九种数据预处理的方法

- 1. 标准化（Standardization or Mean Removal and Variance Scaling）
  - 变换后各维特征有0均值，单位方差。也叫z-score规范化（零均值规范化）。计算方式是将特征值减去均值，除以标准差。
  - sklearn.preprocessing.scale(X)
  - 一般会把train和test集放在一起做标准化，或者在train集上做标准化后，用同样的标准化器去标准化test集，此时可以用scaler
  - scaler = sklearn.preprocessing.StandardScaler().fit(train)
  - scaler.transform(train)
  - scaler.transform(test)
  - 实际应用中，需要做特征标准化的常见情景：SVM
- 2. 最小-最大规范化
  - 最小-最大规范化对原始数据进行线性变换，变换到[0,1]区间（也可以是其他固定最小最大值的区间）
  - min_max_scaler = sklearn.preprocessing.MinMaxScaler()
  - min_max_scaler.fit_transform(X_train)
- 3.规范化（Normalization）
  - 规范化是将不同变化范围的值映射到相同的固定范围，常见的是[0,1]，此时也称为归一化。
  - 将每个样本变换成unit norm。
  - X = [[1, -1, 2],[2, 0, 0], [0, 1, -1]]
  - sklearn.preprocessing.normalize(X, norm='l2')
  - 得到：
  - array([[0.40, -0.40, 0.81], [1, 0, 0], [0, 0.70, -0.70]])
  - 可以发现对于每一个样本都有，0.4^2+0.4^2+0.81^2=1,这就是L2 norm，变换后每个样本的各维特征的平方和为1。类似地，L1 norm则是变换后每个样本的各维特征的绝对值和为1。还有max norm，则是将每个样本的各维特征除以该样本各维特征的最大值。
  - 在度量样本之间相似性时，如果使用的是二次型kernel，需要做Normalization
- 4. 特征二值化（Binarization）
  - 给定阈值，将特征转换为0/1
  - binarizer = sklearn.preprocessing.Binarizer(threshold=1.1)
  - binarizer.transform(X)
- 5. 标签二值化（Label binarization）
  - lb = sklearn.preprocessing.LabelBinarizer()
- 6. 类别特征编码
  - 有时候特征是类别型的，而一些算法的输入必须是数值型，此时需要对其编码。
  - enc = preprocessing.OneHotEncoder()
  - enc.fit([[0, 0, 3], [1, 1, 0], [0, 2, 1], [1, 0, 2]])
  - enc.transform([[0, 1, 3]]).toarray() #array([[1., 0., 0., 1., 0., 0., 0., 0., 1.]])

    - 上面这个例子，第一维特征有两种值0和1，用两位去编码。第二维用三位，第三维用四位。
    - 另一种编码方式
    - newdf=pd.get_dummies(df,columns=["gender","title"],dummy_na=True)
- 7.标签编码（Label encoding）
    - le = sklearn.preprocessing.LabelEncoder()
    - le.fit([1, 2, 2, 6])
    - le.transform([1, 1, 2, 6]) #array([0, 0, 1, 2])
    - #非数值型转化为数值型
    - le.fit(["paris", "paris", "tokyo", "amsterdam"])
    - le.transform(["tokyo", "tokyo", "paris"]) #array([2, 2, 1])
- 8.特征中含异常值时
    - sklearn.preprocessing.robust_scale
- 9.生成多项式特征
    - 这个其实涉及到特征工程了，多项式特征/交叉特征。
    - poly = sklearn.preprocessing.PolynomialFeatures(2)
    - poly.fit_transform(X)
    - 原始特征：(X1,X2)
    - 转化后： (1,X1,X2,X1^2,X1X2,(X2)^2)