Reference: https://www.kaggle.com/alexisbcook/categorical-variables

- **Problem**:
  - How to deal with categorical variable ?
  - You will get an error if you try to plug these variables into most machine learning models in Python without preprocessing them first.

- What is categorical variable ?
  - In data, if the responses fall into a fixed set of categories.
    For example, consider a survey that asks how often you eat breakfast and provides four options: "Never", "Rarely", "Most days", or "Every day".
  - If the responses fall into categories
    For example, what brand of car they owned ?

- **Solutions**
  1. Drop categorical variables
  2. Label encoding
  3. One-Hot Encoding

1. Drop Categorical variables
    - Advantage: easy approach; refer to previous section how to drop the columns
    - Disadvantage: this can only be done if the data does not contain the useful information
    - Code:
        - Find columns with select_dtypes() method
        - Drop those columns

2. Label Encoding

Breakfast
Every day
Never
Rarely
Most days
Never

Breakfast
3
0
1
2
0

- Assign each unique value to a different integer
- Not all categorical variables have a clear ordering in the values, but we refer to those that do as **ordinal variables**.
- For tree-based models (like decision trees and random forests), you can expect label encoding to work well with ordinal variables.

- We refer to the number of unique entries of a categorical variable as the **cardinality** of that categorical variable. For instance, the `'Street'` variable has cardinality 2.

- Code:
    - Avoid changing the original data, use copy() method
    - Use LabelEncoder class from scikit-learn to encode each column with categorical data

3. One-Hot Encoding

Color
Red
Red
Yellow
Green
Yellow

Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1
0	1	0

- Creates new columns indicating the presence (or absence) of each possible value in the original data.
- In contrast to label encoding, one-hot encoding *does not* assume an ordering of the categories. Thus, you can expect this approach to work particularly well if there is no clear ordering in categorical data (e.g., "Red" is neither *more* nor *less* than "Yellow"). We refer to categorical variables without an intrinsic ranking as **nominal variables**.

- One-hot encoding generally does not perform well if the categorical variable takes on a large number of values (i.e., you generally won't use it for variables taking more than 15 different values).

- Code:
    - Use OneHotEncoder class from scikit-learn to get one-hot encodings.
    - We set handle_unknown='ignore' to avoid errors when the validation data contains classes that aren't represented in the training data, and
    - setting sparse=False ensures that the encoded columns are returned as a numpy array (instead of a sparse matrix).

- For large datasets with many rows, one-hot encoding can greatly expand the size of the dataset. For this reason, we typically will only one-hot encode columns with relatively low cardinality. Then, high cardinality columns can either be dropped from the dataset, or we can use label encoding.

- **Understand code**
    - What's the difference between transform and fit_transform ?

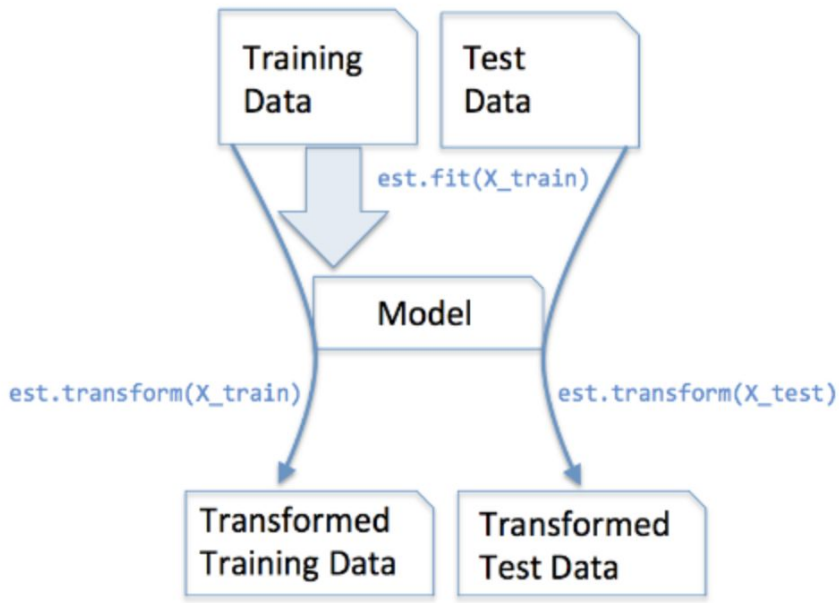
59

In **scikit-learn estimator api**,

`fit()` : used for generating learning model parameters from training data

`transform()` : parameters generated from `fit()` method,applied upon model to generate transformed data set.

`fit_transform()` : combination of `fit()` and `transform()` api on same data set



Source:https://stackoverflow.com/questions/23838056/what-is-the-difference-between-transform-and-fit-transform-in-sklearn