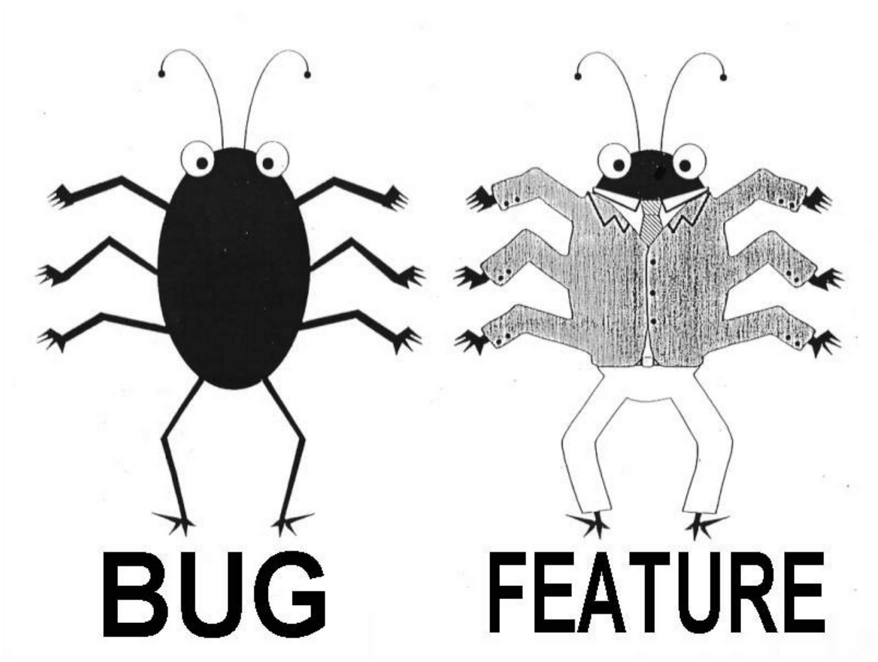
Getting Data ready for modelling: Feature engineering, Feature Selection, Dimension Reduction (Part 1)

Akash Desarda Follow
Dec 10, 2018 · 6 min read



Even in the fancy world of Machine Learning, like Humans can't ride their vehicles on the rough road properly, ML Algorithms too, can't produce the expected result on lots of unwanted miscellany data. So let's dive in and explore all our option for optimizing Data.

Note: This might be a bit on the lengthy side. So I am dividing it into three part series. So explore this part by part.

. . .

Part 1: Feature Engineering

What is feature Engineering and why to bother about it?

Feature engineering is the process of **using domain knowledge of the data to create features** that make machine learning algorithms work. Feature engineering is **fundamental to the application of machine**

learning and is both difficult and expensive. But when done properly, can result in wonders. It is the difference between the same algorithm working poorly and working awesome.

It is important and can't be ignored. Let's see it the general idea:

- **Data Cleaning & Preprocessing**

→ Handling outliers

→Handling missing values

→Handling skewness

- **Scaling**
- **Encoding**

1. ***Data Cleaning & Preprocessing:*** In the real world we never get tailored-made data perfect for an algorithm. We need to make it so, in the following ways.

i) Handling outliers: Outliers are the data point which doesn't follow the general trend of data. Many algorithms are sensitive towards outliers. So the question is what to do?

→ If there are few, so remove them completely. You can set a threshold value to identify them and then remove them. If a column has lots of outliers, it is better to completely remove the column, same goes with rows.

→ You can convert everything into 'Log form', as log brings everything on the same distance. (works on numeric data only though)

→ **Visualize the data** using scatterplots, histograms and box and whisker plots and look for extreme values. There are tons of other technique used to handle outliers, I suggest you go through them.

ii) Handling missing values: Why Missing Value Treatment?

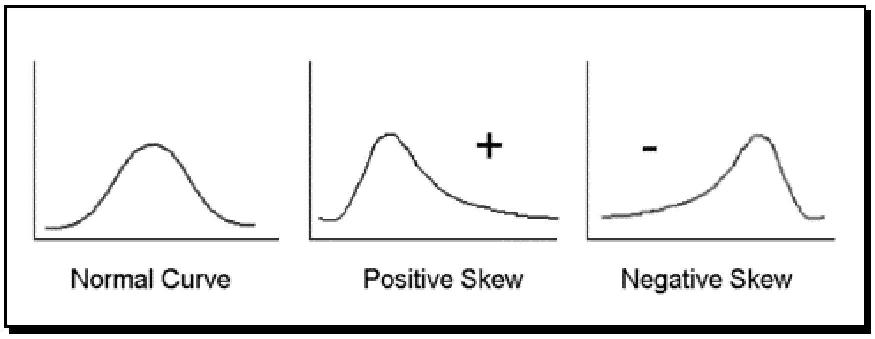
Missing data in the training data set can reduce the power/fit of a model. Missing values can lead to a biased model because we have not analyzed the behaviour and relationship with other variables correctly. This is useful because **some algorithms are unable to work with or exploit missing data.** Therefore, it is important to identify and mark this missing data. Once marked, replacement values can be prepared.

→ Replacing missing value either mean, median, mode (it totally depends on the judgement call). You can use **sklearn.preprocessing.impute** for the same purpose.

→ If needed you can replace with entirely new data (again judgement call).

→ Or you can remove the complete column if there are just too many missing values. So this needs mostly judgement call, again !!!

iii) Skewness: Skewness is a measure of the asymmetry of a distribution. Skewness is a measure of symmetry, or more precisely, the lack of symmetry.



The basic three types of Skewness

Why handle Skewness?

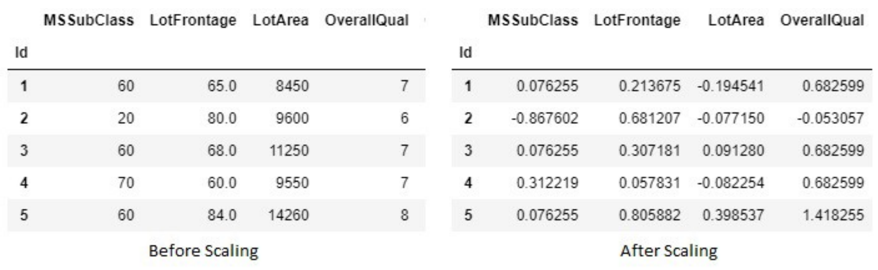
→ Many model building techniques have the assumption that predictor values are distributed normally and have a symmetrical shape. Hence, it is sometimes paramount to deal with skewness.

→ Symmetric distribution is preferred over skewed distribution as it is easier to interpret and generate inferences.

→ log transformation, square root transformation, etc are used.

*2. **Scaling:*** Like Handling missing values is kind of compulsory, scaling is not. But that does not mean it is less important technique. Think of a scenario where, *one of your column (say A)has values in the range of 10k to 100k and one column has a value of the range of 0 to 1(say B), then A will have an inappropriate advantage over B as it will carry more weight.*

→ Scaling modifies features to lie between a given minimum and maximum value, often between zero and one, or so that the maximum absolute value of each feature is scaled to unit size to improve the numerical stability of some models.



Id	MSSubClass	LotFrontage	LotArea	OverallQual
1	60	65.0	8450	7
2	20	80.0	9600	6
3	60	68.0	11250	7
4	70	60.0	9550	7
5	60	84.0	14260	8

Before Scaling

Id	MSSubClass	LotFrontage	LotArea	OverallQual
1	0.076255	0.213675	-0.194541	0.682599
2	-0.867602	0.681207	-0.077150	-0.053057
3	0.076255	0.307181	0.091280	0.682599
4	0.312219	0.057831	-0.082254	0.682599
5	0.076255	0.805882	0.398537	1.418255

After Scaling

Effect Of Scaling

→ But Standardization/Scaling cannot be applied to categorical data, hence we split the categorical and numerical data in order to standardize the numerical data.

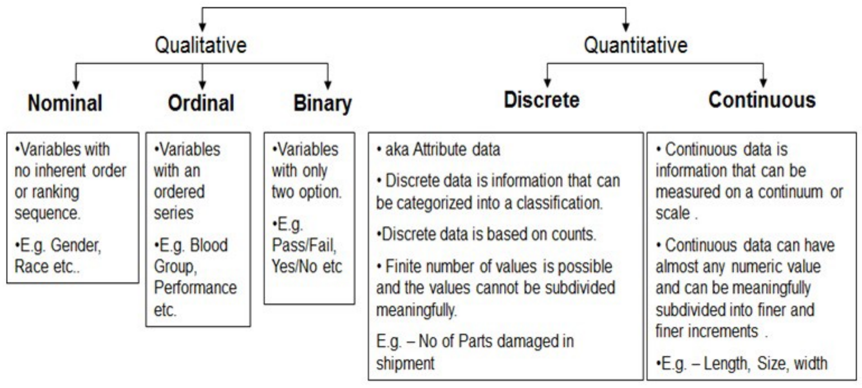
→ *MinMax scaler, Standard Scler, Normalizer* etc are some techniques. All can be performed using **'sklearn.preprocessing.scaler'**

(I suggest you visit this blog to study Scaling in more in-depth)

3. Encoding: So, What and Why is Encoding?

Most algorithms we use work with numerical values whereas more often than not categorical data are in text/string (male, female) or bin (0–4, 4–8 etc.) form.

One option is to leave these variables out of the algorithms and use only numeric data. But in doing so we can lose out on some critical information. Hence, it is generally a good idea to include the categorical variables into your algorithms by encoding them to convert to numeric values but, first let's learn a thing or two about our categorical variables.



Nominal
- •Variables with no inherent order or ranking sequence.
- •E.g. Gender, Race etc..

Ordinal
- •Variables with an ordered series
- •E.g. Blood Group, Performance etc.

Binary
- •Variables with only two option.
- •E.g. Pass/Fail, Yes/No etc

Discrete
- • aka Attribute data
- • Discrete data is information that can be categorized into a classification.
- •Discrete data is based on counts.
- • Finite number of values is possible and the values cannot be subdivided meaningfully.
- E.g. – No of Parts damaged in shipment

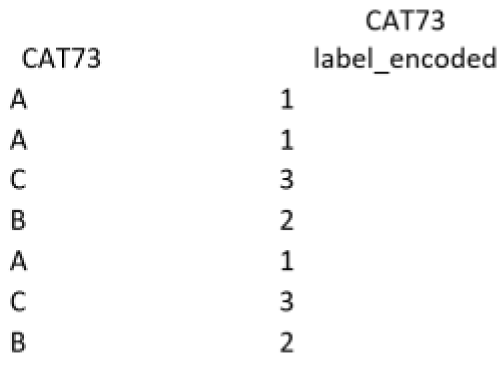
Continuous
- • Continuous data is information that can be measured on a continuum or scale .
- • Continuous data can have almost any numeric value and can be meaningfully subdivided into finer and finer increments .
- •E.g. – Length, Size, width

Types of Variable

There are generally two types of encoding done on data, Label Encoding & One hot Encoding (or pandas.get_dummies).

i) Label Encoding: Each category is given one label (e.g. 0, 1, 2 etc). Label encoding is a handy technique to encode categorical variables. However, such encoded nominal variables might end up in being misinterpreted as ordinal. **So Label Encoding is done on only Ordinal types of data (which have some sense of order).**

→So even after Label Encoding, all the data will not lose it ranking or level of importance.



CAT73	CAT73 label_encoded
A	1
A	1
C	3
B	2
A	1
C	3
B	2

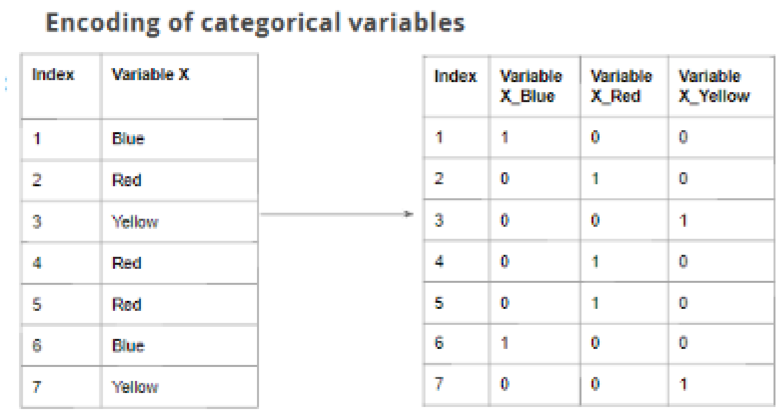
eg of Label Encoding

Can be performed using **‘sklearn.preprocessing.LabelEncoder’**

ii) One Hot Encoding: Label encoding cannot be performed on Nominal or Binary, as we can’t rank them based on their property. Every data is treated equally. Consider following two categorical variables and their values as eg

→ *Colour*: Blue, Green, Red, Yellow

→ *Educational Qualification*: Primary School, Secondary School, Graduate, Post-Graduate, Phd.



Encoding of categorical variables

Index	Variable X
1	Blue
2	Red
3	Yellow
4	Red
5	Red
6	Blue
7	Yellow

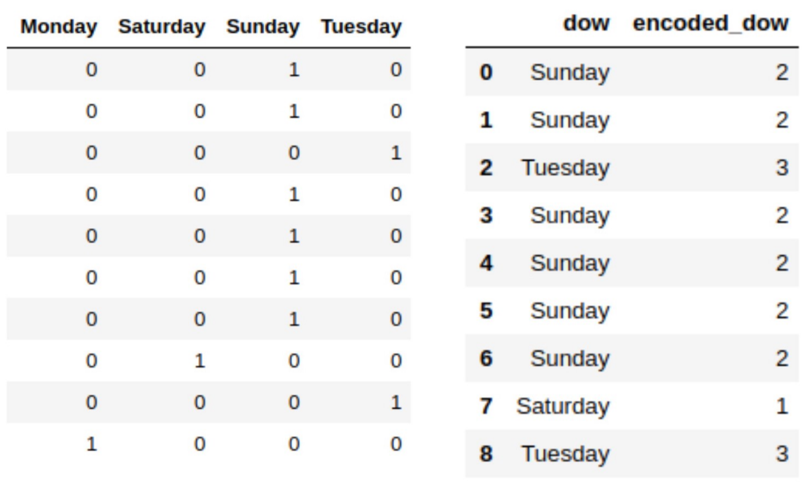
Index	Variable X_Blue	Variable X_Red	Variable X_Yellow
1	1	0	0
2	0	1	0
3	0	0	1
4	0	1	0
5	0	1	0
6	1	0	0
7	0	0	1

Eg Of One hot Encoding

Can be performed using **‘pd.get_dummies’** or **‘sklearn.preprocessing.OneHotEncoder’**

A dataset with more dimensions requires more parameters for the model to understand, and that means more rows to reliably learn those parameters. The effect of using One Hot Encoder is the addition of a number of columns (dimensions).

If the number of rows in the dataset is fixed, addition of extra dimensions without adding more information for the models to learn from can have a detrimental effect on the eventual model accuracy.

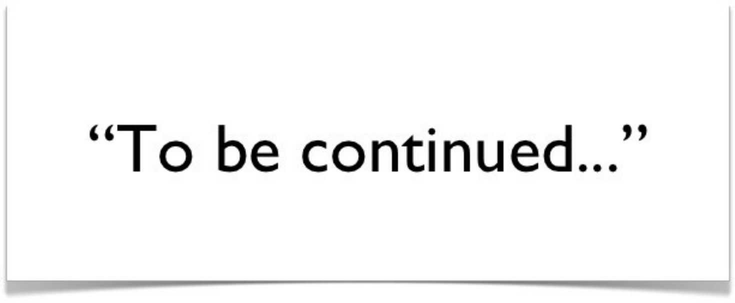


Monday	Saturday	Sunday	Tuesday
0	0	1	0
0	0	1	0
0	0	0	1
0	0	1	0
0	0	1	0
0	0	1	0
0	0	1	0
0	1	0	0
0	0	0	1
1	0	0	0

	dow	encoded_dow
0	Sunday	2
1	Sunday	2
2	Tuesday	3
3	Sunday	2
4	Sunday	2
5	Sunday	2
6	Sunday	2
7	Saturday	1
8	Tuesday	3

One Hot Encoding vs Label Encoding

With this said Part 1 comes to an end. Do read Part 2 where Feature extraction and ultra important Dimension Reduction will be discussed.



"To be continued..."