Voice Control	14500
4	Audi A1	Sedans	Black	16200	85	Automatic	Alloy wheels,Sport package,Sport suspension,Vo...	16790

```
df_colors.head()
```

	Beige	Black	Blue	Bronze	Brown	Green	Grey	Orange	Red	Silver	Violet	White	Yellow
0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	1	0	0	0	0
2	0	1	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	1	0	0	0	0	0	0	0	0
4	0	1	0	0	0	0	0	0	0	0	0	0	0

We will now merge them into data frames using the join method. But we should drop the “Body Color” column, we don’t need it anymore because it’s not numerical.

```
df = (df.drop(['Body Color'], axis = 1)).join(df_colors)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4800 entries, 0 to 4799
Data columns (total 20 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   make_model    4800 non-null   object
 1   body_type     4800 non-null   object
 2   km            4800 non-null   int64
 3   hp            4800 non-null   int64
 4   Gearing Type  4800 non-null   object
 5   Extras        4800 non-null   object
 6   price         4800 non-null   int64
 7   Beige         4800 non-null   uint8
 8   Black         4800 non-null   uint8
 9   Blue          4800 non-null   uint8
 10  Bronze        4800 non-null   uint8
 11  Brown         4800 non-null   uint8
 12  Green         4800 non-null   uint8
 13  Grey          4800 non-null   uint8
 14  Orange        4800 non-null   uint8
 15  Red           4800 non-null   uint8
 16  Silver        4800 non-null   uint8
 17  Violet        4800 non-null   uint8
 18  White         4800 non-null   uint8
 19  Yellow        4800 non-null   uint8
dtypes: int64(3), object(4), uint8(13)
memory usage: 323.6+ KB
```

With this syntax, we dropped the “Body Color” column and added our dummy variables. As you can see, all colors have a column and column types are numerical.

2 — get_dummies with 'drop_first' parameter

One column transformed into 13 columns. We can use the 'drop_first' parameter and decrease one column. We can take 12 columns. Normally the default value of this parameter is 'False', we just set it to 'True'. Let's see how it works.

```
pd.get_dummies(df['Body Color'], drop_first = True)
```

```
1 pd.get_dummies(df['Body Color'])
```

	Beige	Black	Blue	Bronze	Brown	Green	Grey	Orange	Red	Silver	Violet	White	Yellow
0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	1	0	0	0	0
2	0	1	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	1	0	0	0	0	0	0	0	0
4	0	1	0	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
4795	0	0	0	0	0	0	0	0	0	0	0	1	0
4796	0	0	0	0	0	0	0	0	0	0	0	1	0
4797	0	0	0	0	0	0	0	0	0	1	0	0	0
4798	0	0	0	0	0	0	0	0	0	1	0	0	0
4799	0	0	0	0	0	0	0	0	0	1	0	0	0

4800 rows × 13 columns

```
1 pd.get_dummies(df['Body Color'], drop_first = True)
```

	Black	Blue	Bronze	Brown	Green	Grey	Orange	Red	Silver	Violet	White	Yellow
0	1	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	1	0	0	0	0	0	0	0	0
4	1	0	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...
4795	0	0	0	0	0	0	0	0	0	0	1	0
4796	0	0	0	0	0	0	0	0	0	0	1	0
4797	0	0	0	0	0	0	0	0	1	0	0	0
4798	0	0	0	0	0	0	0	0	1	0	0	0
4799	0	0	0	0	0	0	0	0	1	0	0	0

4800 rows × 12 columns

Check the number of columns, instead of 13 we got 12 columns.

It removes the first column of the get_dummies dataframe. The first column for the "Body Color" column is Beige. If there is a beige car, all columns are 0. When all columns are 0, the model knows it's a beige car. Check out the example below.

```
pd.get_dummies(df['Body Color']).loc[[133]]
```

	Beige	Black	Blue	Bronze	Brown	Green	Grey	Orange	Red	Silver	Violet	White	Yellow
133	1	0	0	0	0	0	0	0	0	0	0	0	0

```
pd.get_dummies(df['Body Color'], drop_first = True).loc[[133]]
```

	Black	Blue	Bronze	Brown	Green	Grey	Orange	Red	Silver	Violet	White	Yellow
133	0	0	0	0	0	0	0	0	0	0	0	0

More columns mean less performance and more training time. Imagine we have 20 columns that are not numerical. If we use ‘drop_first’, we get 20 columns less. So it is useful to use the drop_first parameter for model performance.

3 — get_dummies with ‘prefix’ parameter

If the dataframe had the “Upholstery Color” column, we would also get a black or a brown column for the tile color after get_dummies except “Body Color”. Multiple columns with the same name can cause problems. We can use the ‘prefix’ parameter to avoid this situation.

```
pd.get_dummies(df['Body Color'], drop_first = True, prefix = 'BC')
```

```
pd.get_dummies(df['Body Color'], drop_first = True, prefix = 'BC')
```

	BC_Black	BC_Blue	BC_Bronze	BC_Brown	BC_Green	BC_Grey	BC_Orange	BC_Red	BC_Silver	BC_Violet	BC_White	BC_Yellow
0	1	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	1	0	0	0	0	0	0	0	0
4	1	0	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...
4795	0	0	0	0	0	0	0	0	0	0	1	0
4796	0	0	0	0	0	0	0	0	0	0	1	0
4797	0	0	0	0	0	0	0	0	1	0	0	0
4798	0	0	0	0	0	0	0	0	1	0	0	0
4799	0	0	0	0	0	0	0	0	1	0	0	0

4800 rows × 12 columns

This parameter adds the word as a prefix with an underscore. For the example above, we have used the BC prefix.

4— get_dummies with 'columns' parameter

We can apply get_dummies directly to a dataframe instead of applying it individually. It will automatically add the column name as a prefix for each dummy variable.

```
pd.get_dummies(df, drop_first = True)
```

```
1 pd.get_dummies(df, drop_first = True)
```

	km	hp	price	make_model_Audi A2	make_model_Audi A3	body_type_Convertible	body_type_Coupe	body_type_Off-Road	body_type_Other	body_type_Seda
0	56013	85	15770	0	0	0	0	0	0	
1	80000	85	14500	0	0	0	0	0	0	
2	83450	85	14640	0	0	0	0	0	0	
3	73000	85	14500	0	0	0	0	0	0	
4	16200	85	16790	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	
4795	54	85	25000	0	1	0	0	0	0	
4796	50	85	24980	0	1	0	0	0	0	
4797	6666	85	24980	0	1	0	0	0	0	
4798	10	85	24980	0	1	0	0	0	0	
4799	10	85	24980	0	1	0	0	0	0	

4800 rows × 349 columns

But the dataframe evolved to 349 columns because the "Extras" column has many unique values. We have to consider this column individually. In this case, we can use the columns parameter.

```
pd.get_dummies(df, columns = ['make_model', 'body_type', 'Body
Color', 'Gearing Type'], drop_first = True)
```

```
pd.get_dummies(df, columns = ['make_model', 'body_type', 'Body Color', 'Gearing Type'], drop_first = True)
```

	km	hp	Extras	price	make_model_Audi A2	make_model_Audi A3	body_type_Convertible	body_type_Coupe	body_type_Off-Road	body_type_Other
0	56013	85	Alloy wheels,Catalytic Converter,Voice Control	15770	0	0	0	0	0	0
1	80000	85	Alloy wheels,Sport seats,Sport suspension,Voic...	14500	0	0	0	0	0	0
2	83450	85	Alloy wheels,Voice Control	14640	0	0	0	0	0	0
3	73000	85	Alloy wheels,Sport seats,Voice Control	14500	0	0	0	0	0	0
4	16200	85	Alloy wheels,Sport package,Sport suspension,Vo...	16790	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...
4795	54	85	Alloy wheels,Sport seats,Sport suspension,Voic...	25000	0	1	0	0	0	0
4796	50	85	Alloy wheels	24980	0	1	0	0	0	0
4797	6666	85	Alloy wheels,Roof rack	24980	0	1	0	0	0	0
4798	10	85	Alloy wheels,Roof rack	24980	0	1	0	0	0	0
4799	10	85	Alloy wheels,Roof rack	24980	0	1	0	0	0	0

4800 rows × 26 columns

With the columns parameter, we can apply the columns we want. So we can handle custom columns later with str.get_dummies.

4 — str.get_dummies

The str.get_dummies method is a version of get_dummies that can be applied to a series. It is a string-handling version. The str.get_dummies () method divides each string in the given series with the separator. There is only one separator parameter.

In this dataframe, there are some features for cars in the “Extras” column. For example; alloy wheels, sports seats, voice control, etc. These features are important in terms of price. Check below, it says there are 325 unique values. However, there are 16 extra features. Cars have a different number of extra features.

```
df['Extras'].unique()
```

```
array(['Alloy wheels,Catalytic Converter,Voice Control',
       'Alloy wheels,Sport seats,Sport suspension,Voice Control',
       'Alloy wheels,Voice Control',
       'Alloy wheels,Sport seats,Voice Control',
       'Alloy wheels,Sport package,Sport suspension,Voice Control',
       'Alloy wheels,Sport package,Sport seats,Sport suspension',
       'Alloy wheels', 'Alloy wheels,Shift paddles',
       'Alloy wheels,Catalytic Converter,Sport package,Sport seats,Sport suspension,Voice Control',
       'Alloy wheels,Sport seats,Sport suspension',
       'Alloy wheels,Sport package,Sport seats',
       'Alloy wheels,Sport package',
       'Alloy wheels,Catalytic Converter,Shift paddles,Voice Control',
       'Alloy wheels,Shift paddles,Sport package,Voice Control',
       'Alloy wheels,Catalytic Converter,Sport seats,Voice Control,Winter tyres',
       'Alloy wheels,Winter tyres', 'Alloy wheels,Catalytic Converter',
       'Catalytic Converter', 'Alloy wheels,Sport suspension',
       'Catalytic Converter,Sport suspension,Voice Control',
       'Alloy wheels,Sport seats',
       'Alloy wheels,Touch screen,Voice Control',
```

```
df['Extras'].nunique()
```

```
325
```

If we apply get_dummies directly, it adds 325 more columns. So we have to use str.get_dummies. We use a comma (,) as the separator because the values in the data are separated by commas (,).

```
df['Extras'].str.get_dummies(',')
```

```
df['Extras'].str.get_dummies(',')
```

	Alloy wheels	Cab or rented Car	Catalytic Converter	Handicapped enabled	Right hand drive	Roof rack	Shift paddles	Ski bag	Sport package	Sport seats	Sport suspension	Touch screen	Trailer hitch	Tuned car	Voice Control	Winter tyres
0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0
1	1	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0
2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
3	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0
4	1	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4795	1	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0
4796	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4797	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
4798	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
4799	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

4800 rows × 16 columns

As you can see, there are 16 extra features. Check for the first row. It has alloy wheels, catalytic converter, and sound control; these are 1, others are 0.

Now let’s get the latest version of the dataframe by merging them;

```
df = pd.get_dummies(df, columns = ['make_model', 'body_type', 'Body Color', 'Gearing Type'], drop_first = True)
```

```
df_extra = df['Extras'].str.get_dummies(',')
```

```
df_dummy = (df.drop(['Extras'], axis = 1)).join(df_extra)
```

```
df_dummy.shape
```

```
(4800, 41)
```

```
df_dummy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4800 entries, 0 to 4799
Data columns (total 41 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   km                            4800 non-null   int64
 1   hp                            4800 non-null   int64
 2   price                         4800 non-null   int64
 3   make_model_Audi A2            4800 non-null   uint8
 4   make_model_Audi A3            4800 non-null   uint8
 5   body_type_Convertible         4800 non-null   uint8
 6   body_type_Coupe               4800 non-null   uint8
 7   body_type_Off-Road            4800 non-null   uint8
 8   body_type_Other               4800 non-null   uint8
 9   body_type_Sedans              4800 non-null   uint8
 10  body_type_Station wagon       4800 non-null   uint8
 11  Body Color_Black              4800 non-null   uint8
 12  Body Color_Blue               4800 non-null   uint8
 13  Body Color_Bronze             4800 non-null   uint8
 14  Body Color_Brown              4800 non-null   uint8
 15  Body Color_Green              4800 non-null   uint8
 16  Body Color_Grey               4800 non-null   uint8
 17  Body Color_Orange             4800 non-null   uint8
 18  Body Color_Red                4800 non-null   uint8
 19  Body Color_Silver             4800 non-null   uint8
 20  Body Color_Violet             4800 non-null   uint8
 21  Body Color_White              4800 non-null   uint8
 22  Body Color_Yellow             4800 non-null   uint8
 23  Gearing Type_Manual           4800 non-null   uint8
 24  Gearing Type_Semi-automatic   4800 non-null   uint8
 25  Alloy wheels                  4800 non-null   int64
 26  Cab or rented Car             4800 non-null   int64
 27  Catalytic Converter           4800 non-null   int64
 28  Handicapped enabled           4800 non-null   int64
 29  Right hand drive              4800 non-null   int64
 30  Roof rack                     4800 non-null   int64
 31  Shift paddles                 4800 non-null   int64
 32  Ski bag                       4800 non-null   int64
 33  Sport package                 4800 non-null   int64
 34  Sport seats                   4800 non-null   int64
 35  Sport suspension              4800 non-null   int64
 36  Touch screen                  4800 non-null   int64
 37  Trailer hitch                 4800 non-null   int64
 38  Tuned car                     4800 non-null   int64
 39  Voice Control                 4800 non-null   int64
 40  Winter tyres                  4800 non-null   int64
dtypes: int64(19), uint8(22)
memory usage: 815.8 KB
```

All columns are numeric. Our data is now ready for the model.

Conclusion

Pandas get_dummies is the final touch on data before modeling. Because we have to make all non-numerical columns numerical. We have to use it or some other method to get the encoding tables / dummy variables. But as you can see, get_dummies is the easiest way and it has many parameters that make our model more readable and smoother.

Import the data, handle with missing values and outliers, then apply get_dummies. It is ready for the model.

Pandas | Data Science | Machine Learning | Data Analysis | Exploratory Data Analysis

Follow

Written by Hasan Ersan YAĞCI

190 Followers

Data Scientist

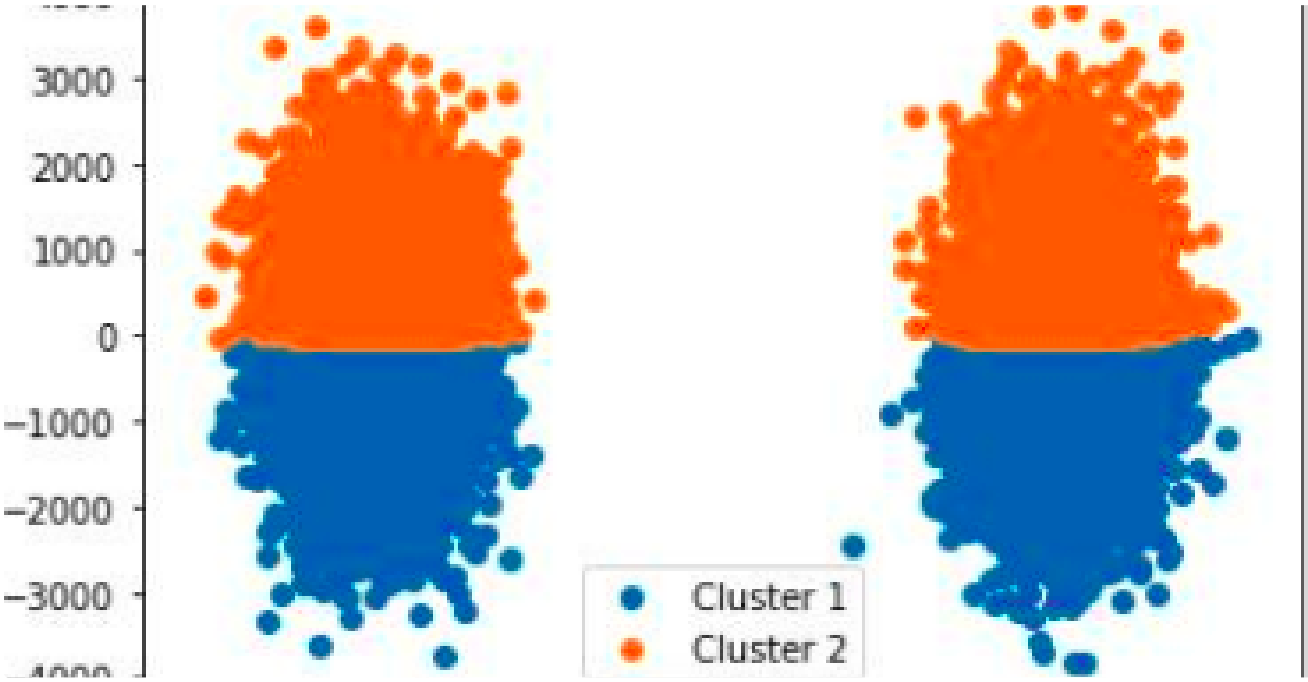
More from Hasan Ersan YAĞCI

Hasan Ersan YAĞCI

Detecting and Handling Outliers with Pandas

Data analysis is a long process. There are some steps to do this. First of all, we need to recognize the data. We have to know every...

Jan 15, 2021 815 9

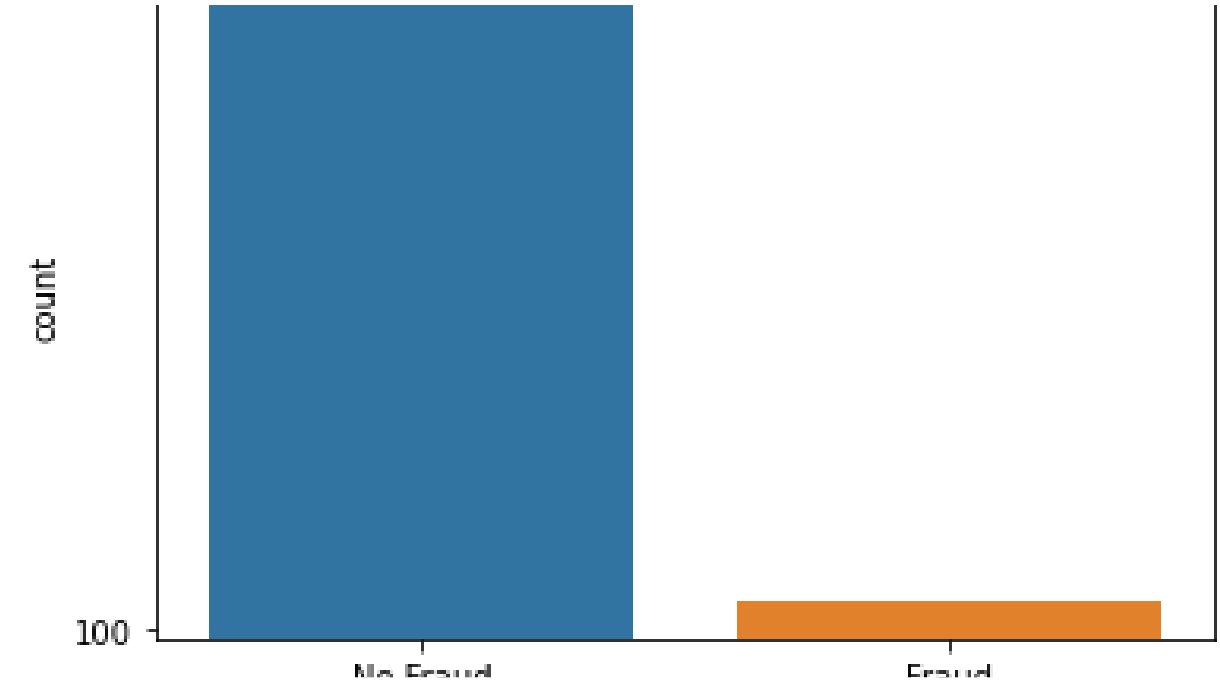


Hasan Ersan YAĞCI

Feature Scaling with Scikit-Learn for Data Science

In the data science process, we need to do some preprocessing before machine learning algorithms. These can be some basic data analysis...

Feb 22, 2021 491



Hasan Ersan YAĞCI

Under-Sampling Methods for Imbalanced Data (ClusterCentroids, RandomUnderSampler, NearMiss)

The imbalance of data is a big problem for classification tasks. In python, there is a library to allow to use of many algorithms to handle...

Jul 15, 2021 234

Hasan Ersan YAĞCI

Feature Selection with BorutaPy, RFE and Univariate Feature Selection

Feature selection is one of the important part of the machine learning pipeline. We may have to struggle with a lot of features or useless...

Apr 4, 2021 402

See all from Hasan Ersan YAĞCI

Recommended from Medium

Subha

How to handle categorical features?

If you are working with data then I'm 200% sure that you would have come across categorical features. Categorical data is nothing but...

Feb 8 1

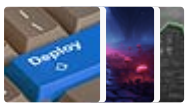
Jainvidip

Plot like a Pro: Matplotlib 101

Welcome to a full comprehensive guide that will guide you from no knowledge about matplotlib to a prodigy which will allow you to plot like...

Jun 6 10

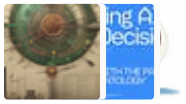
Lists



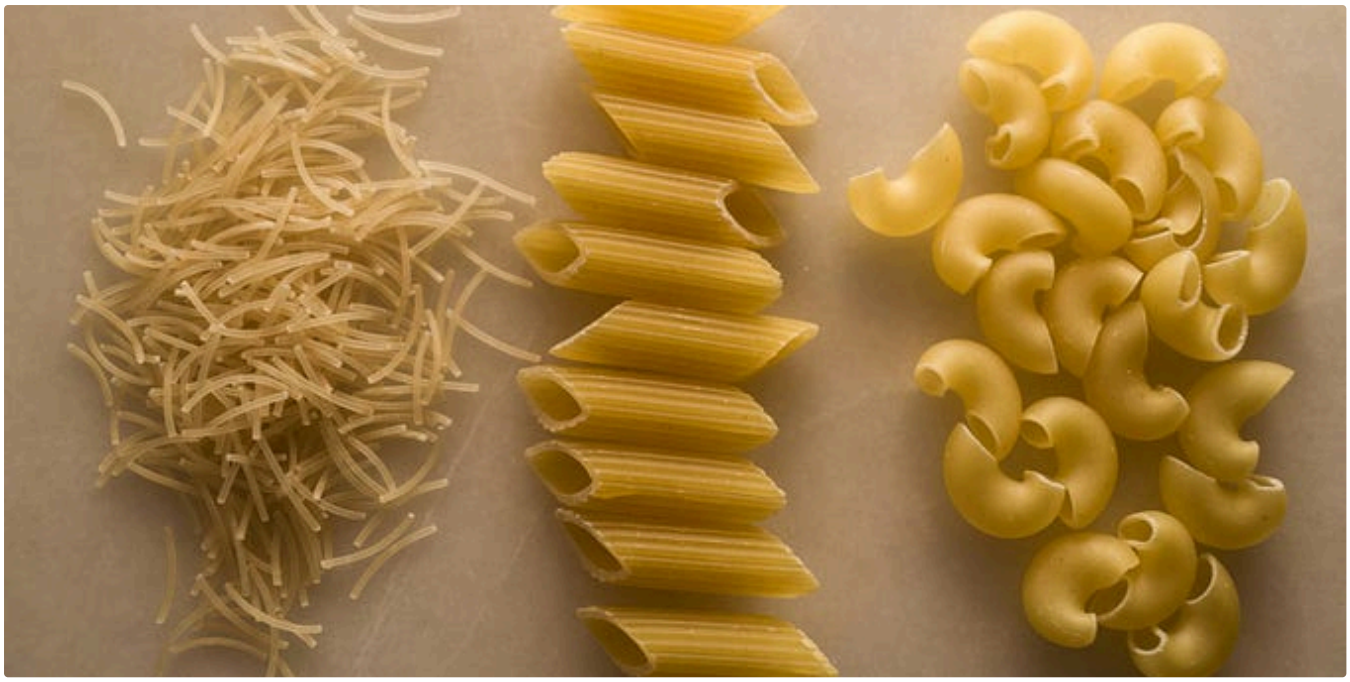
Predictive Modeling w/ Python
20 stories · 1359 saves

Practical Guides to Machine Learning
10 stories · 1638 saves

Natural Language Processing
1567 stories · 1112 saves



data science and AI
40 stories · 197 saves

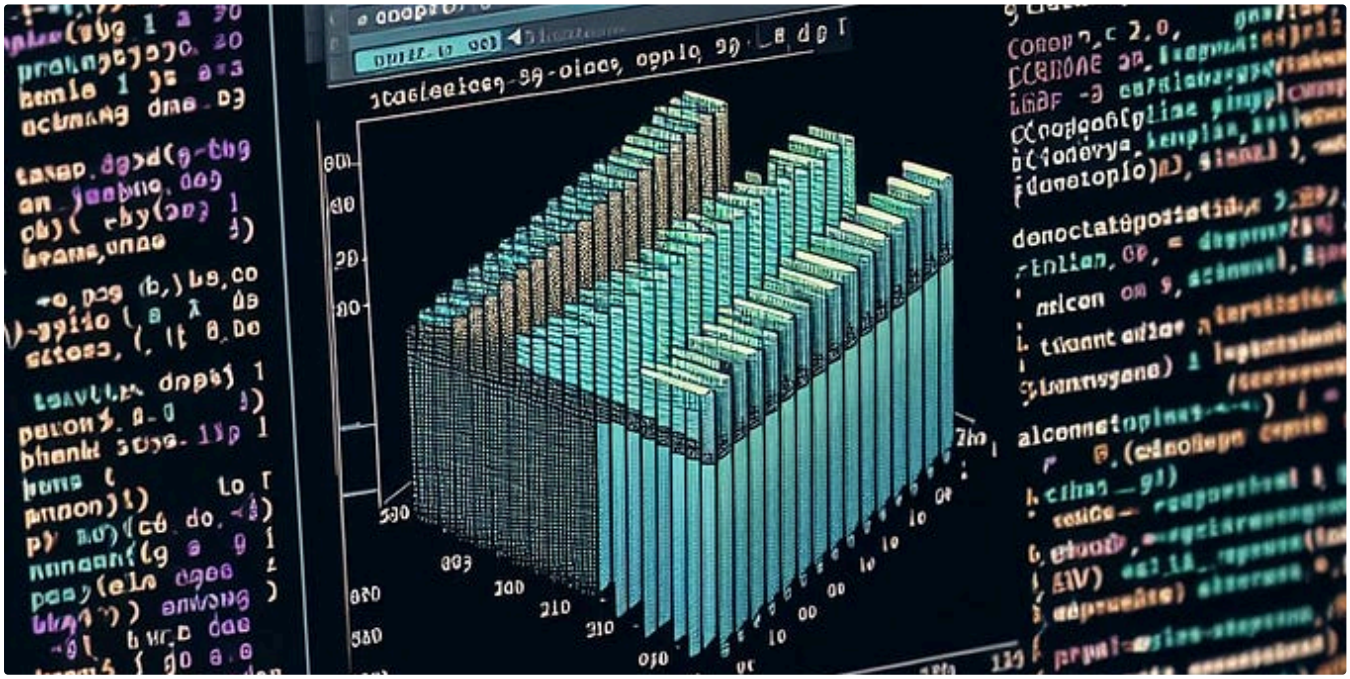


Juan Jose Munoz in Towards Data Science

Encoding Categorical Variables: A Deep Dive into Target Encoding

Data comes in different shapes and forms. One of those shapes and forms is known as categorical data.

Feb 5 500 4

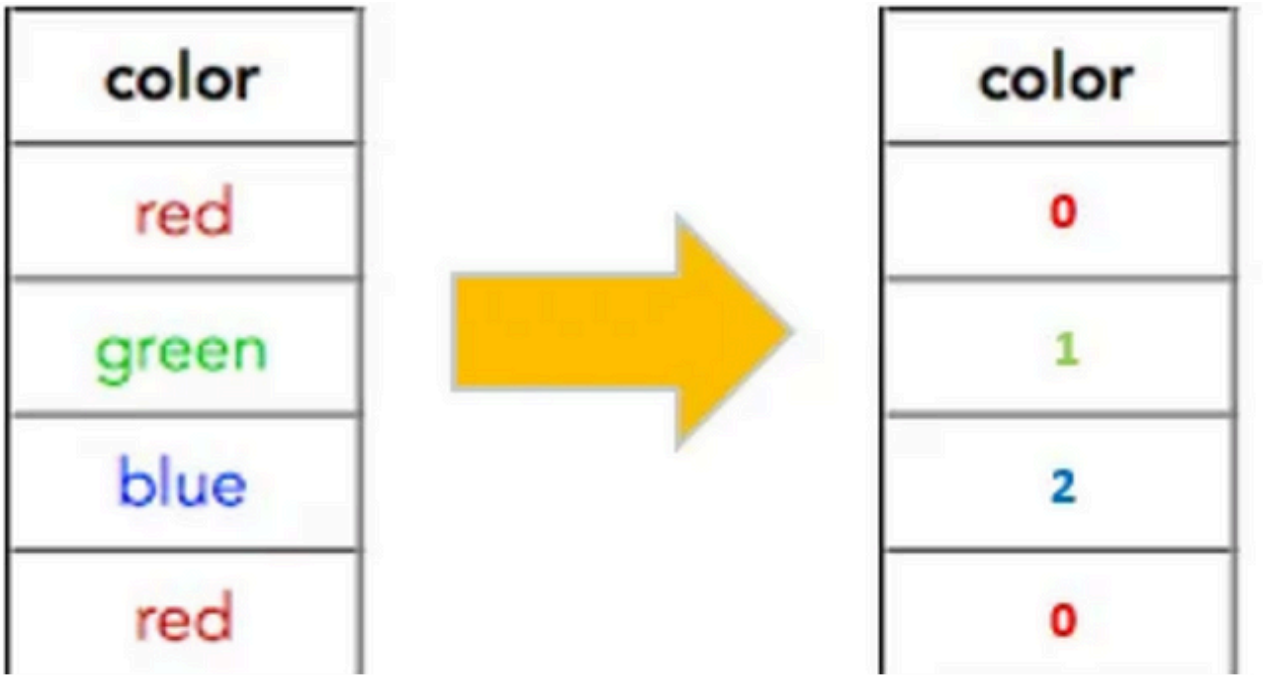


Omar

Identify missing values in each column with pandas.

The isnull().sum() method is a powerful tool for identifying missing values in each column of a pandas DataFrame. Missing values, also...

Feb 25

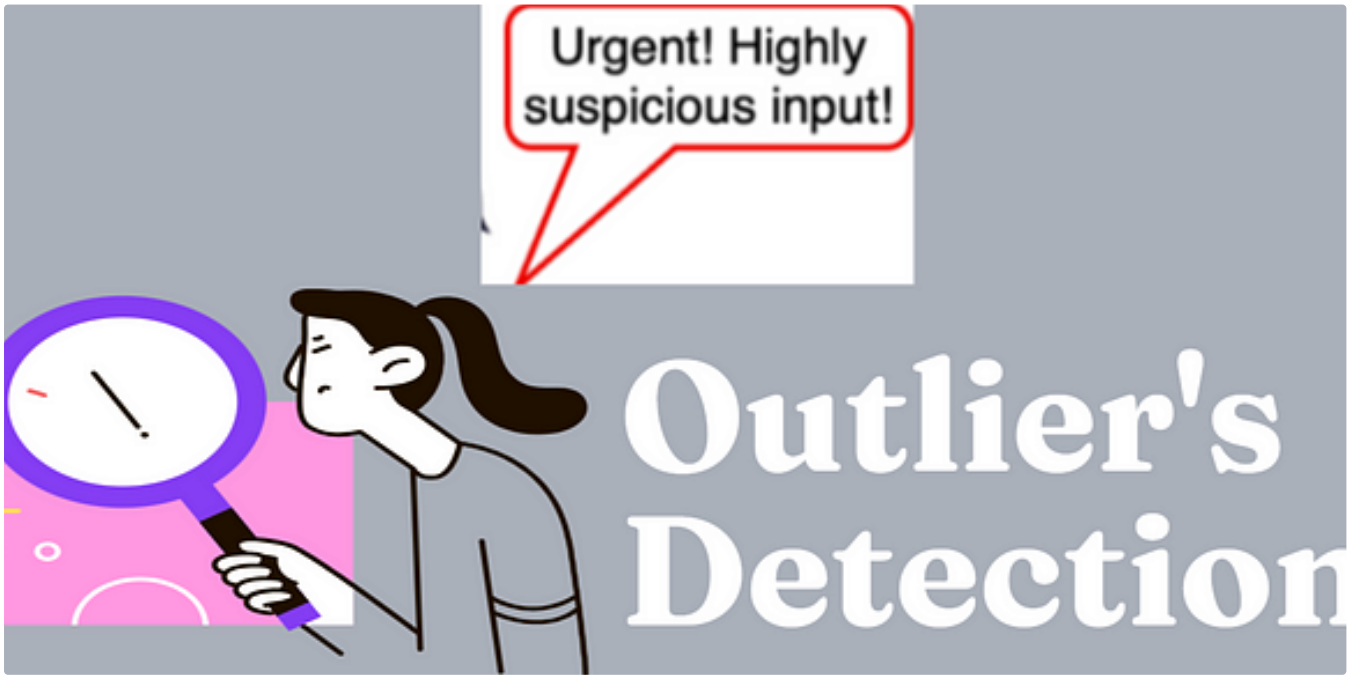


Sunny Kumar

What is label encoding? Application of label encoder in machine learning and deep learning models.

In the process of creating ML models we deal with datasets having multiple type of datatypes. There is wide range from numerical to...

Jan 13 5



Ayesha sidhikha

Outliers Detection

Outliers in a dataset are data points that significantly deviate from the majority of observations.

Jan 13 16

See more recommendations