Homework 0

Leyang Shen

1. First, obtain the data source via filename. For each line, the first 279 attributes are stored in X and last element is stored in y.
   Second, transform the question marks into nan by numpy.

```
import numpy as np


# get data from user and append the data into X and y

def import_data(filename):
    X = []
    y = []
    file = open(filename)
    for data in file:
        value = data.strip().split(',')
        processed_data = list(map(change_question_mark, value))
        att, cor_class = processed_data[:278], int(value[279])
        X.append(att)
        y.append(cor_class)
    return X,y


# change the question mark into nah

def change_question_mark(string):
    if string == '?':
        string = np.NaN
    return float(string)
```

2. a. Replace the nan into median.

b. Because mean is the result of a model over the “errors” But one is more likely to expect a common center through regular data. Thus a median is preferred as the center because it disregards the two extremes at the sides.

c. Filter any element that is nan in the list. And return the list after filtering.

```
def discard_missing(X):
  X = filter(lambda x: x is not 'nan', X)
  return X
```

3. Make sure the input t_f is less than 1 because it is the fraction number we want to abstract from the dataset. First we get the total number of data values based on the input fraction and the length of dataset: num = int(t_f *len(dataset_list)). We also need to shuffle the data in the original list in order to make it random: shuffle = random.shuffle[dataset_list]. Then we allocate the training set and test set accordingly: train = shuffle[: num] and test = shuffle[num:].

4.
```
import random



# make sure input t_f as fraction that is less than 1.



def train_test_split(X, y, t_f):
  x_num = int(t_f * len(X))  # for example: num is the total random nums of data in X training set
  y_num = int(t_f * len(y))
  shuffle_x = random.shuffle(X)
  shuffle_y = random.shuffle(y)
  X_train = shuffle_x[:x_num]  # first t_f fraction of shuffled list
  y_train = shuffle_y[:y_num]
  X_test = shuffle_x[x_num:]  # last 1-t_f traction of shuffled list
  y_test = shuffle_y[y_num:]
  return X_train, y_train, X_test, y_test
```