Chapter 2

Feature Engineering

Feature Engineering

A feature is an attribute of a data set that is used in a machine learning process. There is a view amongst certain machine learning practitioners that only those attributes which are meaningful to a machine learning problem are to be called as features, but this view has to be taken with a pinch of salt. In fact, selection of the subset of features which are meaningful for machine learning is a sub-area of feature engineering which draws a lot of research interest. The features in a data set are also called its dimensions. So a data set having '*n*' features is called an *n*-dimensional data set.

Consider an example of data stored in a sql database table. Table is made up of rows and columns. Table contains Integer data, String data, Date fields etc. Now consider the date column. We want to do some analysis, but this field is not directly useful. So, first we write a program (or script) to extract the day on any particular date and create a separate column with that information. Now we have 7 days (Monday Sunday etc.) stored in a new column. Again, we create a script to check if it is a day weekend or a weekday. We created another field this weekend. It contains True if the day is weekend otherwise False. We can use this for our analysis or to build predictive models. This is a Feature. So, the form of attribute (or field) which is useful in the data science process is called Feature. Features may depend on the type of requirement. In the example above we are more interested in getting the weekend information. Other analysis ma

need the day of week or month of year for analysis. In that case month of the year

is the feature. So, this is completely application dependent.

Feature is used in:

1. In Computer Vision techniques image pixels, edges, corners etc.
2. In Speech recognition sound levels, noise levels etc.

What is feature engineering?

Feature engineering refers to the process of translating a data set into features such that these features are able to represent the data set more effectively and result in a better learning performance.

As we know already, feature engineering is an important preprocessing step for machine learning. It has two major elements:

1. feature transformation

2. feature subset selection

Feature Engineering methods

Feature transformation transforms the data – structured or unstructured, into a new set of features which can represent the underlying problem which machine learning is trying to solve. There are two variants of feature transformation:

1. Feature Construction
2. Feature Extraction

Both are sometimes known as feature discovery.

Feature Construction process discovers missing information about the relationships between features and augments the feature space by creating additional features. Hence, if there are '*n*' features or dimensions in a data set, after feature construction '*m*' more features or dimensions may get added. So at the end, the data set will become '*n* + *m*' dimensional.

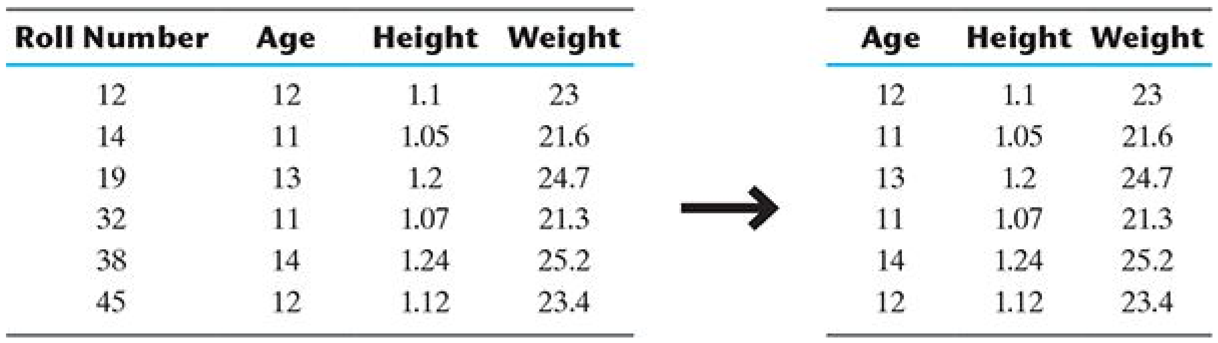
Feature Extraction is the process of extracting or creating a new set of features from the original set of features using some functional mapping.

Unlike feature transformation, in case of **feature subset selection** (or simply **feature selection**) no new feature is generated. The objective of feature selection is to derive a subset of features from the full feature set which is most meaningful in the context of a specific machine learning problem. So, essentially the job of feature selection is to derive a subset *Fj* (*F*1, *F*2, …, *Fm*) of *Fi* (*F*1, *F*2, …, *Fn*), where $m < n$, such that *Fj* is most meaningful and gets the best result for a machine learning problem.

FEATURE SUBSET SELECTION

Feature selection is arguably the most critical pre-processing activity in any machine learning project. It intends to select a subset of system attributes or features which makes a most meaningful contribution in a machine learning activity. Let's quickly discuss a practical example to understand the philosophy behind feature selection. Say we are trying to predict the weight of students based on past information about similar students, which is captured in a 'student weight' data set. The student weight data set has features such as Roll Number, Age, Height, and Weight. We can well understand that roll number can have no bearing, whatsoever, in predicting student weight. So we can eliminate the feature roll number and build a feature subset to be considered in this machine learning problem. The subset of features is expected to give better results than the full set

The same has been **depicted**



Roll Number	Age	Height	Weight
12	12	1.1	23
14	11	1.05	21.6
19	13	1.2	24.7
32	11	1.07	21.3
38	14	1.24	25.2
45	12	1.12	23.4

$\longrightarrow$

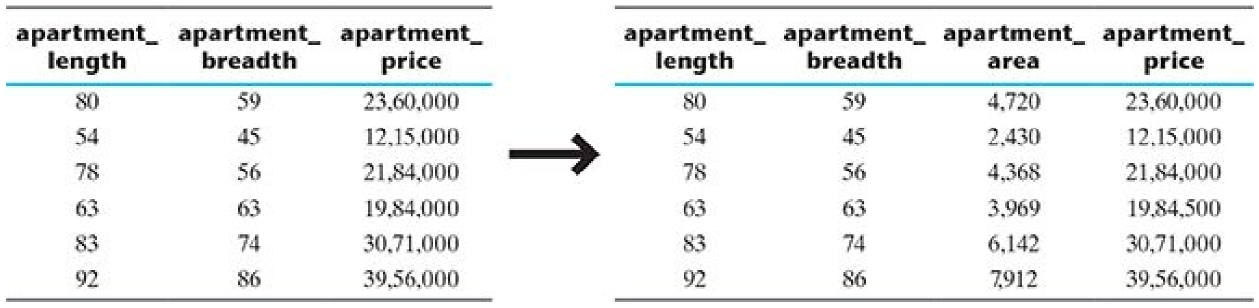
Age	Height	Weight
12	1.1	23
11	1.05	21.6
13	1.2	24.7
11	1.07	21.3
14	1.24	25.2
12	1.12	23.4

Feature selection

Feature Construction - Min Max Normalization

Feature construction involves transforming a given set of input features to generate a new set of more powerful features. To understand more clearly, let’s take the example of a real estate data set having details of all apartments sold in a specific region.

The data set has three features – apartment length, apartment breadth, and price of the apartment. If it is used as an input to a regression problem, such data can be training data for the regression model. So given the training data, the model should be able to predict the price of an apartment whose price is not known or which has just come up for sale. However, instead of using length and breadth of the apartment as a predictor, it is much convenient and makes more sense to use the area of the apartment, which is not an existing feature of the data set. So such a feature, namely apartment area, can be added to the data set. In other words, we transform the three-dimensional data set to a four-dimensional data set, with the newly ‘discovered’ feature apartment area being added to the original data set.This is depicted



apartment_ length	apartment_ breadth	apartment_ price
80	59	23,60,000
54	45	12,15,000
78	56	21,84,000
63	63	19,84,000
83	74	30,71,000
92	86	39,56,000

$\rightarrow$

apartment_ length	apartment_ breadth	apartment_ area	apartment_ price
80	59	4,720	23,60,000
54	45	2,430	12,15,000
78	56	4,368	21,84,000
63	63	3,969	19,84,500
83	74	6,142	30,71,000
92	86	7,912	39,56,000

Feature construction (example 1)

Note: Though for the sake of simplicity the features apartment length and apartment breadth have been retained in reality, it makes more sense to exclude these features when building the model.

There are certain situations where feature construction is an essential activity before we can start with the machine learning task. These situations are

- when features have categorical value and machine learning needs numeric value inputs
- when features having numeric (continuous) values and need to be converted to ordinal values
- when text-specific feature construction needs to be done

Min Max normalization:

This can be done using pre-processing functions available in scikit-learn

One min max normalize object is created, and is given for doing the transformation. Here the maximum value transforms to 1 and minimum value transforms to zero.

Once min_max transformer,contains information,it can be used for transforming test data

Standardization

Standardization is another scaling technique where the values are centered around the mean with a unit standard deviation. This means that the mean of the attribute becomes zero and the resultant distribution has a unit standard deviation.

Here’s the formula for standardization:

$$X' = \frac{X - \mu}{\sigma}$$

μ is the mean of the feature values and σ is the standard deviation of the feature values. Note that in this case, the values are not restricted to a particular range.

Handling Categorical Variables

Categorical features are those features which contain value from a predefined set For example, gender (Male and Female), colour (Red, Blue, Green), Noise Level Mery High High, Good, Low, Very Low) ete Categorical features bin the data in predefined categories.

Also some problems may arise due to categorical variables. One problem is the large number of categories. Consider an example of Indian cities. We have a data of 10000 rows and the city of India as one of the attribute (Pune, Ajmer, Jaipur Mumbai, Kanpur etc.), Let say for 10000 rows of data we have 500 different cities Now if we train this model and start predicting our model is highly depending on the city. It overfits training data to reduce the error and perform poor on generalized data. Also, if some city which is not a part of the current set comes, the model fails to predict the outcome for that.

To handle this situation, we can think of considering states instead of cities. So, we have to deal with only 29 states and 7 union territories. By reducing the number of levels, we can create a better generalized model.
Another problem is that some of the models only consider numerical attributes. Their cost function is based on the numerical value of the feature. This problem can be handled in two ways:

Replacing each value with a number We can replace each category with a number. For example color (Red 1, Green 2, Blue 3). After making this transformation, the categorical attribute will contain only numbers and we can train our algorithm. But this technique will not always explain the data correctly In above example we replaced Red with and Blue with. As we know, all the three colors have equal weightage in original data. But here we are giving more weight to blue or in another way Green is near to Red as compared to Blue Which is not correct.

Create dummy variable: To deal with the above problem, we follow a dummy variable creation approach. In this method we create one new feature for each of the categories. Each feature contains values of zero and one. If the category is equal to the feature, it will get value 1 otherwise zero. In the case of color we will create three features (is Red, is Blue, is_Green).

At one time, only one of the features can take value 1. After creating new features, we can drop the old feature (color in this case).

Also, we can do this task by creating a number of categories -1 features, In the above example we can skip creating is_Green variable. If the value of is_Red and is_Blue is zero, that means color is green. So, we are not losing any information but removing redundancy.

Countvectorizer Process:

In order to use textual data for predictive modeling, the text must be parsed to remove certain words – this process is called tokenization. These words need to then be encoded as integers, or floating-point values, for use as inputs in machine learning algorithms. This process is called feature extraction (or vectorization).

Scikit-learn's CountVectorizer is used to convert a collection of text documents to a vector of term/token counts. It also enables the pre-processing of text data prior to generating the vector representation. This functionality makes it a highly flexible feature representation module for text.

Tfidfvectorizer

Scikit-learn's Tfidftransformer and Tfidfvectorizer aim to do the same thing, which is to convert a collection of raw documents to a matrix of TF-IDF features. The differences between the two modules can be quite confusing and it's hard to know when to use which. This article shows you how to correctly use each module, the differences between the two and some guidelines on what to use when.

Tfidftransformer Usage

1. Dataset and Imports

Below we have 5 toy documents, all about my cat and my mouse who live happily together in my house. We are going to use this toy dataset to compute the tf-idf scores of words in these documents.

2. Initialize CountVectorizer

In order to start using TfidfTransformer you will first have to create a CountVectorizer to count the number of words (term frequency), limit your vocabulary size, apply stop words etc.

Compute the IDF values

Now we are going to compute the IDF values by calling tfidf_transformer.fit(word_count_vector) on the word counts we computed earlier.

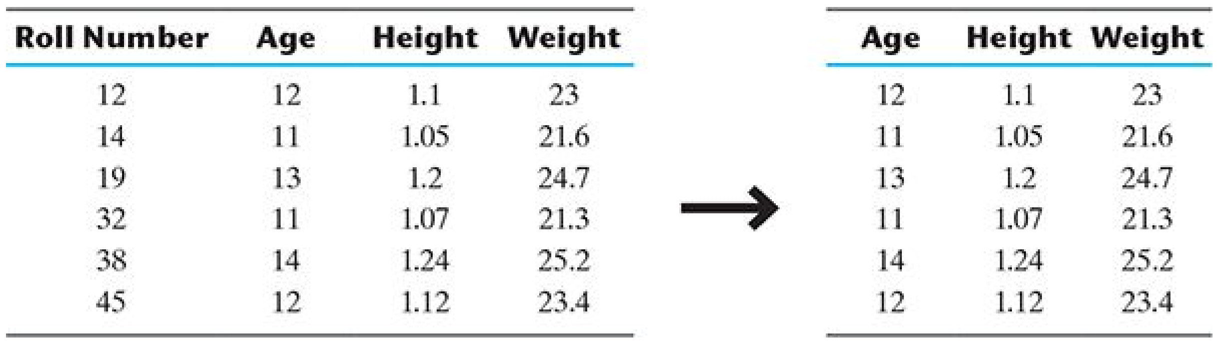
Compute the TFIDF score for your documents

Once you have the IDF values, you can now compute the tf-idf scores for any document or set of documents. Let’s compute tf-idf scores for the 5 documents in our collection.

Feature Selection:

Feature selection is arguably the most critical pre-processing activity in any machine learning project. It intends to select a subset of system attributes or features which makes a most meaningful contribution in a machine learning activity. Let’s quickly discuss a practical example to understand the philosophy behind feature selection. Say we are trying to predict the weight of students based on past information about similar students, which is captured in a ‘student weight’ data set. The student weight data set has features such as Roll Number, Age, Height, and Weight. We can well understand that roll number can have no bearing, whatsoever, in predicting student weight. So we can eliminate the feature roll number and build a feature subset to be considered in this machine learning problem. The subset of features is expected to give better results than the full set.

The same has been depicted



Roll Number	Age	Height	Weight
12	12	1.1	23
14	11	1.05	21.6
19	13	1.2	24.7
32	11	1.07	21.3
38	14	1.24	25.2
45	12	1.12	23.4

$\longrightarrow$

Age	Height	Weight
12	1.1	23
11	1.05	21.6
13	1.2	24.7
11	1.07	21.3
14	1.24	25.2
12	1.12	23.4

Feature selection

Missing data

Missing data is a very common problem in machine learning: Missing data means we don't get all feature value for some data points. We have to make some assumptions or handle this case carefully. There are multiple ways to handle missing data.

Remove features which contain missing values: One of the simplest ways to analyse the data, is to check for all the features which contain missing values and remove them. Wecan train our model with remaining features. Whenever we get test data, we have to remove features from that also. This technique is useful when we have a good set of features and only a few columns contain missing values and their correlation with dependent variables is not high. But if most of the columns contain missing values, this technique is not preferred.

Remove rows containing missing values: We can remove the rows from data which contains any data point missing. But if the test data also contains the missing values, our machine learning algorithm will fail to predict correct output. Also, if a large number of rows contains missing values, this method will reduce training data.

Replace missing values with significant data: This technique works well in most situations. Different types of data missing values can be replaced by different values. For numerical attributes, we can replace missing values by:

1.Mean of the feature values.

2.Median of the feature values.

3.Zero.

For categorical attributes, we can create a new category default for each missing data point and do the analysis.