Capstone1_Milestone_Report

March 1, 2020

1 Image recognition for invasive plants - Milestone Report

1.1 Kudzu

Tangles of kudzu overwhelm trees in Georgia while cane toads threaten habitats in over a dozen countries worldwide. These are just two invasive species of many which can have damaging effects on the environment, the economy, and even human health. Despite widespread impact, efforts to track the location and spread of invasive species are so costly that they're difficult to undertake at scale.

Currently, ecosystem and plant distribution monitoring depends on expert knowledge. Trained scientists visit designated areas and take note of the species inhabiting them. Using such a highly qualified workforce is expensive, time inefficient, and insufficient since humans cannot cover large areas when sampling.

1.2 Consumer

The main consumers for this project will be government agencies around the world interested in preserving natural ecosystems and control the highly invasive plant known as "kudzu". Environmental managers will be able to make more accurate predictions about kudzu invasion and presence, thus making better decisions and increase efficient use of economic resources.

1.3 1. First Data Exploration

In [1]:
```
#Import libraries
from pathlib import Path

%matplotlib inline
import matplotlib.pyplot as plt
from IPython.display import display
import pandas as pd

import numpy as np

from PIL import Image

# open the image
kudzu = Image.open('train/3.jpg')
```

```
forest = Image.open('train/1.jpg')
# Get the image size
imgk_size = kudzu.size
imgf_size = forest.size
print("The kudzu image size is: {}".format(imgk_size))
print("The forest image size is: {}".format(imgf_size))
# check if image of kudzu and forest loads properly
kudzu
```

```
The kudzu image size is: (1154, 866)
The forest image size is: (1154, 866)
```

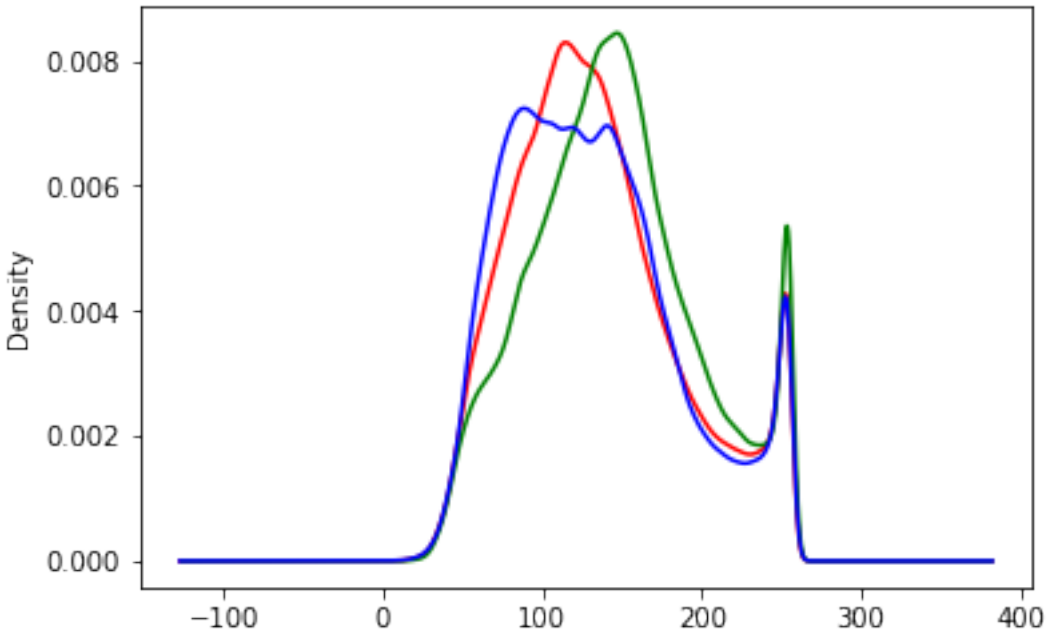
Out[1]:

In [2]: forest

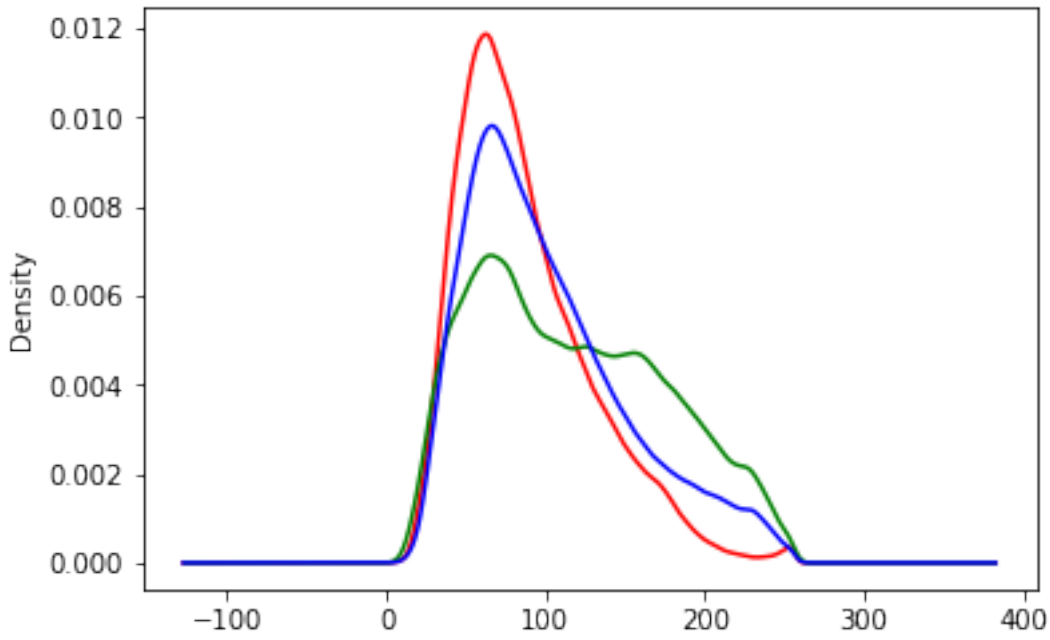
Out[2]:

In [2]:
```
#1. Explore color channels using functions learned from DataCamp

def plot_kde(channel, color):
    """ Plots a kernel density estimate for the given data.

        `channel` must be a 2d array
        `color` must be a color string, e.g. 'r', 'g', or 'b'
    """
    data = channel.flatten()
    return pd.Series(data).plot.density(c=color)

# create the list of channels
channels = ['r', 'g','b']

def plot_rgb(image_data):
    # use enumerate to loop over colors and indexes
    for ix, color in enumerate(channels):
        plot_kde(image_data[:,:,ix],color)


    plt.show()
```

```
# NumPy array image data
kudzu_data=np.array(kudzu)


# plot the rgb densities kudzu image
plot_rgb(kudzu_data)
```



In [9]:
```
# NumPy array of forest image data
forest = Image.open('train/1.jpg')
forest_data=np.array(forest)

# plot the rgb densities for the forest image
plot_rgb(forest_data)
```



Looking at these plots we can clearly observe that there is a difference in color channel density between an invaded and not invaded site. This first image exploration give us a better idea on the feasibility that a model can distinguish between a forest plot that is free from kudzo and plots where kudzu has invaded.

1.4 2. Further Data Analysis - Transform Image into usable data

Further data analysis will be carried out using the *sklearn* library to implement image manipulation and initial models machine learning models.

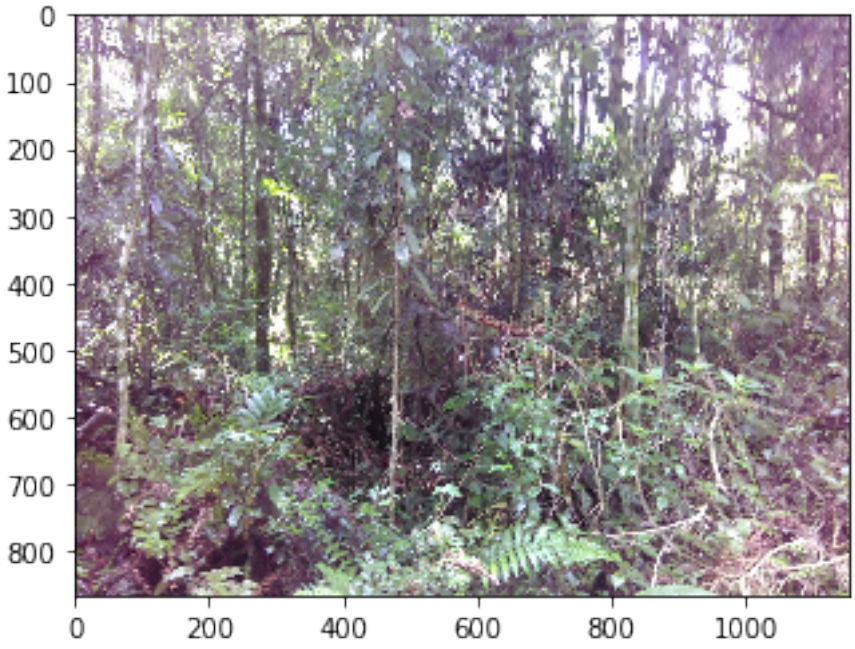
In [4]:
```
#Import python libraries

# used to change filepaths
import os

import matplotlib as mpl
import matplotlib.pyplot as plt
from IPython.display import display
%matplotlib inline

import pandas as pd
import numpy as np

from PIL import Image


from skimage.feature import hog
```

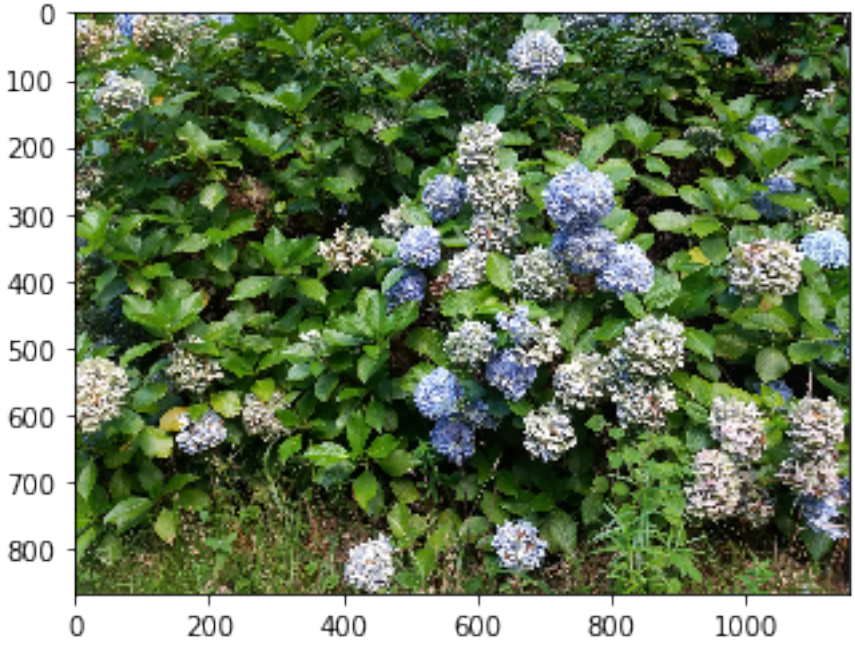
```
        from skimage.color import rgb2grey

        from sklearn.preprocessing import StandardScaler
        from sklearn.decomposition import PCA

        # import train_test_split from sklearn's model selection module
        from sklearn.model_selection import train_test_split

        # import SVC from sklearn's svm module
        from sklearn.svm import SVC

        # import accuracy_score from sklearn's metrics module
        from sklearn.metrics import roc_curve, auc, accuracy_score
```

In [5]:
```
        #2. display image of invaded and not invaded sites

        # load the labels using pandas
        labels = pd.read_csv("train_labels.csv", index_col=0)

        # show the first five rows of the dataframe using head
        display(labels.head(5))

        def get_image(row_id, root="train/"):
            """
            Converts an image number into the file path where the image is located,
            opens the image, and returns the image as a numpy array.
            """
            filename = "{}.jpg".format(row_id)
            file_path = os.path.join(root, filename)
            img = Image.open(file_path)
            return np.array(img)

        # subset the dataframe to just forest (not invaded site is 0.0) get the value of the t
        forest_row = labels[labels.invasive == 0.0].index[2]

        # show the corresponding image of forest
        plt.imshow(get_image(forest_row))
        plt.show()

        # subset the dataframe to just kudzu (invaded site is 1.0) get the value of the third
        kudzu_row = labels[labels.invasive == 1.0].index[2]

        # show the corresponding image of a Bombus
        plt.imshow(get_image(kudzu_row))
        plt.show()
```

```
      invasive
name
```

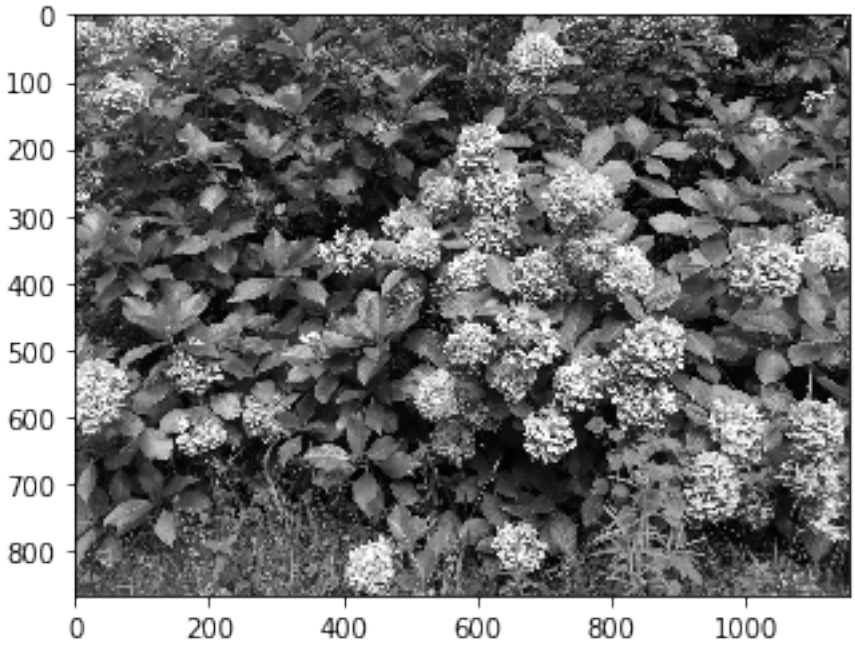
1	0
2	0
3	1
4	0
5	1


0
100
200
300
400
500
600
700
800
0
200
400
600
800
1000


0
100
200
300
400
500
600
700
800
0
200
400
600
800
1000

In [6]:
```
#3. Simplify image and convert to greyscale to explore further kudzu invaded images

# load images using our get_image function
kudzu = get_image(kudzu_row)


# print the shape of the bombus image
print('Color kudzu image has shape: ', kudzu.shape)

# convert the bombus image to greyscale
grey_kudzu = rgb2grey(kudzu)

# show the greyscale image
plt.imshow(grey_kudzu, cmap=mpl.cm.gray)

# greyscale bombus image only has one channel
print('Greyscale kudzu image has shape: ', grey_kudzu.shape)
```

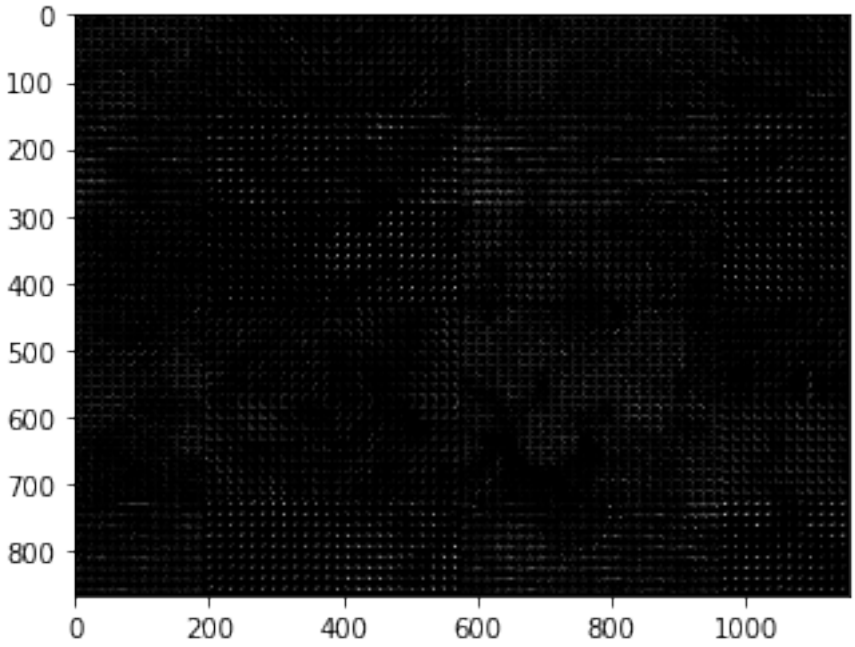
```
Color kudzu image has shape:  (866, 1154, 3)
Greyscale kudzu image has shape:  (866, 1154)
```



In [7]:
```
#4. Histogram of oriented gradients: shape within an image can be inferred by its edge
```

```
        # run HOG using greyscale kudzu image
        hog_features, hog_image = hog(grey_kudzu,
                                      visualize=True,
                                      block_norm='L2-Hys',
                                      pixels_per_cell=(16, 16))

        # show our hog_image with a grey colormap
        plt.imshow(hog_image, cmap=mpl.cm.gray)
```

Out[7]: <matplotlib.image.AxesImage at 0x1a2679afd0>



In [8]:
```
        #5. Create image features and flatten into a single row

        def create_features(img):
            # flatten three channel color image
            color_features = img.flatten()
            # convert image to greyscale
            grey_image = rgb2grey(img)
            # get HOG features from greyscale image
            hog_features = hog(grey_image, block_norm='L2-Hys', pixels_per_cell=(16, 16))
            # combine color and hog features into a single array
            flat_features = np.hstack((color_features,hog_features))
            return flat_features

        kudzu_features = create_features(kudzu)
```

```
        # print shape
        print(kudzu_features.shape)
```

```
(3292932,)
```

In [ ]: