```
import numpy
import cv2
import imutils
import pandas
from tqdm import tqdm
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model selection import train test split
from sklearn.metrics import accuracy score

import keras
from keras.models import Model
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D, GlobalAveragePooling2D
from keras.layers import Dropout, Flatten, Dense
from keras.models import Sequential


# initialize the list of class labels MobileNet SSD was trained to
# detect, then generate a set of bounding box colors for each class
CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
           "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
           "dog", "horse", "motorbike", "person", "pottedplant", "sheep",
           "sofa", "train", "tvmonitor"]

# load our serialized model from disk
net = cv2.dnn.readNetFromCaffe("MobileNetSSD deploy.prototxt.txt",
"MobileNetSSD deploy.caffemodel")

df train = pandas.read csv('labels.csv')
df test = pandas.read csv('sample submission.csv')

targets series = pandas.Series(df train['breed'])
one hot = pandas.get dummies(targets series, sparse = True)
one hot labels = numpy.asarray(one hot)

img size = 150


x train = []
y train = []
x test = []


for i, sample in enumerate(df train['id']):
    image = cv2.imread('train/' + sample + '.jpg')
    (h, w) = image.shape[:2]
    blob = cv2.dnn.blobFromImage(cv2.resize(image, (300, 300)), 0.007843, (300,
    300), 127.5)

    # pass the blob through the network and obtain the detections and
    # predictions
    net.setInput(blob)
    detections = net.forward()

    # detect dog with highest confidence
    dogmatch = filter(lambda x: x[1] == 12, detections[0][0])
    # if dog is not detected try similar classes
    if len(dogmatch) == 0:
        dogmatch = filter(lambda x: x[1] in (8, 3, 17, 13, 10), detections[0][0])

    if len(dogmatch) > 0:
        box = dogmatch[0][3:7] * numpy.array([w, h, w, h])
        (startX, startY, endX, endY) = box.astype("int").clip(min=0)
        image = image[startY:endY, startX:endX]

    x train.append(cv2.resize(image, (img size, img size)))
    y train.append(one hot labels[i])


for i, sample in enumerate(df test['id']):
    image = cv2.imread('test/' + sample + '.jpg')
    (h, w) = image.shape[:2]
    blob = cv2.dnn.blobFromImage(cv2.resize(image, (300, 300)), 0.007843, (300,
    300), 127.5)
```

```
72
73          # pass the blob through the network and obtain the detections and
74          # predictions
75          net.setInput(blob)
76          detections = net.forward()
77
78          # detect dog with highest confidence
79          dogmatch = filter(lambda x: x[1] == 12, detections[0][0])
80          # if dog is not detected try similar classes
81          if len(dogmatch) == 0:
82              dogmatch = filter(lambda x: x[1] in (8, 3, 17, 13, 10), detections[0][0])
83
84          if len(dogmatch) > 0:
85              box = dogmatch[0][3:7] * numpy.array([w, h, w, h])
86              (startX, startY, endX, endY) = box.astype("int").clip(min=0)
87              image = image[startY:endY, startX:endX]
88
89          x test.append(cv2.resize(image, (img size, img size)))
90
91
92      y train raw = numpy.array(y train, numpy.uint8)
93      x train raw = numpy.array(x train, numpy.float32) / 255.
94      x test  = numpy.array(x test, numpy.float32) / 255.
95
96      X train, X test, y train, y test = train test split(x train raw, y train raw,
        test size=0.3, random state=1)
97
98      model = Sequential()
99      model.add(Conv2D(filters=16, kernel size=2, padding='same', activation='relu',
100                           input shape=(img size, img size, 3)))
101     model.add(MaxPooling2D(pool size=2))
102     model.add(Conv2D(filters=32, kernel size=2, padding='same', activation='relu'))
103     model.add(MaxPooling2D(pool size=2))
104     model.add(Conv2D(filters=64, kernel size=2, padding='same', activation='relu'))
105     model.add(MaxPooling2D(pool size=2))
106     model.add(Dropout(0.3))
107     model.add(Flatten())
108     model.add(Dense(500, activation='relu'))
109     model.add(Dropout(0.4))
110     model.add(Dense(120, activation='softmax'))
111
112     model.compile(optimizer='rmsprop', loss='categorical crossentropy',
        metrics=['accuracy'])
113
114
115     model.summary()
116
117
118     model.fit(X train, y train, epochs=1, validation data=(X test, y test), verbose=1)
119
120     preds = model.predict(x test, verbose=1)
121
122     sub = pandas.DataFrame(preds)
123     # Set column names to those generated by the one-hot encoding earlier
124     col names = one hot.columns.values
125     sub.columns = col names
126     # Insert the column id from the sample submission at the start of the data frame
127     sub.insert(0, 'id', df test['id'])
128     sub.to csv('out.csv', index=False)
129
```