```
import pandas as pd
import numpy as np
import os

os.chdir('/Users/mattparker/Desktop/Kaggle Sync')       # change working directory

          #D:\My Data\Desktop\Kaggle\Shelter Animal Outcomes\Data directory for Windows
train = pd.read_csv('train.csv')                        # read data in as pandas dataframe 'train'


train['DateTime'] = pd.to_datetime(train['DateTime'])   # convert DateTime column to true pandas
datetime


train['Intact'] = train['SexuponOutcome']   # create new columns to split SexuponOutcome into
train['Sex'] = train['SexuponOutcome']      # one for Intact, one for Sex


for i in range(len(train)):
    if train.loc[i, 'Intact'] != 'Unknown':                         # if the value is not 'Unknown'
        if type(train.loc[i, 'Intact']) == str:                     # and the value is a string
            train.loc[i, 'Intact'] = train['Intact'][i].split(' ')[0]   # split it on space, and set the Intact value
equal to the first in the list
            train.loc[i, 'Sex'] = train['Sex'][i].split(' ')[1]        # and set the Sex value equal to the sec-
ond in the list
        else:                                                   # and if it's not an 'Unknown' and it's not a string
            train.loc[i, 'Intact'] = 'Unknown'                          # set it equal to 'Unknown'
            train.loc[i, 'Sex'] = 'Unknown'                             # set it equal to 'Unknown'


train['Age'] = train['AgeuponOutcome']                              # create new column to clean Age-
uponOutcome

for i in range(len(train)):
    if len(str(train.loc[i, 'Age'])) > 4:                           # this is my way of checking if it's not a
'nan' or similar entry, since it's hard to define those by type
        if train.loc[i, 'Age'].split(' ')[1] == 'day':                  # default format is 'number timelenght',
e.g. '2 years'. Split on the space, and if the second item is 'day'
            train.loc[i, 'Age'] = train.loc[i, 'Age'].split(' ')[0]     # set value equal to the first item ('num-
ber')
        elif train.loc[i, 'Age'].split(' ')[1] == 'days':               # and repeat for 'days', 'week', 'weeks',
etc. with assumption month = 30 days, year = 365
            train.loc[i, 'Age'] = train.loc[i, 'Age'].split(' ')[0]
        elif train.loc[i, 'Age'].split(' ')[1] == 'week':
            train.loc[i, 'Age'] = 7*int(train.loc[i, 'Age'].split(' ')[0])
        elif train.loc[i, 'Age'].split(' ')[1] == 'weeks':
            train.loc[i, 'Age'] = 7*int(train.loc[i, 'Age'].split(' ')[0])
        elif train.loc[i, 'Age'].split(' ')[1] == 'month':
            train.loc[i, 'Age'] = 30*int(train.loc[i, 'Age'].split(' ')[0])
```

```
    elif train.loc[i, 'Age'].split(' ')[1] == 'months':
        train.loc[i, 'Age'] = 30*int(train.loc[i, 'Age'].split(' ')[0])
    elif train.loc[i, 'Age'].split(' ')[1] == 'year':
        train.loc[i, 'Age'] = 365*int(train.loc[i, 'Age'].split(' ')[0])
    elif train.loc[i, 'Age'].split(' ')[1] == 'years':
        train.loc[i, 'Age'] = 365*int(train.loc[i, 'Age'].split(' ')[0])
  else:
      train.loc[i, 'Age'] = -1



train['Mix'] = train['Breed']                          # Create a new column 'Mix' to be changed to
binary indicating presence of Mix or not

for i in range(len(train)):                            # iterate through the train data set
   if train.loc[i, "Breed"][-3:] == 'Mix':                 # if the 'Breed' field ends in 'Mix',
      train.loc[i, 'Mix'] = 1                          # change 'Mix' column for that animal to 1 (binary
for presence)
      train.loc[i, 'Breed'] = train.loc[i, 'Breed'][:-4]      # and remove ' Mix' from the Breed column
   else:                                           # If 'Breed' does NOT end in 'Mix',
      train.loc[i, 'Mix'] = 0                          # change 'Mix' column for that animal to 0 (binary
for absence)

Breeds_List = []                    # create empty list of unique breeds present, to be added to
train['Breeds'] = np.empty((len(train),0)).tolist()        # create column of empty lists, to be turned
into vector of binaries for presence mapping to Breeds_List

for i in range(len(train)):                            # iterate through the train data set
   train.loc[i, 'Breed'] = train.loc[i, 'Breed'].split('/')    # split on '/' so we have a list of breeds for
each animal
   for j in train.loc[i, 'Breed']:                         # for each breed present in that animal,
      if j not in Breeds_List:                             # if it's not already in Breeds_List,
         Breeds_List.append(j)                             # add it to Breeds_List

Breeds_List.sort()                  # Sort Breeds_List alphabetically

for i in range(len(train)):                         # iterate through the train data set
   for j in range(len(Breeds_List)):                    # for each integer up to the length of the unique
Breeds_List,
      if Breeds_List[j] in train.loc[i, 'Breed']:     # if unique breed j is an element of the list of
breeds present in animal i
         train.loc[i, 'Breeds'].append(1)           # append the binary 1 (for presence) to the breeds
binary list for that animal
      else:                                      # otherwise, if breed j is not present,
         train.loc[i, 'Breeds'].append(0)           # append a 0 for absence


Colors_List = []                                   # Create new empty list Colors_List which will contain
unique color values
```

```
train['Colors'] = np.empty((len(train),0)).tolist()     # Create new column of empty values 'Colors' which will encode binaries for presence of colors in Colors_List

for i in range(len(train)):                              # Iterate through integers in len(train)
   train.loc[i, 'Color'] = train.loc[i, 'Color'].split('/')    # And redefine the 'Color' values as lists split on forward slash,
   for j in train.loc[i, 'Color']:                       # Then for each color in that list for that particular entry (animal)
      if j not in Colors_List:                           # If it's not already listed somewhere in Colors_List,
         Colors_List.append(j)                           # Append it

Colors_List.sort()            # and sort alphabetically.

for i in range(len(train)):                           # Iterate through integers in len(train)
   for j in range(len(Colors_List)):                  # and for each integer (index) in len(Colors_List)
      if Colors_List[j] in train.loc[i, 'Color']:     # If the value at that position in Colors_List is in the list of colors present in the animal in question,
         train.loc[i, 'Colors'].append(1)             # Append a '1' (for presence) to that animal's list in the Colors column, to correspond to positive presence of that color
      else:                                           # Otherwise,
         train.loc[i, 'Colors'].append(0)             # Append a '0' to indicate absence of that color in that animal.
```