GWD-EDA-homework

September 13, 2020

0.0.1 Global Wheat Detection : Exploratory Data Analysis (EDA)

https://www.kaggle.com/rohitsingh9990/simpleeda-visualizations

https://www.kaggle.com/devvindan/wheat-detection-eda

. . (, ,) ( ) 3000 , , , 1000 .

[?]

. . . .
, , , .

[?] * train.csv - * sample_submission.csv - * train.zip - (3422) * test.zip - (10)

[train.csv] * image_id - * width - * height - * bbox - [xmin, ymin, width, height] * source -

[?]

. .

[2]:
```
import warnings
warnings.simplefilter('ignore')
```

[28]:
```
import socket
hostname = socket.gethostname()
print(hostname)
```

bbox

[27]:
```
!pwd
```

```
/home/hyejoo/work/repos/Study/Wheat
```

0.0.2

[4]:
```
import pandas as pd  # package for high-performance, easy-to-use data␣
↪structures and data analysis
import numpy as np  # fundamental package for scientific computing with Python
import matplotlib
import os
```

[5]:
```
from PIL import Image, ImageDraw
from ast import literal_eval
import matplotlib.pyplot as plt # for plotting
import seaborn as sns # for making plots with seaborn
color = sns.color_palette()
import plotly.offline as py
py.init_notebook_mode(connected=True)
from plotly.offline import init_notebook_mode, iplot
init_notebook_mode(connected=True)
import plotly.graph_objs as go
import plotly.offline as offline
offline.init_notebook_mode()
```

0.0.3

[29]:
```
#csv
BASE_PATH = '.'
train = pd.read_csv(f'{BASE_PATH}/train.csv')
print(len(train))
display(train.head())
```

```
147793
```

	image_id	width	height	bbox	source
0	b6ab77fd7	1024	1024	[834.0, 222.0, 56.0, 36.0]	usask_1
1	b6ab77fd7	1024	1024	[226.0, 548.0, 130.0, 58.0]	usask_1
2	b6ab77fd7	1024	1024	[377.0, 504.0, 74.0, 160.0]	usask_1
3	b6ab77fd7	1024	1024	[834.0, 95.0, 109.0, 107.0]	usask_1
4	b6ab77fd7	1024	1024	[26.0, 144.0, 124.0, 117.0]	usask_1

[30]:
```
#
BASE_PATH = '.'
TRAIN_DIR = f'{BASE_PATH}/train'
TEST_DIR = f'{BASE_PATH}/test'
print(TRAIN_DIR)
print(TEST_DIR)
```

```
./train
./test
```

[31]:
```
print('Size of train data', train.shape)
```

```
Size of train data (147793, 5)
```

bbox

[9]:
```
unique_images = train['image_id'].unique()
num_total = len(os.listdir(TRAIN_DIR))
num_annotated = len(unique_images)
```

[10]:
```
print("bbox       {} , bbox       {}    .".format(num_total, num_total -␣
↪num_annotated))
```

```
bbox       3422 , bbox       49    .
```

[11]:
```
train['height'].unique() == [1024]
```

[11]:
```
array([ True])
```

[12]:
```
train['width'].unique() == [1024]
```

[12]:
```
array([ True])
```

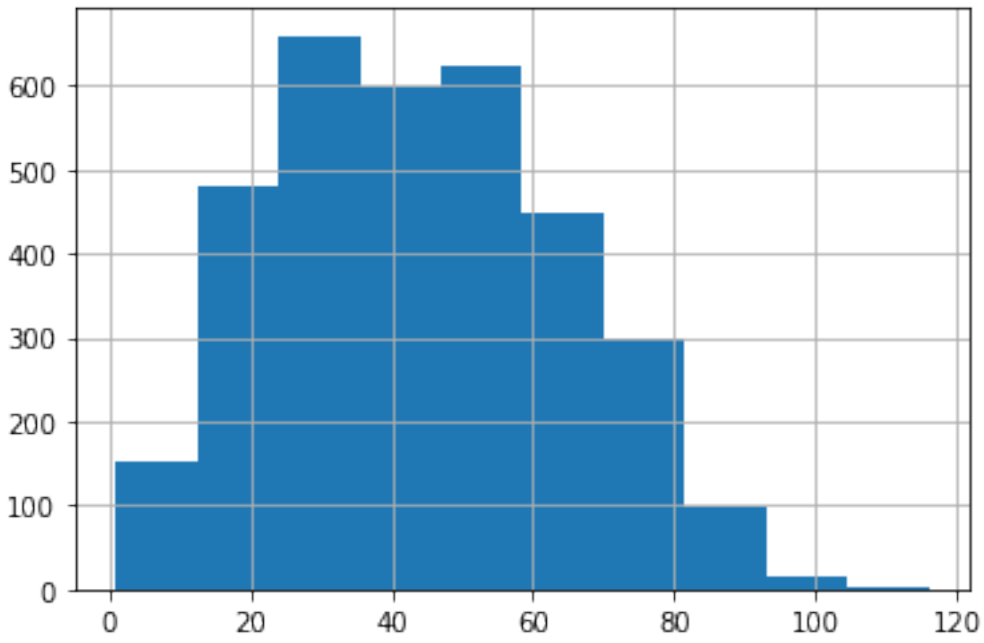
bbox

[13]:
```
train['image_id'].value_counts()
```

[13]:
```
35b935b6c    116
f1794c924    108
f79336a8e    104
134d4a01c    101
4cf91995d     97
             ...
6a62af644      1
76595919e      1
a5cb30c38      1
86296fc32      1
22f341965      1
Name: image_id, Length: 3373, dtype: int64
```

[14]:
```
train['image_id'].value_counts().hist()
```

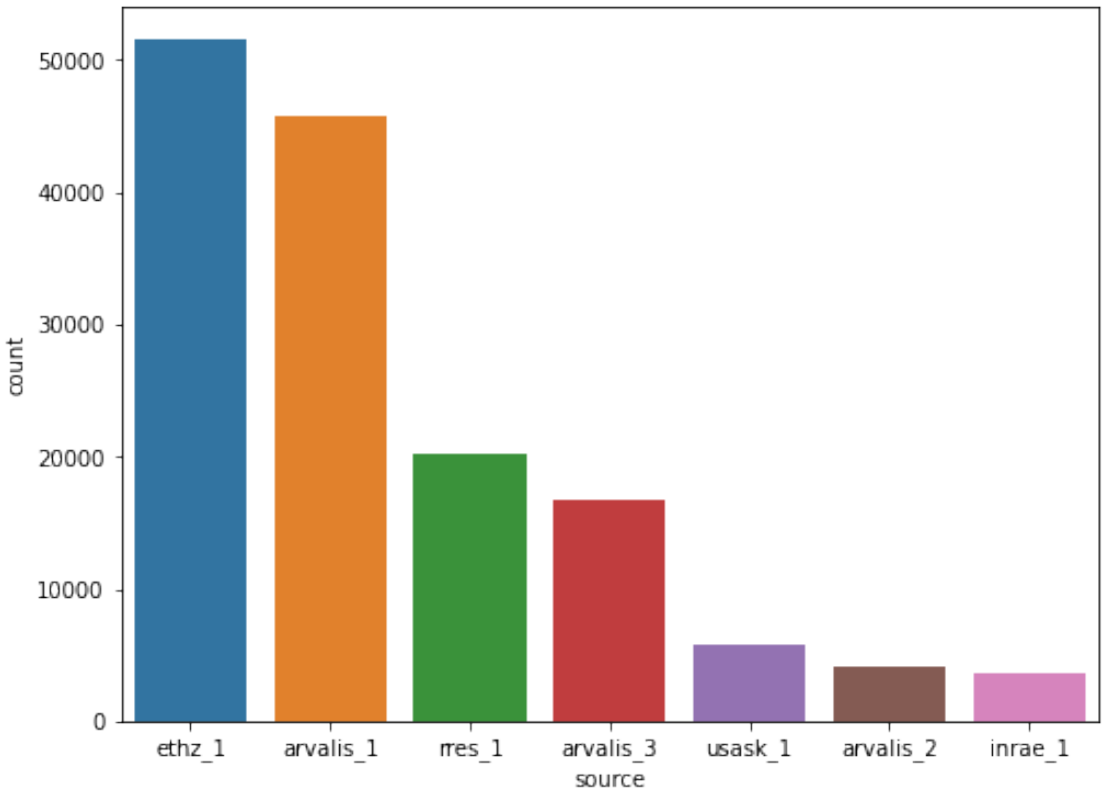
[14]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f01d2f4e390>
```



[15]:
```
import seaborn as sns
import matplotlib.pyplot as plt

plt.subplots(1,1, figsize=(8,6))
sns.countplot(train['source'], order=train['source'].value_counts().index)
```

[15]: <matplotlib.axes._subplots.AxesSubplot at 0x7f01d82b28d0>



- source
- ethz_1 arvalis_1 2 (65 %).
- .

[18]:
```
display(train.describe())
```

```
          width    height
count  147793.0  147793.0
mean     1024.0    1024.0
std         0.0       0.0
min      1024.0    1024.0
25%      1024.0    1024.0
50%      1024.0    1024.0
75%      1024.0    1024.0
max      1024.0    1024.0
```

* * . * isnull() True . * : df.isnull().sum()

[32]:
```
#
df = pd.DataFrame({"a":[1,2,3,4],"b":[1,2,3,6],"c":[1, None, 3, 5]})
display(df)
```

```
total = df.isnull().sum()
print(total)
```

```
   a  b    c
0  1  1  1.0
1  2  2  NaN
2  3  3  3.0
3  4  6  5.0


a    0
b    0
c    1
dtype: int64
```

[21]:
```
total = train.isnull().sum().sort_values(ascending = False)
percent = (train.isnull().sum()/train.isnull().count()*100).
 ↪sort_values(ascending = False)
missing_train_data  = pd.concat([total, percent], axis=1, keys=['Total',␣
 ↪'Percent'])
#print(len(missing_train_data))
print("    ")
missing_train_data.head()
```

[21]:

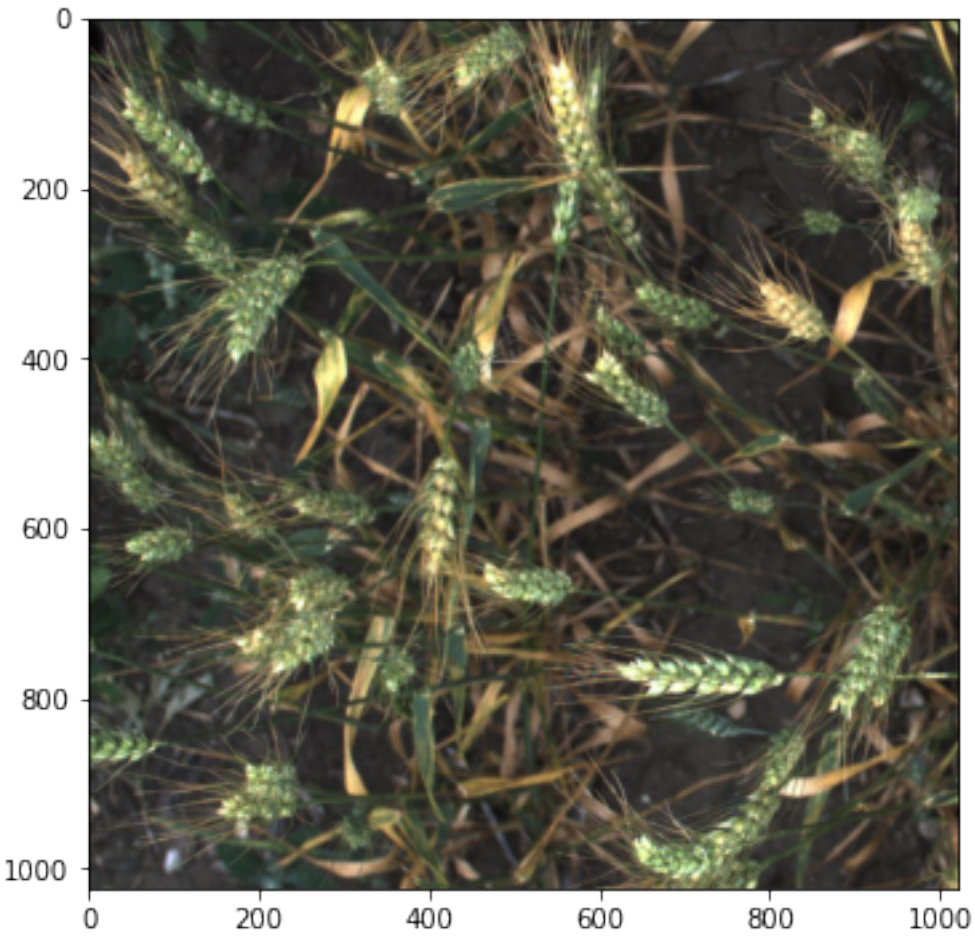
	Total	Percent
source	0	0.0
bbox	0	0.0
height	0	0.0
width	0	0.0
image_id	0	0.0

0.0.4 Train

[36]:
```
#
image_path = os.path.join(TRAIN_DIR, 'b6ab77fd7.jpg')
print(image_path)
image = Image.open(image_path)
draw = ImageDraw.Draw(image)

plt.figure(figsize = (6,6))
plt.imshow(image)
plt.show()
```

```
./train/b6ab77fd7.jpg
```



[25]:
```
train['image_id'] == 'b6ab77fd7'
```

[25]:
```
0          True
1          True
2          True
3          True
4          True
           ...
147788    False
147789    False
147790    False
147791    False
147792    False
Name: image_id, Length: 147793, dtype: bool
```

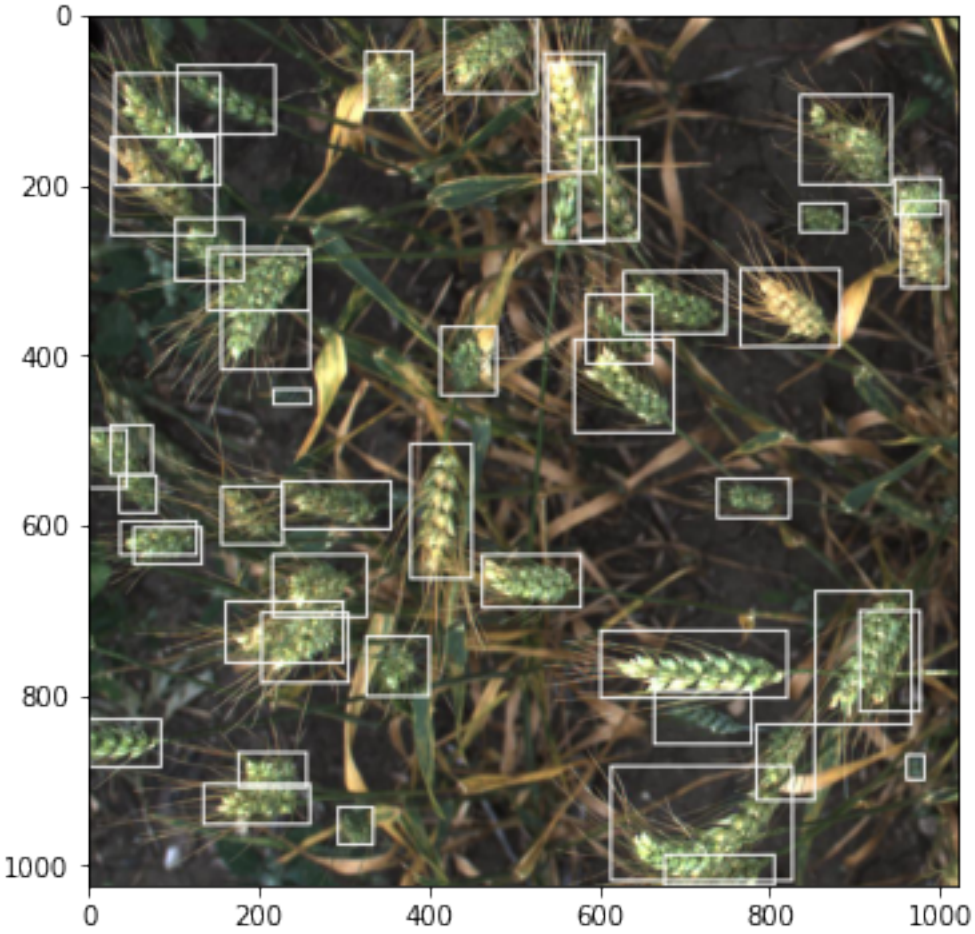
[37]:
```
#     bbox
image_path = os.path.join(TRAIN_DIR, 'b6ab77fd7.jpg')
```

```
print(image_path)
image = Image.open(image_path)
draw = ImageDraw.Draw(image)

bboxes = [literal_eval(box) for box in train[train['image_id'] ==␣
↪'b6ab77fd7']['bbox']]
for bbox in bboxes:
    #print(bbox)
    draw.rectangle([bbox[0], bbox[1], bbox[0] + bbox[2], bbox[1] + bbox[3]],␣
↪width=3)


plt.figure(figsize = (6,6))
plt.imshow(image)
plt.show()
```

./train/b6ab77fd7.jpg

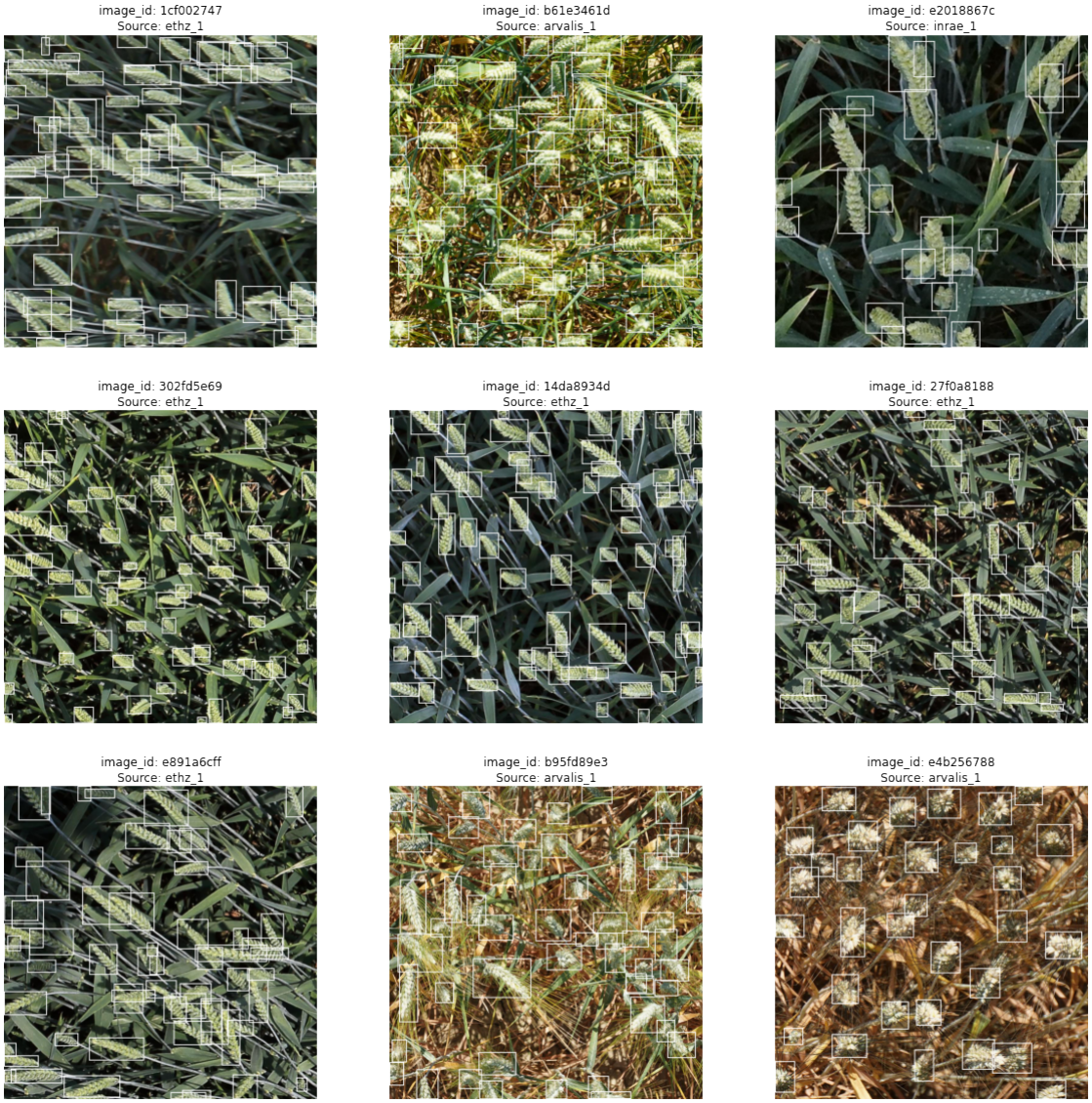


[49]:
```
def display_images_large(images):
    f, ax = plt.subplots(3,3, figsize=(20, 20))
    for i, image_id in enumerate(images):
        image_path = os.path.join(TRAIN_DIR, f'{image_id}.jpg')
        image = Image.open(image_path)
        bboxes = [literal_eval(box) for box in train[train['image_id'] ==␣
↪image_id]['bbox']]
        # draw rectangles on image
        draw = ImageDraw.Draw(image)
        for bbox in bboxes:
            draw.rectangle([bbox[0], bbox[1], bbox[0] + bbox[2], bbox[1] +␣
↪bbox[3]], width=3)

        ax[i//3, i%3].imshow(image)
        ax[i//3, i%3].axis('off')
        source = train[train['image_id'] == image_id]['source'].values[0]
        ax[i//3, i%3].set_title(f"image_id: {image_id}\nSource: {source}")

    plt.show()
```

[50]:
```
images = train.sample(n=9, random_state=42)['image_id'].values
display_images_large(images)
```



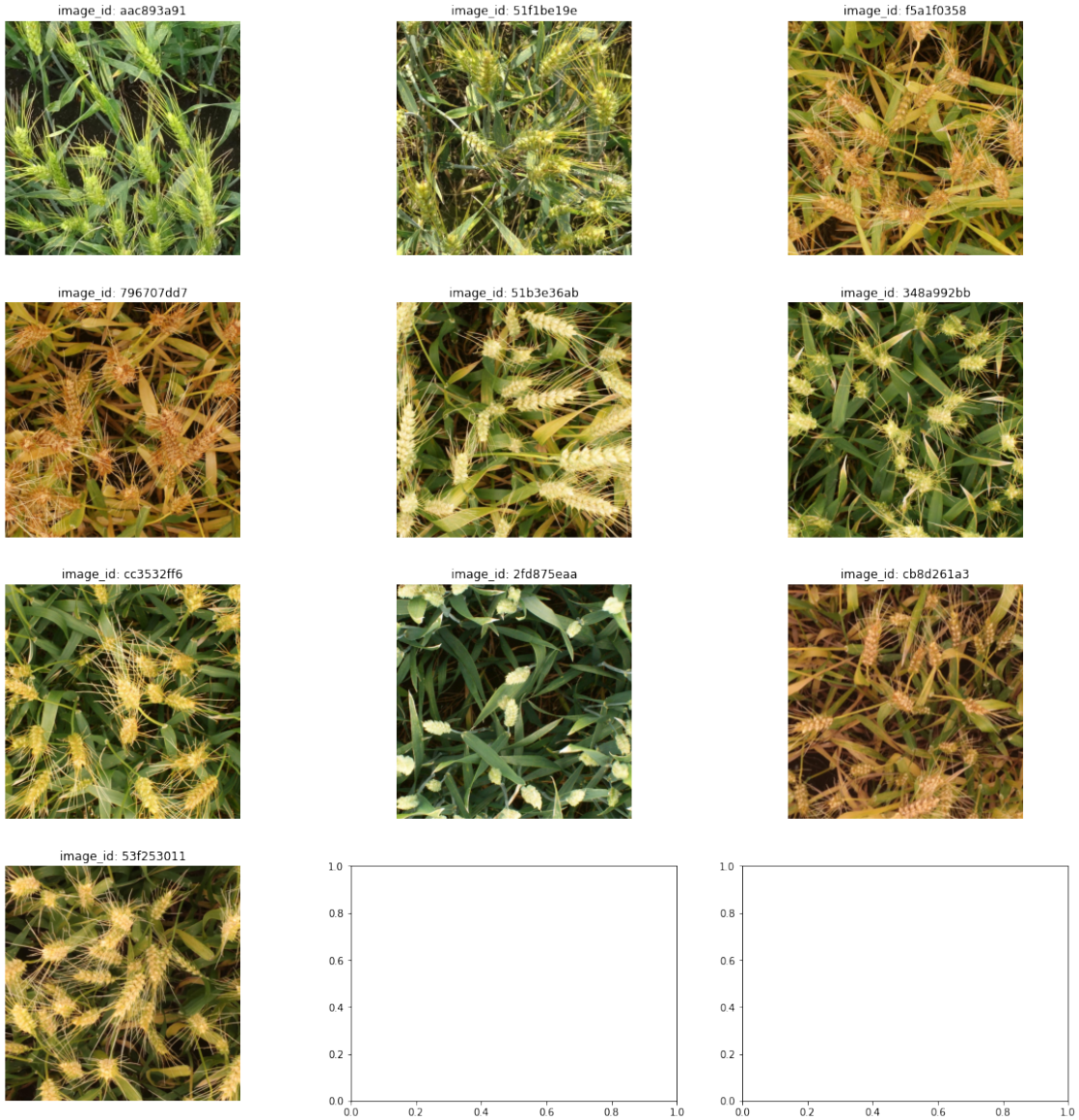
0.0.5 Test

[51]:
```
def display_test_images(images):
    f, ax = plt.subplots(4,3, figsize=(20, 20))
    for i, image_id in enumerate(images):
        image_path = os.path.join(TEST_DIR, f'{image_id}.jpg')
        image = Image.open(image_path)

        ax[i//3, i%3].imshow(image)
        ax[i//3, i%3].axis('off')
        ax[i//3, i%3].set_title(f"image_id: {image_id}")
```

```
    plt.show()
```

[52]:
```
# since we need to predict bounding boxes for test images, hence below images␣
↪do not have any bounding boxes
test_images = submission.image_id.values
display_test_images(test_images)
```



0.0.6

- . .

[53]:
```
#number of unique images in the dataframe
len(train['image_id'].unique())
```

[53]: 3373

[54]:
```
#number of images in the training directory
len(os.listdir(TRAIN_DIR))
```

[54]: 3422

[55]:
```
#obtaining a list of all images which have no wheat heads in them
#append .jpg to image ids for easier handling
#train['image_id'] = train['image_id'].apply(lambda x: str(x) + '.jpg')
unique_imgs_wbox = list(train['image_id'].unique())
all_unique_imgs = os.listdir(TRAIN_DIR)
all_unique_imgs2 = []
for i in all_unique_imgs:
    i = i.split('.')[0]
    all_unique_imgs2.append(i)
no_wheat_imgs = [img_id for img_id in all_unique_imgs2 if img_id not in␣
 ↪unique_imgs_wbox]
len(no_wheat_imgs)
```

[55]: 49

[67]:
```
def display_nowheat_images(images):
    f, ax = plt.subplots(3,3, figsize=(20, 20))
    for i, image_id in enumerate(images):
        image_path = os.path.join(TRAIN_DIR, f'{image_id}.jpg')
        image = Image.open(image_path)

        ax[i//3, i%3].imshow(image)
        ax[i//3, i%3].axis('off')
        ax[i//3, i%3].set_title(f"image_id: {image_id}")

    plt.show()
```

[ ]:
```
img_ids = np.random.choice(no_wheat_imgs, 9)
display_nowheat_images(img_ids)
```

0.0.7

[61]:
```
def show_images(images):
    f, ax = plt.subplots(5,3, figsize=(20, 30))
    #images_to_show = np.random.choice(images, num)

    for image_id in images_to_show:

        image_path = os.path.join(TRAIN_DIR, image_id + ".jpg")
        image = Image.open(image_path)

        # get all bboxes for given image in [xmin, ymin, width, height]
        bboxes = [literal_eval(box) for box in train[train['image_id'] ==␣
↪image_id]['bbox']]

        # visualize them
        draw = ImageDraw.Draw(image)
        for bbox in bboxes:
            draw.rectangle([bbox[0], bbox[1], bbox[0] + bbox[2], bbox[1] +␣
↪bbox[3]], width=3)

        #plt.figure(figsize = (5,5))
        #plt.imshow(image)
        ax[i//3, i%3].imshow(image)
        ax[i//3, i%3].axis('off')
        ax[i//3, i%3].set_title(f"image_id: {image_id}")
        plt.show()
```

[62]:
```
sources = train['source'].unique()
```

[63]:
```
img_ids = np.random.choice(no_wheat_imgs, 15)
```

[64]:
```
images
```

[64]:
```
array(['1cf002747', 'b61e3461d', 'e2018867c', '302fd5e69', '14da8934d',
       '27f0a8188', 'e891a6cff', 'b95fd89e3', 'e4b256788'], dtype=object)
```

[65]:
```
images = train.sample(n=10, random_state=42)['image_id'].values
display_images_large(images)
```

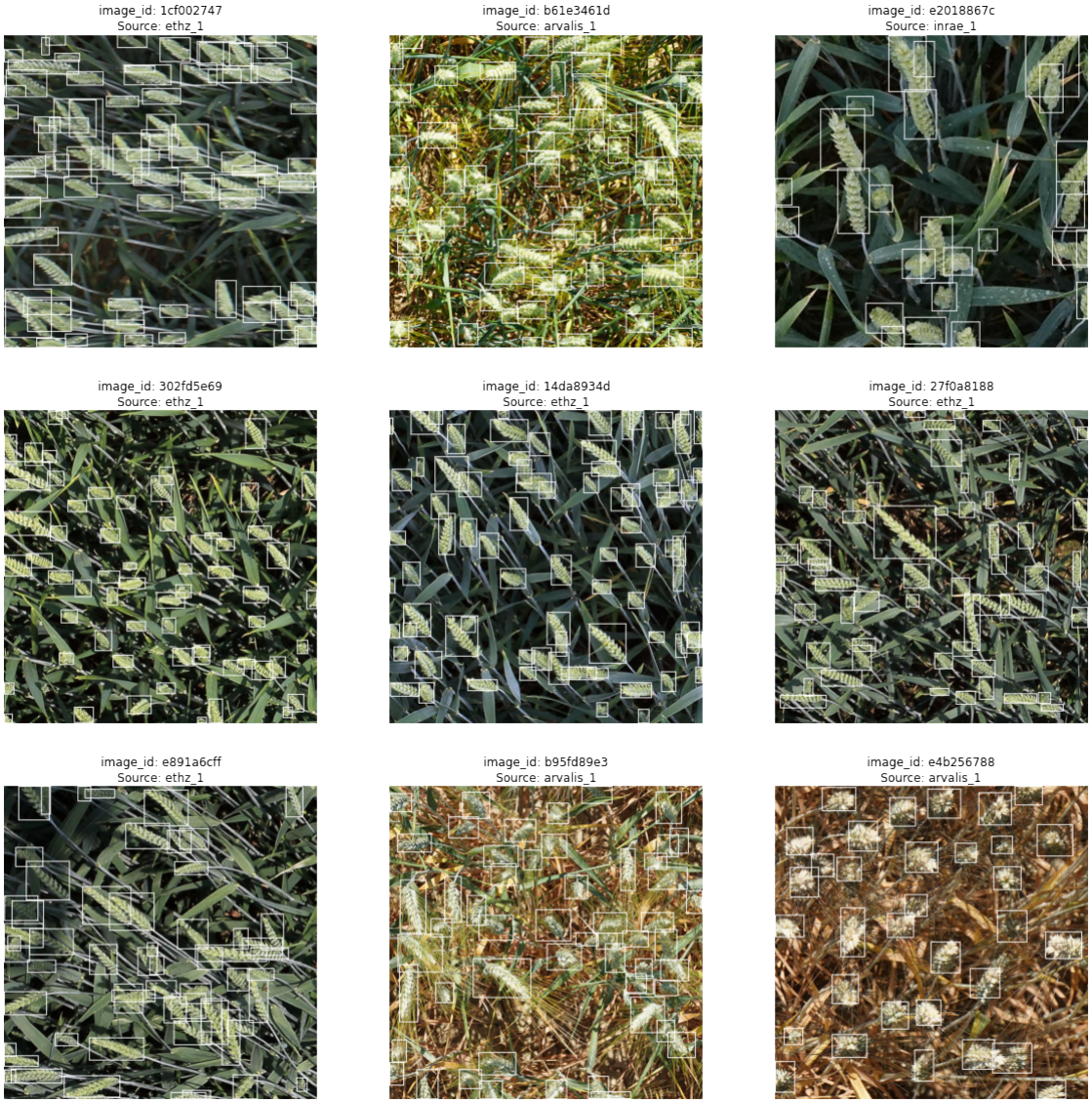
```
      ␣
↪---------------------------------------------------------------------------

       IndexError                                Traceback (most recent call␣
↪last)
```

```
      <ipython-input-65-08e1543b50b3> in <module>
        1 images = train.sample(n=10, random_state=42)['image_id'].values
  ----> 2 display_images_large(images)


      <ipython-input-49-85eef81f7f0d> in display_images_large(images)
       10             draw.rectangle([bbox[0], bbox[1], bbox[0] + bbox[2],␣
↪bbox[1] + bbox[3]], width=3)
       11
  ---> 12         ax[i//3, i%3].imshow(image)
       13         ax[i//3, i%3].axis('off')
       14         source = train[train['image_id'] == image_id]['source'].
↪values[0]


      IndexError: index 3 is out of bounds for axis 0 with size 3
```



[66]:
```
images = np.random.choice(train[train['source'] == source]['image_id'].
 ↪values,15)
show_images(images)
```

```
      ␣
↪---------------------------------------------------------------------------

      NameError                                 Traceback (most recent call␣
↪last)

      <ipython-input-66-990559f519c8> in <module>
```

```
----> 1 images = np.random.choice(train[train['source'] ==␣
↪source]['image_id'].values,15)
      2 show_images(images)


NameError: name 'source' is not defined
```

[ ]:
```
for source in sources:
    print(f"Showing images for {source}:")
    show_images(train[train['source'] == source]['image_id'].unique())
```

[ ]:
```
#!jupyter nbconvert --to pdf GWD-EDA-homework.ipynb
```