Preprocessing techniques

# Missing Values

Once you know a bit more about the missing data you have to decide whether or not you want to keep entries with missing data. According to Chris Albon (Machine Learning with Python Cookbook), from sklearn.impute import MissingIndicator #indicates value is missing with True/False. this decision should partially depend on how random missing values are.

If they are completely at random, they don’t give any extra information and can be omitted. On the other hand, if they’re not at random, the fact that a value is missing is itself information and can be expressed as an extra binary feature.

from sklearn.impute import MissingIndicator

indicates value is missing with True/False n-array. Only one missing value can be given at a time. Hence you may need to convert all types of missing values to one kind mostly np.nan and pass as missing\_values parameter.

from sklearn.impute import SimpleImputer

To fill missing values use imputer that indicates which value should be considered missing and the *strategy* to fill it.

# Polynomial features

Polynomial features are often created when we want to include the notion that there exists a nonlinear relationship between the features and the target.

For example, when degree is set to two and X=x1, x2, the features created will be 1, x1, x2, x1², x1x2 and x2². The interaction\_only parameter let the function know we only want the interaction features, i.e. 1, x1, x2 and x1x2.

NOTE: Missing values need to be handled well to use polynomial features. If NaN = 0 in for missing values, the interaction could generate 0 for missing value.

from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree=3, interaction\_only=True)

concate the new features to original df:

X = pd.concat([X, indicator, polynomials], axis=1)