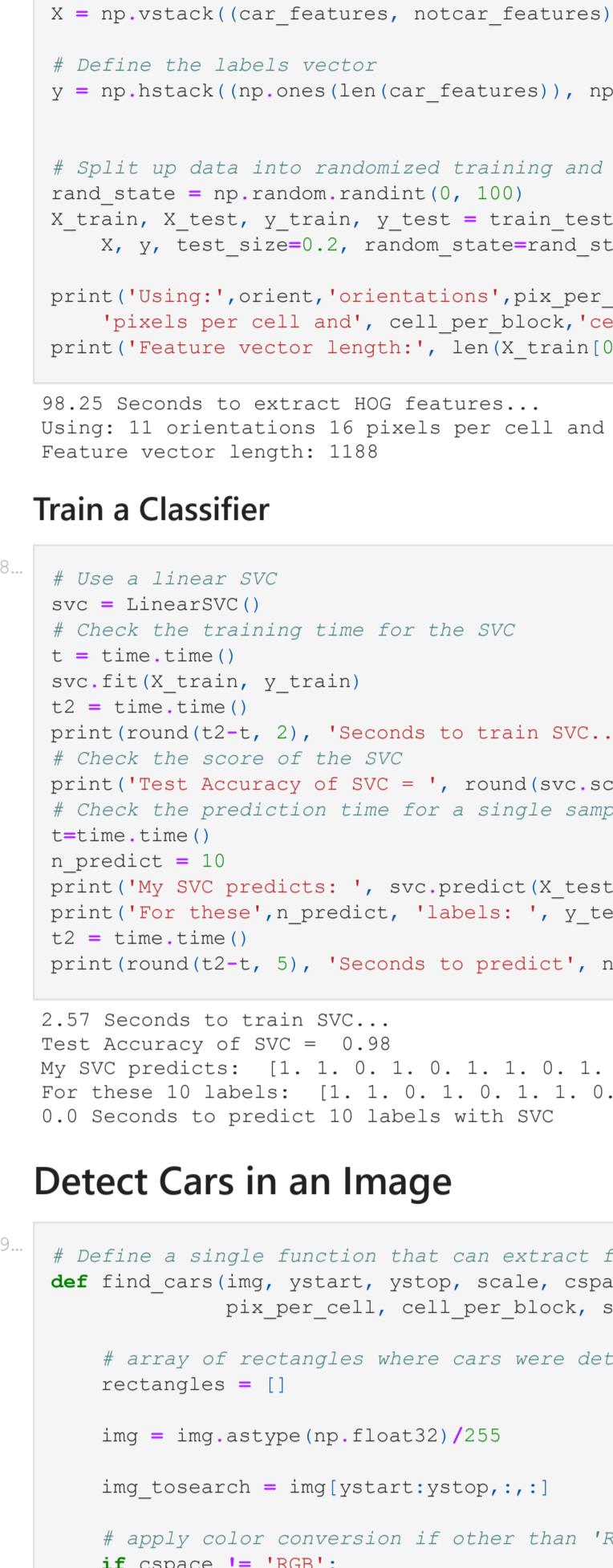


```
In [103]:  
import matplotlib.image as mpimg  
import numpy as np  
import cv2  
import glob  
import time  
import random  
  
# Import hog feature extractor and data normalization method  
from skimage import hog  
from sklearn.preprocessing import StandardScaler  
# Import SVM linear classifier  
from sklearn.svm import LinearSVC  
from sklearn.model_selection import train_test_split  
from sklearn import metrics  
from scipy import signal  
from moviepy.editor import VideoFileClip  
from IPython.display import HTML  
%matplotlib inline
```

## Load Training Data

```
In [104]:  
car_images = glob.glob('training_data/vehicles/**/*.png')  
noncar_images = glob.glob('training_data/non-vehicles/**/*.png')  
print(len(car_images), len(noncar_images))  
  
sample_numbers = [len(car_images), len(noncar_images)]  
X = [1, 2]  
plt.bar(X, sample_numbers)  
plt.xlabel('car', 'non_car')  
plt.ylabel('Sample Numbers')  
  
8792 8968
```

Out[104]: Text(0, 0.5, 'Sample Numbers')



## HOG Features

```
In [105]:  
def get_hog_features(img, orient, pix_per_cell, cell_per_block,  
                     vis=False, feature_vec=True):  
    # Call with two outputs if vis==True  
    if vis == True:  
        features, hog_image = hog(img, orientations=orient,  
                                  pixels_per_cell=pix_per_cell, cells_per_block=cell_per_block,  
                                  transform_sqrt=False,  
                                  feature_vector=feature_vec)  
    else:  
        features = hog(img, orientations=orient,  
                      pixels_per_cell=pix_per_cell, cells_per_block=cell_per_block,  
                      transform_sqrt=False,  
                      feature_vector=feature_vec)  
    return features
```

## Extract HOG Features from Car and Non-Car Images

```
In [106]:  
def extract_features(imgs, cspace='RGB', orient=9,  
                     pix_per_cell=8, cell_per_block=2, hog_channel=0):  
    # Create a list to append feature vectors to  
    features = []  
    # Iterate through the list of images  
    for img in imgs:  
        # Read in each one by one  
        # apply color conversion if other than 'RGB'  
        if cspace != 'RGB':  
            if cspace == 'YUV':  
                feature_image = cv2.cvtColor(image, cv2.COLOR_RGB2YUV)  
            elif cspace == 'LUV':  
                feature_image = cv2.cvtColor(image, cv2.COLOR_RGB2LUV)  
            elif cspace == 'HLS':  
                feature_image = cv2.cvtColor(image, cv2.COLOR_RGB2HLS)  
            elif cspace == 'YUV':  
                feature_image = cv2.cvtColor(image, cv2.COLOR_RGB2YUV)  
            elif cspace == 'YCrCb':  
                feature_image = cv2.cvtColor(image, cv2.COLOR_RGB2YCrCb)  
            else:  
                feature_image = np.copy(image)  
  
        # Call get hog features() with vis=False, feature_vec=True  
        if hog_channel == 'ALL':  
            hog_features = []  
            for channel in range(feature_image.shape[2]):  
                hog_features.append(get_hog_features(feature_image[:, :, channel],  
                                         orient, pix_per_cell, cell_per_block,  
                                         vis=False, feature_vec=True))  
            hog_features = np.ravel(hog_features)  
        else:  
            hog_features = get_hog_features(feature_image[:, :, hog_channel], orient,  
                                         pix_per_cell, cell_per_block, vis=False, feature_vec=True)  
        features.append(hog_features)  
    # Append the new feature vector to the features list  
    # Return list of feature vectors  
    return features
```

## Extract Features for Input Datasets and Combine, Define Labels Vector, Shuffle and Split

```
In [107]:  
# Feature extraction parameters  
colorspace = 'YUV' # Can be RGB, HSV, LUV, HLS, YUV, YCrCb  
orient = 11  
pix_per_cell = 16  
cell_per_block = 2  
hog_channel = "ALL" # Can be 0, 1, 2, or "ALL"  
  
t = time.time()  
car_features = extract_features(car_images, cspace=colorspace, orient=orient,  
                                pix_per_cell=pix_per_cell, cell_per_block=cell_per_block,  
                                hog_channel=hog_channel)  
notcar_features = extract_features(notcar_images, cspace=colorspace, orient=orient,  
                                   pix_per_cell=pix_per_cell, cell_per_block=cell_per_block,  
                                   hog_channel=hog_channel)  
  
t2 = time.time()  
print(round(t2-t, 2), 'Seconds to extract HOG features...')  
# Create an array stack of feature vectors  
X = np.vstack((car_features, notcar_features)).astype(np.float64)  
  
# Define the labels vector  
y = np.hstack((np.ones(len(car_features)), np.zeros(len(notcar_features))))
```

# Split up data into randomized training and test sets

```
rand_state = np.random.randint(0, 100)  
X_train, X_test, y_train, y_test = train_test_split(  
    X, y, test_size=0.2, random_state=rand_state)  
  
print('Using:', orient, 'orientations', pix_per_cell,  
      'pixels per cell and', cell_per_block, 'cells per block')  
print('Feature vector length:', len(X_train[0]))
```

98.25 Seconds to extract HOG features...

Using: 11 orientations 16 pixels per cell and 2 cells per block

Feature vector length: 1188

## Train a Classifier

```
In [108]:  
# Use a linear SVC  
svc = LinearSVC()  
# Check the training time for the SVC  
t = time.time()  
t1 = time.time()  
t2 = time.time()  
print(round(t2-t, 2), 'Seconds to train SVC...')  
# Check the score of the SVC  
print('Test Accuracy of SVC = ', round(svc.score(X_test, y_test), 4))  
# Check the prediction time for a single sample  
ttime = time.time()  
t1 = time.time()  
t2 = time.time()  
print('For these', n_predict, 'labels:', y_test[0:n_predict])  
t2 = time.time()  
print(round(t2-t, 5), 'Seconds to predict', n_predict, 'labels with SVC')
```

2.57 Seconds to train SVC..  
Test Accuracy of SVC = 0.98  
My SVC predicts: [1. 1. 0. 1. 0. 1. 1. 0. 1. 1.]  
For these 10 labels: [1. 0. 1. 0. 1. 1. 0. 1. 1.]  
0.0 Seconds to predict 10 labels with SVC

## Detect Cars in an Image

```
In [109]:  
# Define a single function that can extract features using hog sub-sampling and make predictions  
def find_cars(img, ystart, ystop, cspace, hog_channel, svc, X_scaler, orient,  
              pix_per_cell, cell_per_block, spatial_size, hist_bins, show_all_rectangles=False):  
  
    # Array of rectangles where cars were detected  
    rectangles = []  
  
    img = img.astype(np.float32)/255  
  
    img_tosearch = img[ystart:ystop,:,:]  
  
    # Apply color conversion if other than 'RGB'  
    if cspace == 'RGB':  
        crans_tosearch = cv2.cvtColor(img_tosearch, cv2.COLOR_RGB2HSV)  
    elif cspace == 'HSV':  
        crans_tosearch = cv2.cvtColor(img_tosearch, cv2.COLOR_RGB2LUV)  
    elif cspace == 'LUV':  
        crans_tosearch = cv2.cvtColor(img_tosearch, cv2.COLOR_RGB2HLS)  
    elif cspace == 'YUV':  
        crans_tosearch = cv2.cvtColor(img_tosearch, cv2.COLOR_RGB2YUV)  
    elif cspace == 'YCrCb':  
        crans_tosearch = cv2.cvtColor(img_tosearch, cv2.COLOR_RGB2YCrCb)  
    else:  
        crans_tosearch = np.copy(image)  
  
    # Rescale image if other than 1.0 scale  
    if scale != 1:  
        imshape = crans_tosearch.shape  
        crans_tosearch = cv2.resize(crans_tosearch, (np.int(imshape[1]/scale), np.int(imshape[0]/scale)))
```

```
    # Select colorspace channel for HOG  
    if hog_channel == 'ALL':  
        ch1 = crans_tosearch[:, :, 0]  
        ch2 = crans_tosearch[:, :, 1]  
        ch3 = crans_tosearch[:, :, 2]  
    else:  
        ch1 = crans_tosearch[:, :, hog_channel]  
  
    # Compute individual channel HOG features for the entire image  
    hog1 = get_hog_features(ch1, orient, pix_per_cell, cell_per_block, feature_vec=False)  
    if hog_channel == 'ALL':  
        hog2 = get_hog_features(ch2, orient, pix_per_cell, cell_per_block, feature_vec=False)  
        hog3 = get_hog_features(ch3, orient, pix_per_cell, cell_per_block, feature_vec=False)
```

```
    for xb in range(xstart, xstop):  
        for yb in range(ystart, ystop):  
            ypos = yb*cells_per_step  
            xpos = xb*cells_per_step  
            # Extract HOG for this patch  
            hog_feat1 = hog1[ypos:ypos+blocks_per_window, xpos:xpos+blocks_per_window].ravel()  
            if hog_channel == 'ALL':  
                hog_feat2 = hog2[ypos:ypos+blocks_per_window, xpos:xpos+blocks_per_window].ravel()  
                hog_feat3 = hog3[ypos:ypos+blocks_per_window, xpos:xpos+blocks_per_window].ravel()  
                hog_features = np.concatenate((hog_feat1, hog_feat2, hog_feat3))  
            else:  
                hog_features = hog1[ypos:ypos+blocks_per_window, xpos:xpos+blocks_per_window].ravel()  
  
            xleft = xpos*pix_per_cell  
            ytop = ypos*pix_per_cell  
  
            test_prediction = svc.predict(hog_features.reshape(1, -1))  
  
            if test_prediction == 1 or show_all_rectangles:  
                xbox_left = np.int(xleft*scale)  
                ytop_draw = np.int(ytop*scale)  
                win_draw = np.int(window*scale)  
                rectangles.append(((xbox_left, ytop_draw), (win_draw+ystart), (xbox_left+win_draw, ytop_draw+win_draw+ystart)))
```

return rectangles

## Draw Rectangles onto an Image

```
In [110]:  
# Here boxes your image from the previous exercise  
def draw_boxes(img, bboxes, color=(0, 255, 255), thick=6):  
    # Make a copy of the image  
    imcopy = np.copy(img)  
    random_color = False  
    # Iterate through the bounding boxes  
    for bbox in bboxes:  
        if color == 'random' or random_color:  
            color = (np.random.randint(0, 255), np.random.randint(0, 255), np.random.randint(0, 255))  
        # Draw a rectangle given bbox coordinates  
        cv2.rectangle(imcopy, bbox[0], bbox[1], color, thick)  
    # Return the image copy with boxes drawn  
    return imcopy
```

```
In [111]:  
test_img = mpimg.imread('./test_images/test1.jpg')  
  
rectangles = []  
  
colorspace = 'YUV' # Can be RGB, HSV, LUV, HLS, YUV, YCrCb  
orient = 11  
pix_per_cell = 16  
cell_per_block = 2  
hog_channel = 'ALL' # Can be 0, 1, 2, or "ALL"  
  
ystart = 400  
ystop = 464  
scale = 1.0  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 416  
ystop = 480  
scale = 1.0  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 400  
ystop = 496  
scale = 1.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 432  
ystop = 528  
scale = 1.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 400  
ystop = 528  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 432  
ystop = 560  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 400  
ystop = 596  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 464  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 400  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 432  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 400  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 432  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 400  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 432  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 400  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 432  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 400  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 432  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 400  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 432  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 400  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 432  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 400  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 432  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 400  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 432  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 400  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 432  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 400  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 432  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 400  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 432  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 400  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 432  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 400  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 432  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 400  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 432  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 400  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 432  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 400  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 432  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 400  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 432  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 400  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 432  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 400  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 432  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 400  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 432  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 400  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 432  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 400  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 432  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  
                           orient, pix_per_cell, cell_per_block, None, None))  
ystart = 400  
ystop = 660  
scale = 3.5  
rectangles.append(find_cars(test_img, ystart, ystop, scale, colorspace, hog_channel, svc, None,  

```