```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Jan 18 19:30:22 2019

@author: MT
"""

%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import os
from glob import glob
from PIL import Image

from numpy import array
from numpy import argmax
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder

from sklearn import preprocessing
from sklearn.preprocessing import label_binarize
from sklearn.metrics import confusion_matrix
import itertools

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

from keras.utils.np_utils import to_categorical


# Assign project template data directory
data_dir = "~/Desktop/DS_Projects/CSC8635_ML_Project/data"
# Import metadata df
meta = pd.read_csv(os.path.join(data_dir,'HAM10000_metadata.csv'))

# Iterate through data_dir looking for jpg files and append to images_ls
images_ls = []
for dir,_,_ in os.walk(data_dir):
    images_ls.extend(glob(os.path.join(dir,"*.jpg")))

# Convert images_ls to dataframe and assign variable name
images_df = pd.DataFrame(images_ls)
images_df.columns = ['path']
# Extract image id from path for join with meta df
images_df['image_id'] = images_df['path'].str[-16:-4]

# Join image_df with meta on image id
```

```
meta = pd.merge(meta, images_df, how='left', on=['image_id'])

meta.colum

# Iterate through images, resizing down to to 100x75 pixels, converting
meta['image'] = meta['path'].map(lambda x: np.asarray(Image.open(x).resi
meta.to_pickle(os.path.join(data_dir,"meta_cache.csv"))




##### PART 2 (starting from post-munge)
# Import cached metadata df
meta = pd.read_pickle(os.path.join(data_dir,"meta_cache.csv"))

# Extract predictor variable (images) and labels as seperate vectors
x=meta['image']
y=meta['dx']

# Verify array/image pipleline integrity by converting back to image
plt.imshow(x[0])

# Perform one-hot encoding on the labels
label_encoder = LabelEncoder()
y_enc = label_encoder.fit_transform(y)
y = to_categorical(y_enc, num_classes = 7)

# Iterate through images vector, convert to float and centre (subtract a
x = np.asarray(meta['image'].tolist())
x = x.astype('float32')
x -= np.mean(x, axis=0)

# Split test/train set for predictor and label variables
x_tr, x_test, y_tr, y_test = train_test_split(x, y, test_size=0.20,rando

# Split training set further for cross validation (NOT used for talos op
x_tr_m, x_val_m, y_tr_m, y_val_m = train_test_split(x_tr, y_tr, test_siz
```