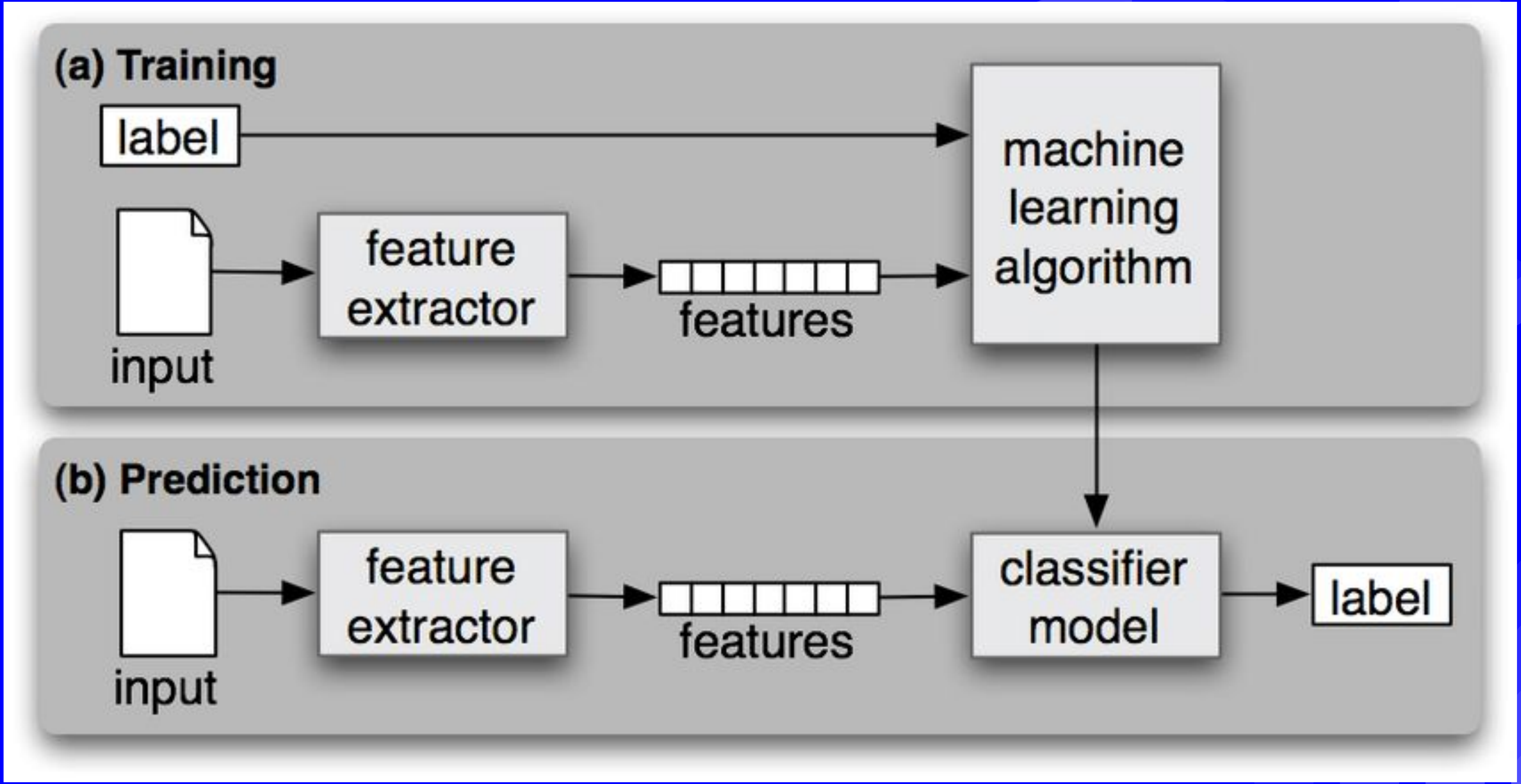
CROP DISEASE USING CNN MODEL

MACHINE LEARNING

- Machine Learning (ML) is concerned with computer programs that automatically improve their performance through experience.

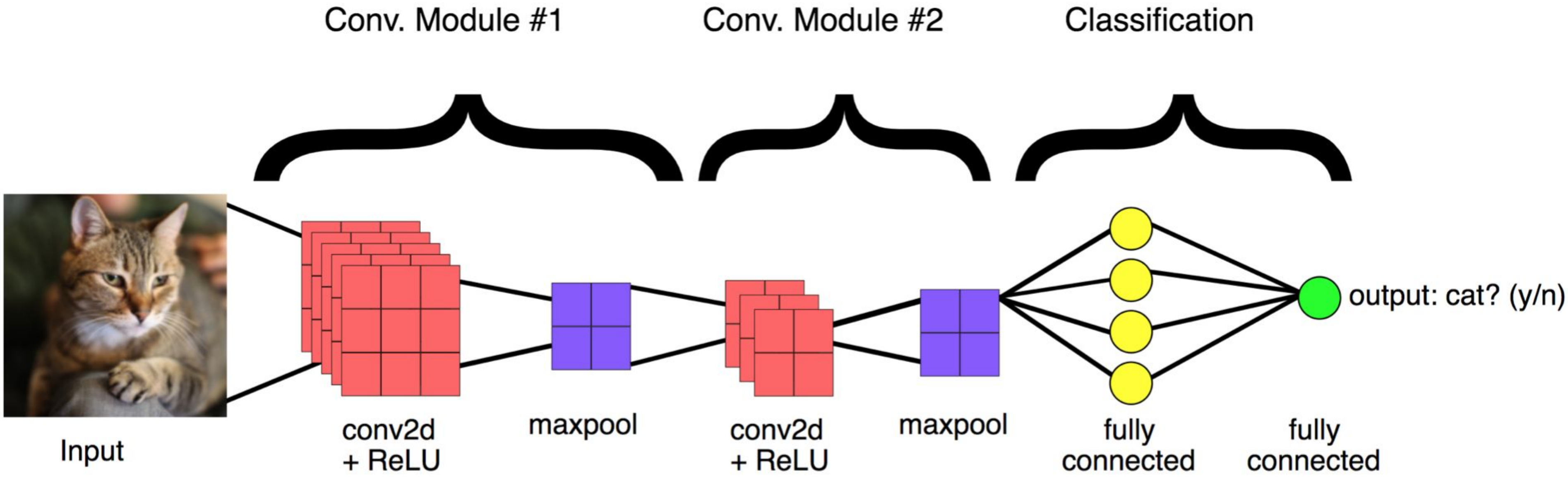

(a) Training
label
machine learning algorithm
input
feature extractor
features
(b) Prediction
input
feature extractor
features
classifier model
label

WHY ML ?

- Increasing demand of data scientist in real world is on peak.
- Increasing computational power.
- Growing progress in available algorithms and theory developed by researchers.
- Flood of available data on internet.

The concept of learning in a ML system

- Learning = Improving with experience at some task
  - Improve over task.
  - With respect to performance
  - Based on experience


Conv. Module #1
Conv. Module #2
Classification
Input
conv2d
+ ReLU
maxpool
conv2d
+ ReLU
maxpool
fully
connected
fully
connected
output: cat? (y/n)

STEPS for model

Step-1 :Input image

Step-2 :Convolution

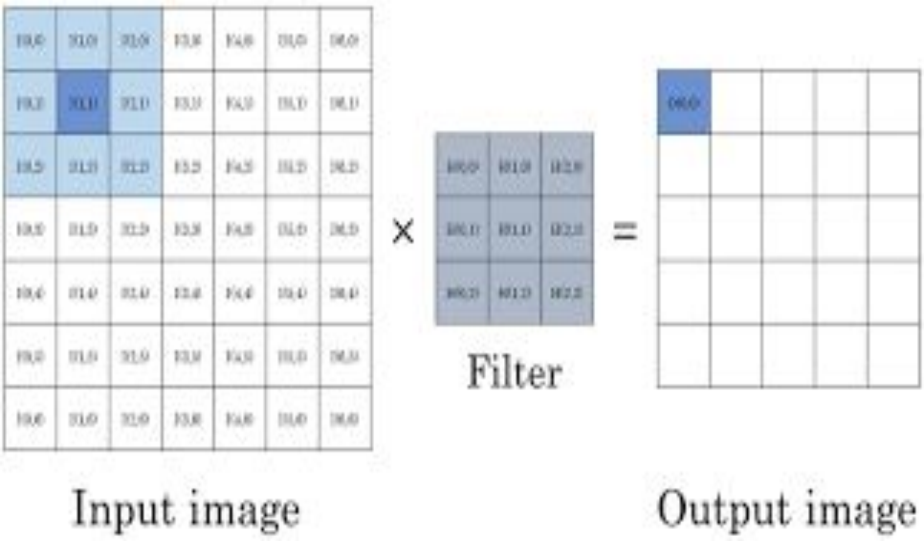
Apply filters for feature detection

Step-3 :Max pooling

Purpose-To detect image in case flipped, inverted...etc.

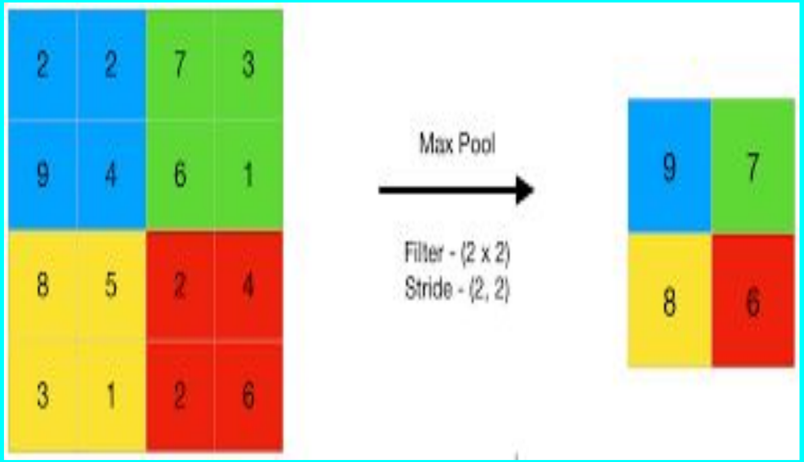
Step-4 :Flattening

Converting data from matrix to column



STEPS for model

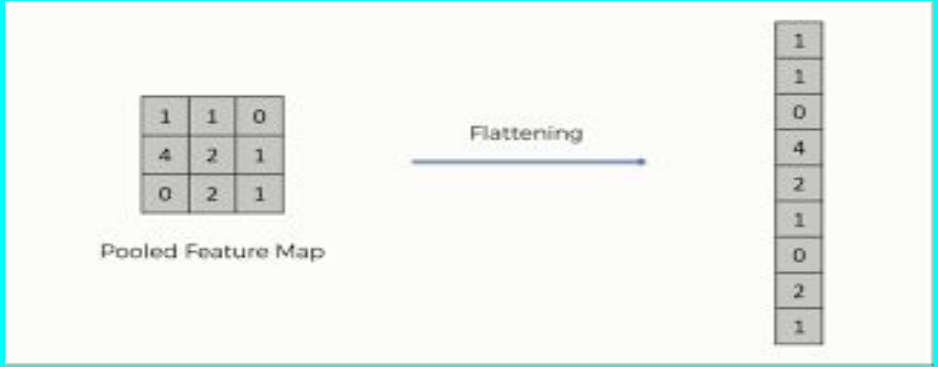
Step - 3 :



2	2	7	3
9	4	6	1
8	5	2	4
3	1	2	6

9	7
8	6

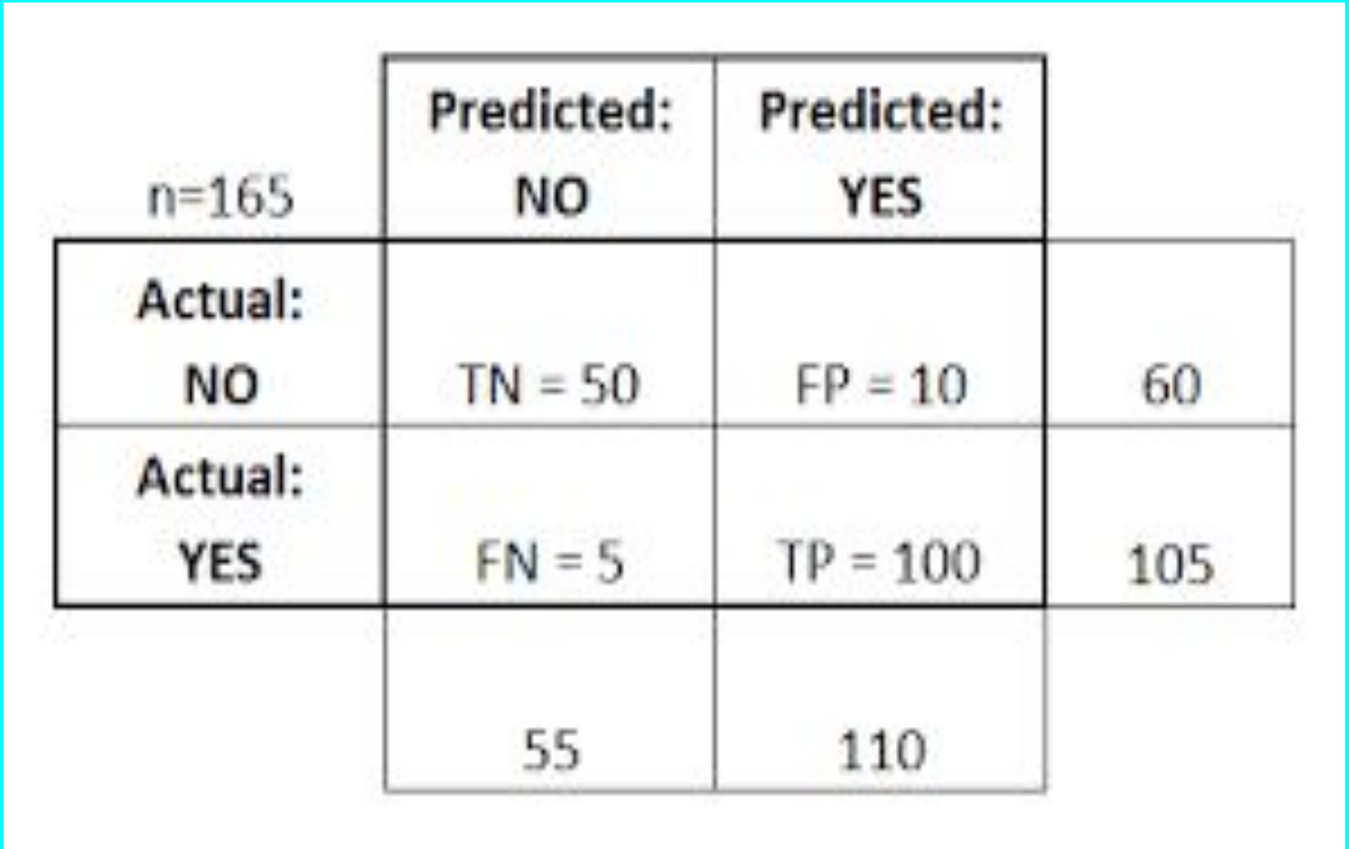
Step - 4 :



1	1	0
4	2	1
0	2	1

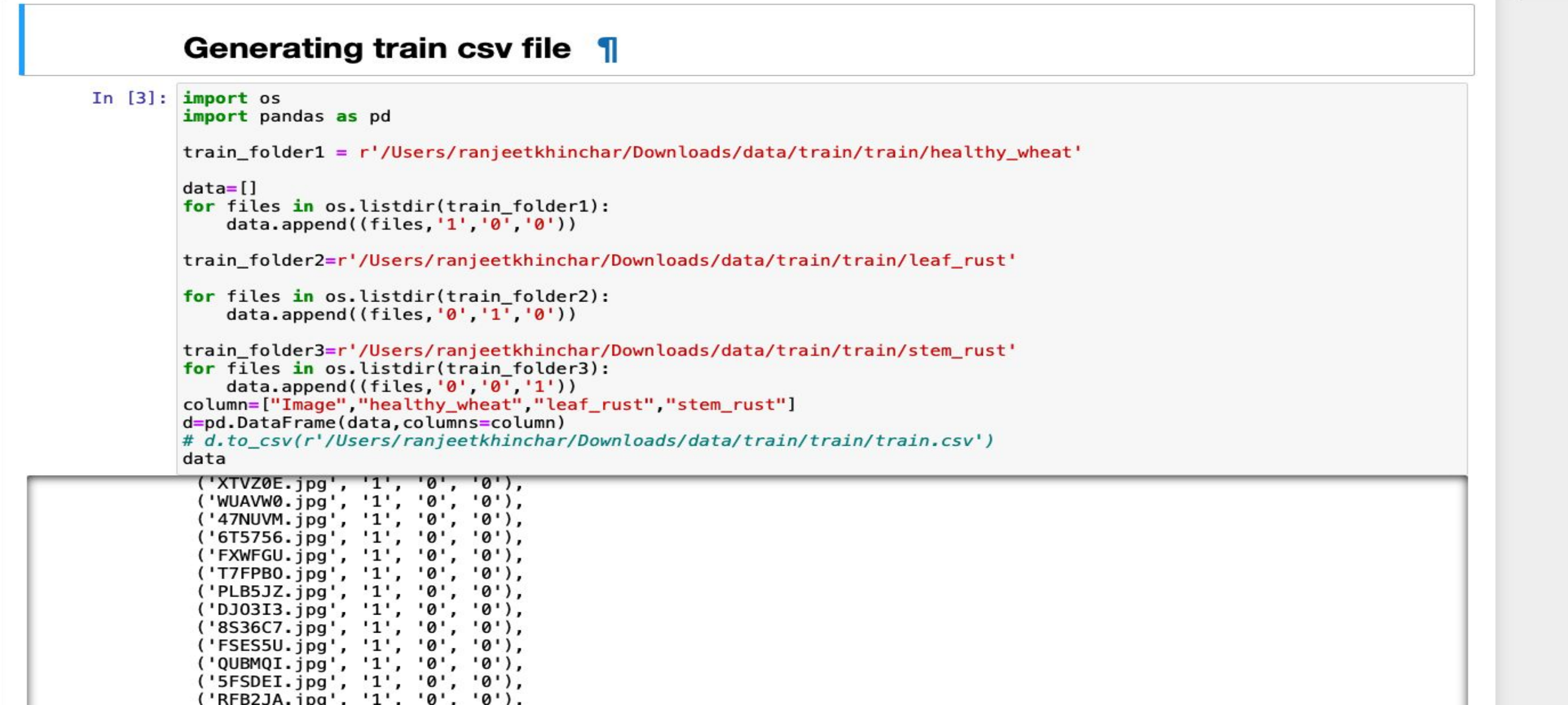
1
1
0
4
2
1
0
2
1

Confusion matrix



n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

LOADING DATAS

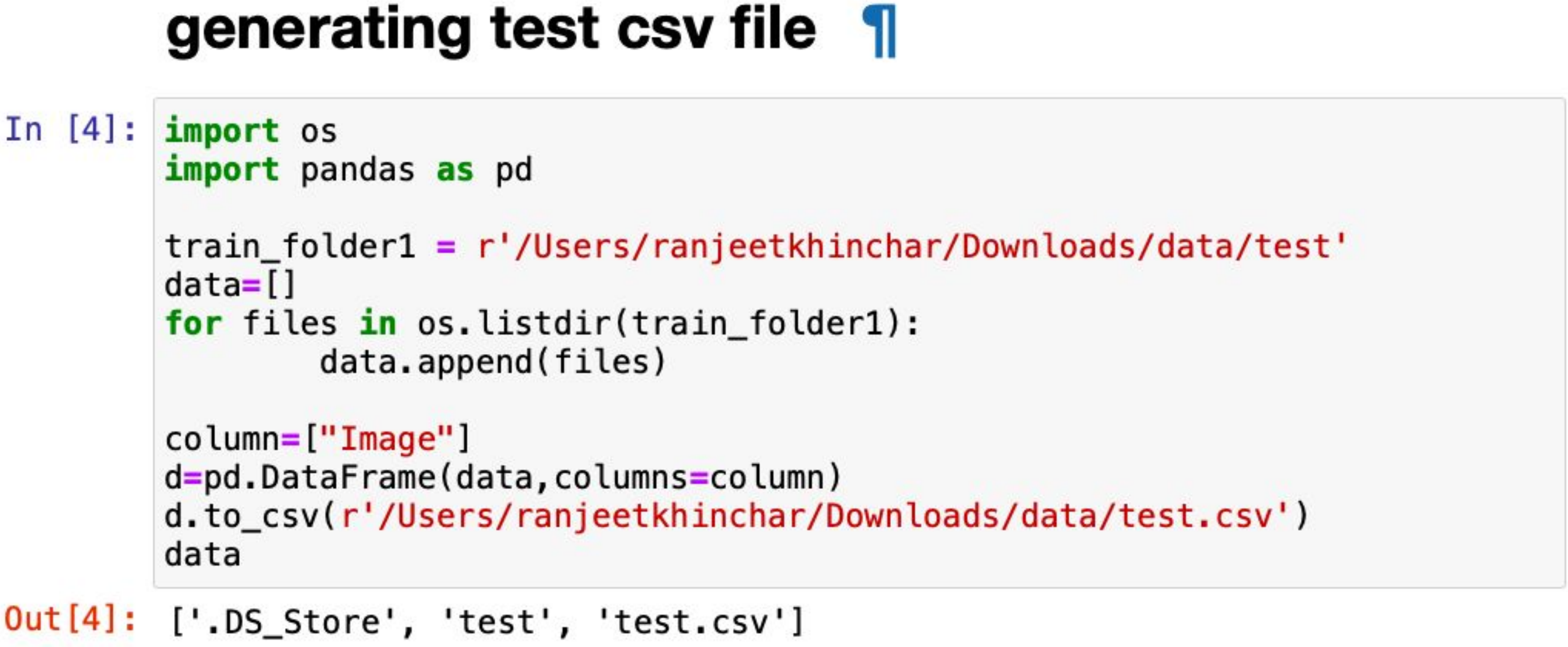


Generating train csv file ¶

In [3]:

```
import os
import pandas as pd

train_folder1 = r'/Users/ranjeetkhinchar/Downloads/data/train/train/healthy_wheat'

data=[]
for files in os.listdir(train_folder1):
    data.append((files,'1','0','0'))

train_folder2=r'/Users/ranjeetkhinchar/Downloads/data/train/train/leaf_rust'

for files in os.listdir(train_folder2):
    data.append((files,'0','1','0'))

train_folder3=r'/Users/ranjeetkhinchar/Downloads/data/train/train/stem_rust'
for files in os.listdir(train_folder3):
    data.append((files,'0','0','1'))
column=["Image","healthy_wheat","leaf_rust","stem_rust"]
d=pd.DataFrame(data,columns=column)
# d.to_csv(r'/Users/ranjeetkhinchar/Downloads/data/train/train/train.csv')
data
```

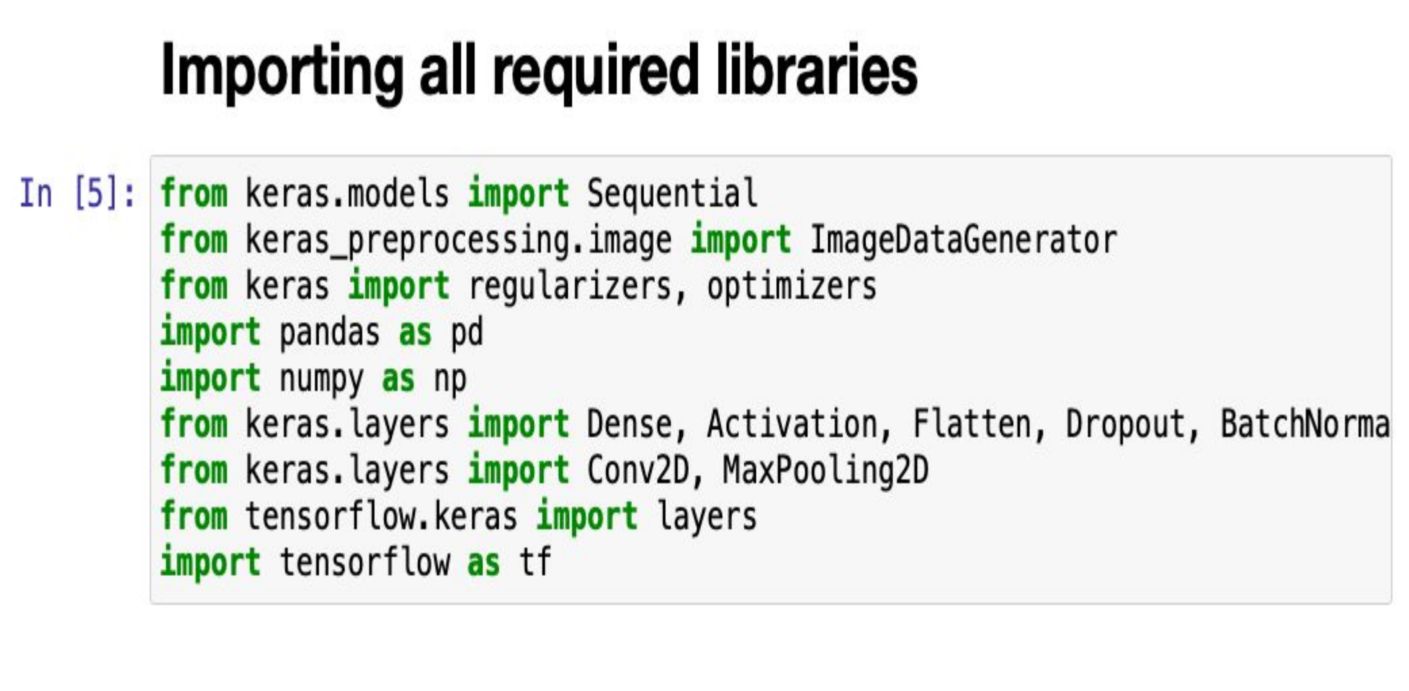
```
 ('XTVZ0E.jpg', '1', '0', '0'),
 ('WUAVW0.jpg', '1', '0', '0'),
 ('47NUVM.jpg', '1', '0', '0'),
 ('6T5756.jpg', '1', '0', '0'),
 ('FXWFGU.jpg', '1', '0', '0'),
 ('T7FPBO.jpg', '1', '0', '0'),
 ('PLB5JZ.jpg', '1', '0', '0'),
 ('DJO3I3.jpg', '1', '0', '0'),
 ('8S36C7.jpg', '1', '0', '0'),
 ('FSES5U.jpg', '1', '0', '0'),
 ('QUBMQI.jpg', '1', '0', '0'),
 ('5FSDEI.jpg', '1', '0', '0'),
 ('RFB2JA.jpg', '1', '0', '0'),
```



generating test csv file ¶

```
In [4]: import os
        import pandas as pd

        train_folder1 = r'/Users/ranjeetkhinchar/Downloads/data/test'
        data=[]
        for files in os.listdir(train_folder1):
                data.append(files)

        column=["Image"]
        d=pd.DataFrame(data,columns=column)
        d.to_csv(r'/Users/ranjeetkhinchar/Downloads/data/test.csv')
        data

Out[4]: ['.DS_Store', 'test', 'test.csv']
```

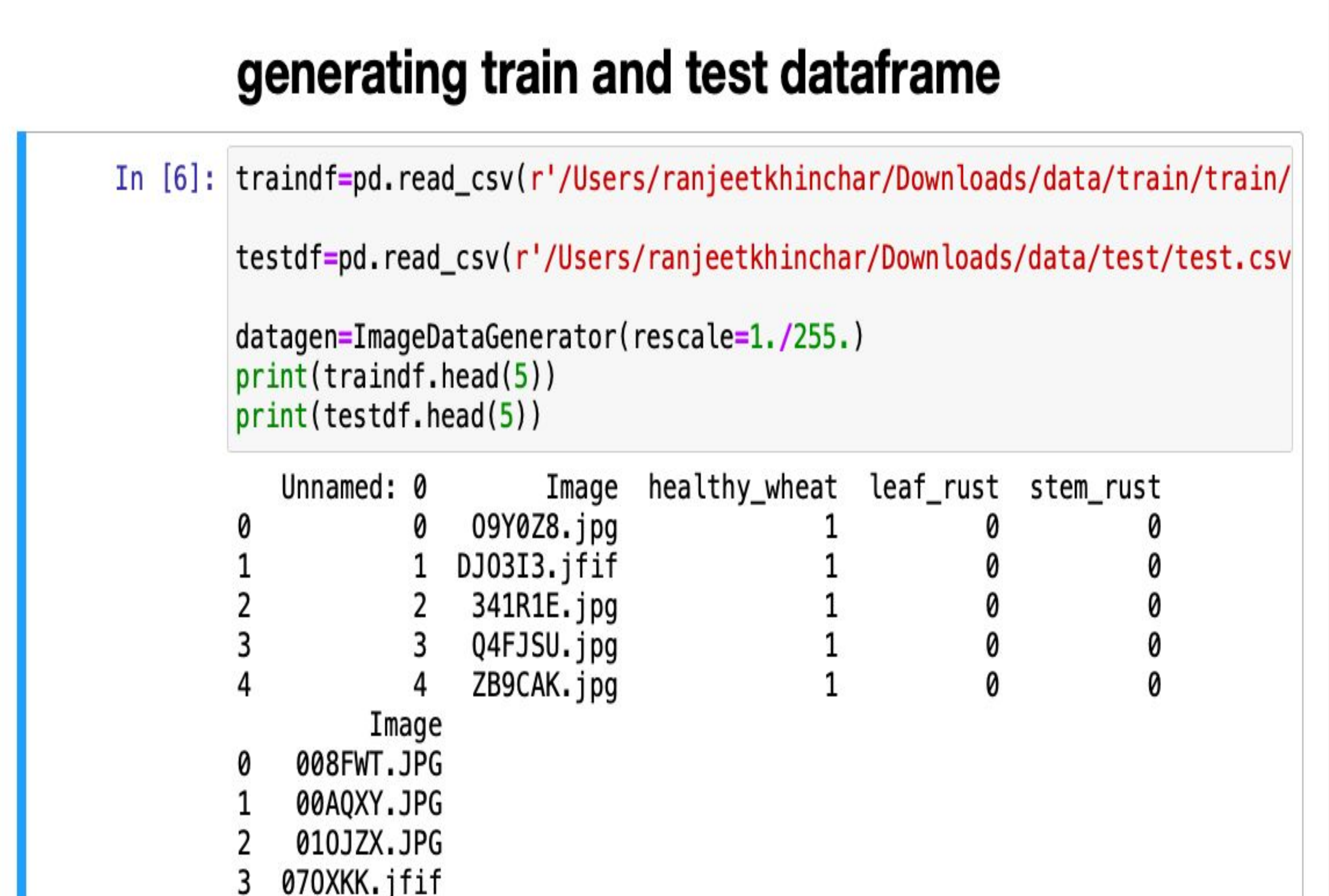
IMPORTING LIBRARIES



Importing all required libraries

In [5]:

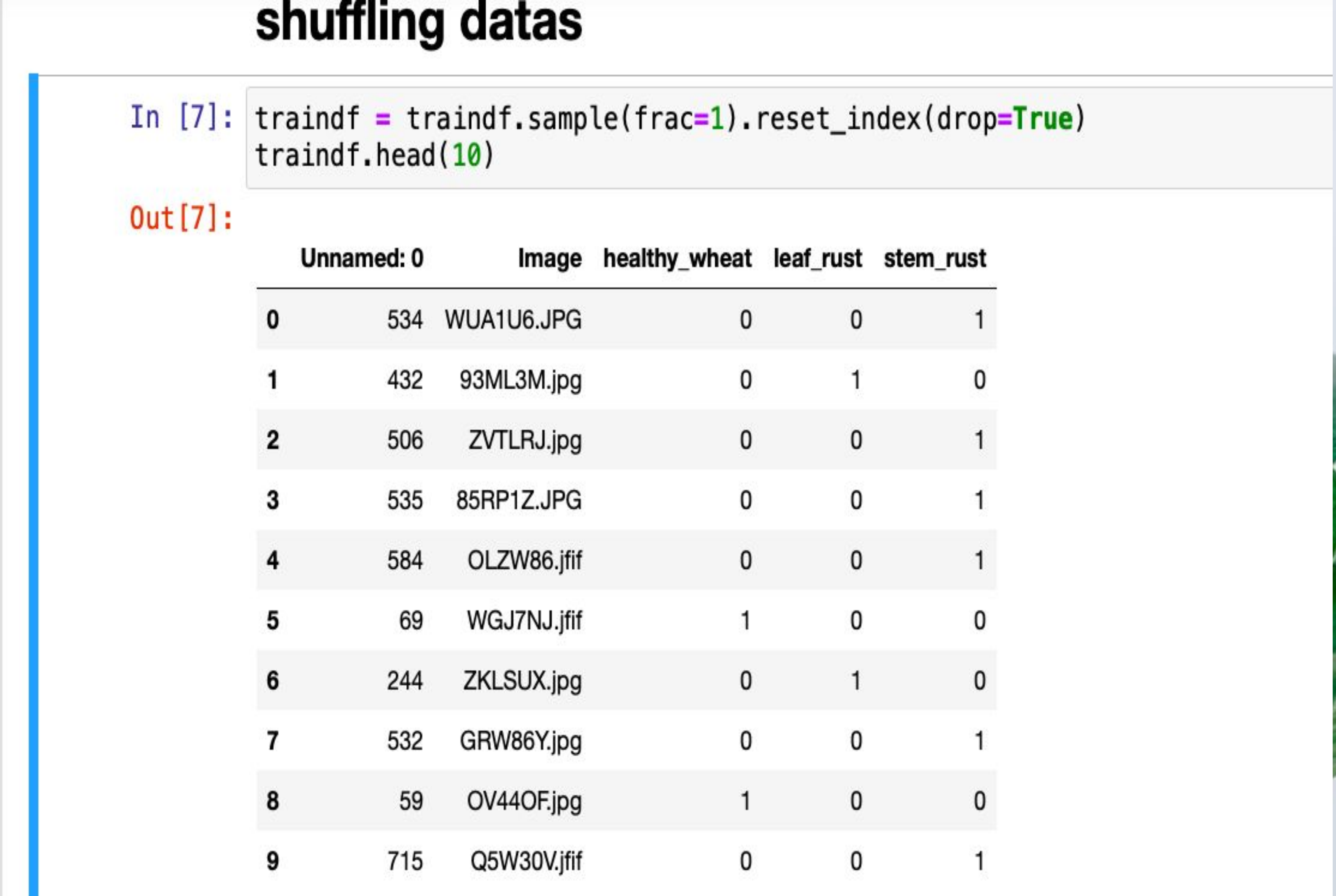
```
from keras.models import Sequential
from keras_preprocessing.image import ImageDataGenerator
from keras import regularizers, optimizers
import pandas as pd
import numpy as np
from keras.layers import Dense, Activation, Flatten, Dropout, BatchNorma
from keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras import layers
import tensorflow as tf
```



generating train and test dataframe

```
In [6]: traindf=pd.read_csv(r'/Users/ranjeetkhinchar/Downloads/data/train/train/

        testdf=pd.read_csv(r'/Users/ranjeetkhinchar/Downloads/data/test/test.csv

        datagen=ImageDataGenerator(rescale=1./255.)
        print(traindf.head(5))
        print(testdf.head(5))
```

```
   Unnamed: 0        Image  healthy_wheat  leaf_rust  stem_rust
0           0   O9Y0Z8.jpg              1          0          0
1           1  DJO3I3.jfif              1          0          0
2           2   341R1E.jpg              1          0          0
3           3   Q4FJSU.jpg              1          0          0
4           4   ZB9CAK.jpg              1          0          0
         Image
0   008FWT.JPG
1   00AQXY.JPG
2   01OJZX.JPG
3  07OXKK.jfif
```



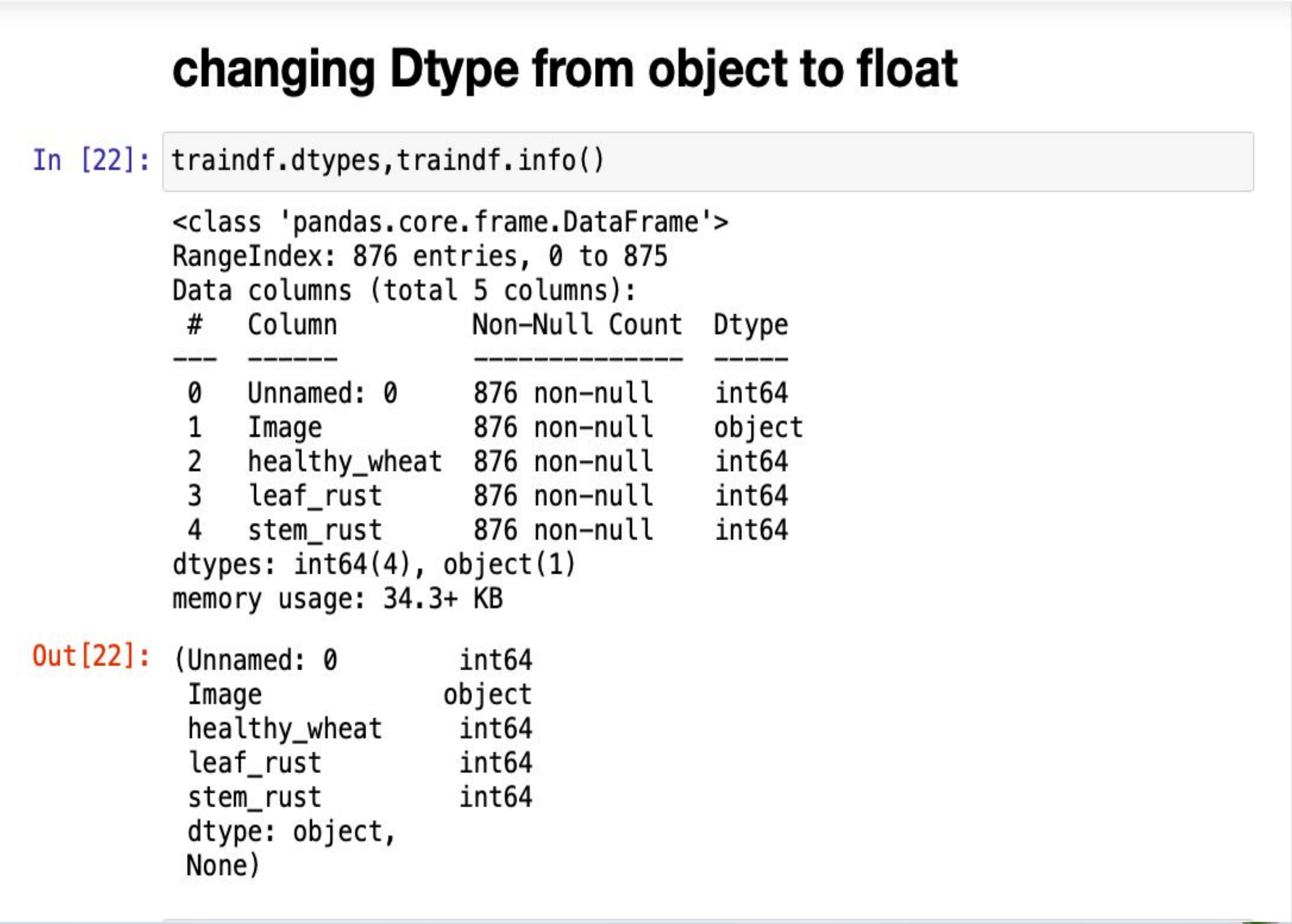
shuffling datas

In [7]:

```
traindf = traindf.sample(frac=1).reset_index(drop=True)
traindf.head(10)
```

Out[7]:

	Unnamed: 0	Image	healthy_wheat	leaf_rust	stem_rust
0	534	WUA1U6.JPG	0	0	1
1	432	93ML3M.jpg	0	1	0
2	506	ZVTLRJ.jpg	0	0	1
3	535	85RP1Z.JPG	0	0	1
4	584	OLZW86.jfif	0	0	1
5	69	WGJ7NJ.jfif	1	0	0
6	244	ZKLSUX.jpg	0	1	0
7	532	GRW86Y.jpg	0	0	1
8	59	OV44OF.jpg	1	0	0
9	715	Q5W30V.jfif	0	0	1

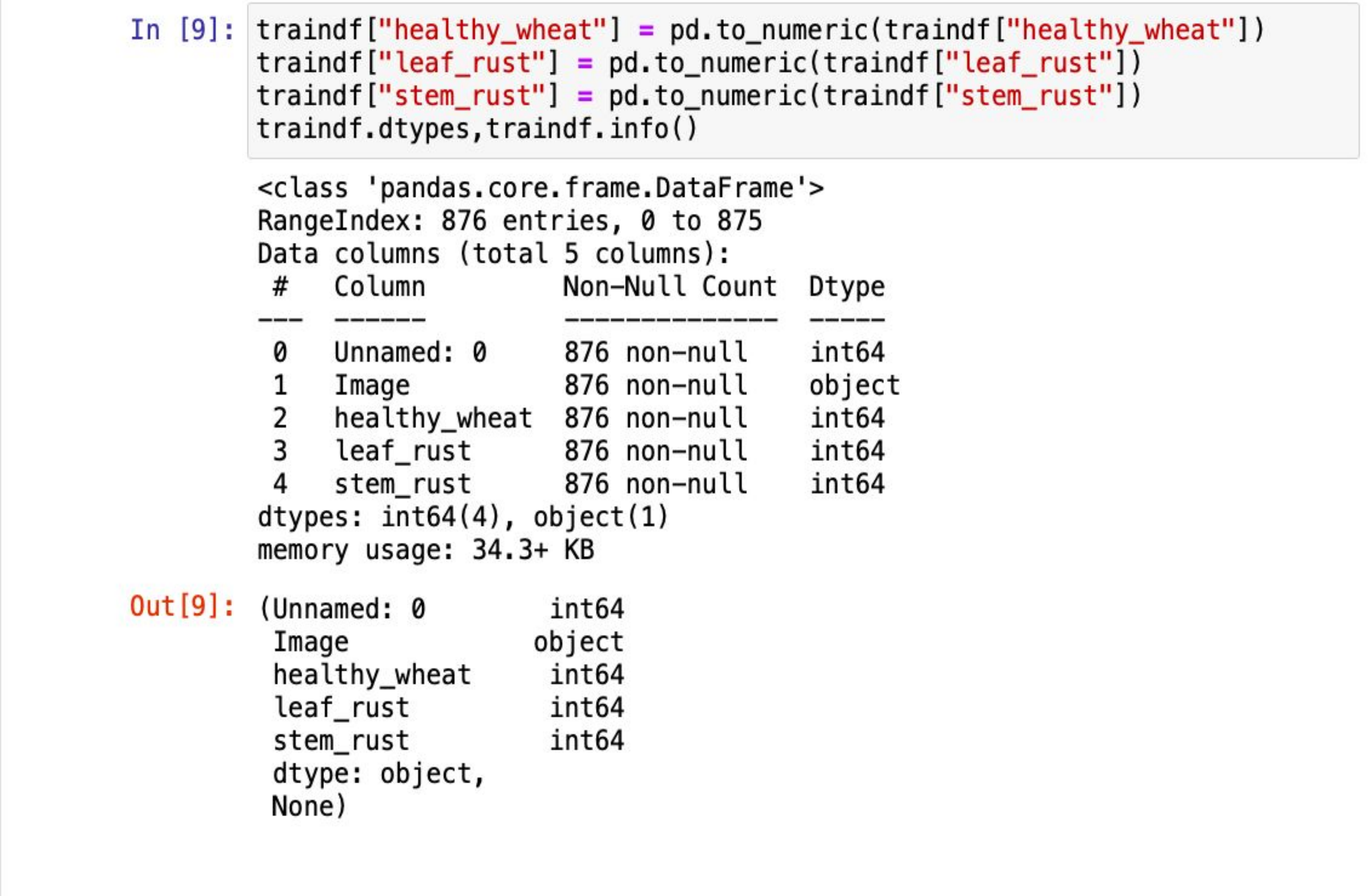


changing Dtype from object to float

```
In [22]: traindf.dtypes,traindf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 876 entries, 0 to 875
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   Unnamed: 0    876 non-null    int64 
 1   Image         876 non-null    object
 2   healthy_wheat 876 non-null    int64 
 3   leaf_rust     876 non-null    int64 
 4   stem_rust     876 non-null    int64 
dtypes: int64(4), object(1)
memory usage: 34.3+ KB
```

```
Out[22]: (Unnamed: 0        int64
          Image            object
          healthy_wheat     int64
          leaf_rust         int64
          stem_rust         int64
          dtype: object,
          None)
```

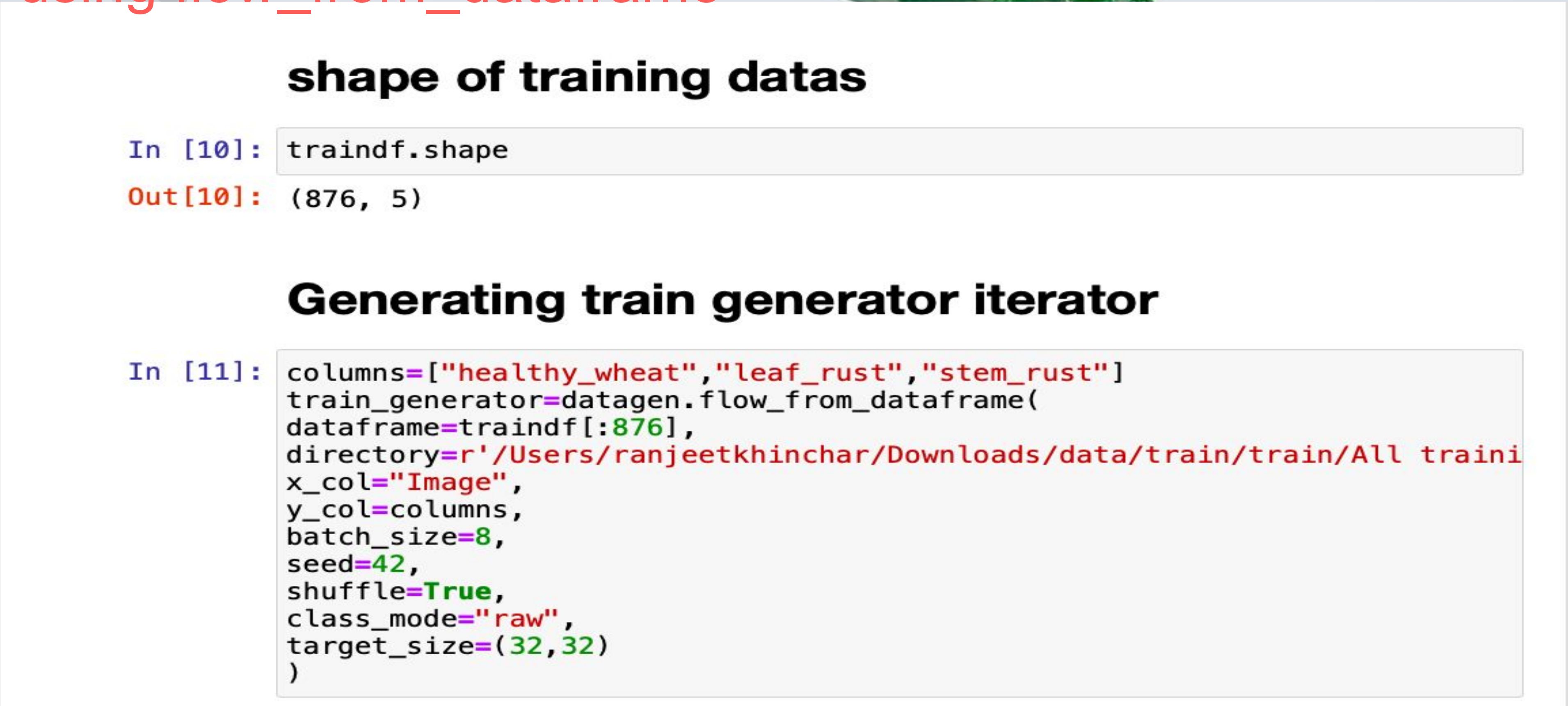


```
In [9]: traindf["healthy_wheat"] = pd.to_numeric(traindf["healthy_wheat"])
        traindf["leaf_rust"] = pd.to_numeric(traindf["leaf_rust"])
        traindf["stem_rust"] = pd.to_numeric(traindf["stem_rust"])
        traindf.dtypes,traindf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 876 entries, 0 to 875
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   Unnamed: 0     876 non-null    int64 
 1   Image          876 non-null    object
 2   healthy_wheat  876 non-null    int64 
 3   leaf_rust      876 non-null    int64 
 4   stem_rust      876 non-null    int64 
dtypes: int64(4), object(1)
memory usage: 34.3+ KB
```

```
Out[9]: (Unnamed: 0        int64
         Image            object
         healthy_wheat     int64
         leaf_rust         int64
         stem_rust         int64
         dtype: object,
         None)
```

Creating training and validating datasets using flow_from_dataframe



shape of training datas

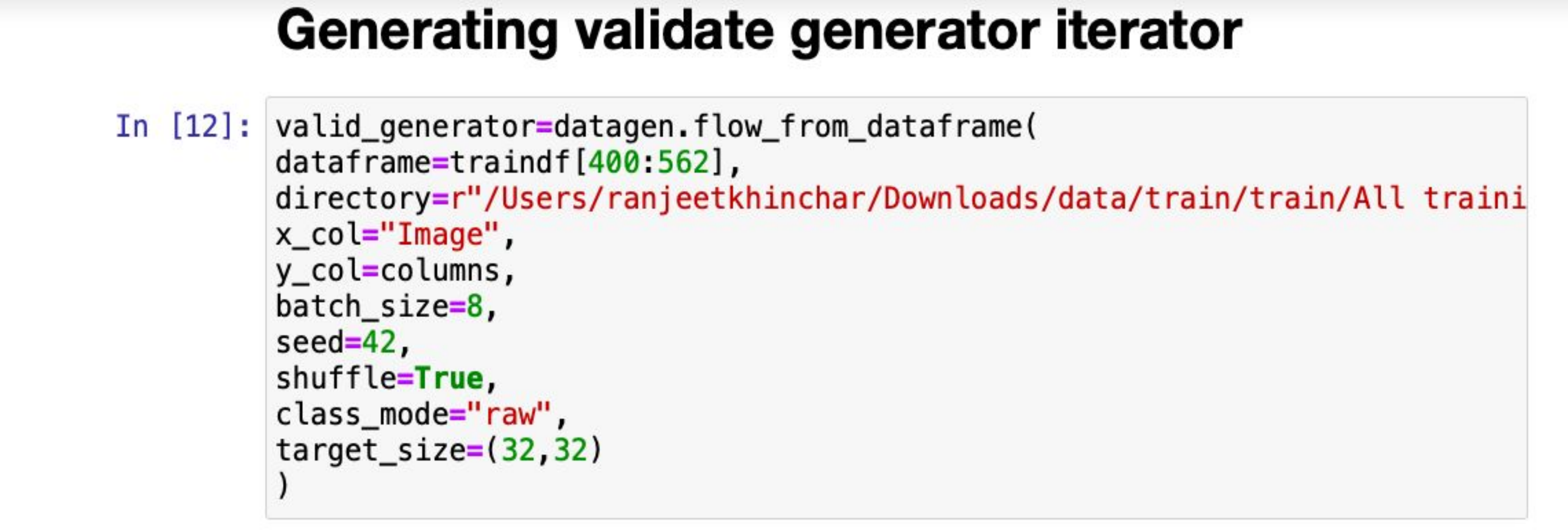
In [10]:
```
traindf.shape
```

Out[10]:
```
(876, 5)
```

Generating train generator iterator

In [11]:
```
columns=["healthy_wheat","leaf_rust","stem_rust"]
train_generator=datagen.flow_from_dataframe(
dataframe=traindf[:876],
directory=r'/Users/ranjeetkhinchar/Downloads/data/train/train/All traini
x_col="Image",
y_col=columns,
batch_size=8,
seed=42,
shuffle=True,
class_mode="raw",
target_size=(32,32)
)
```

Creating training and validating datasets using flow_from_dataframe

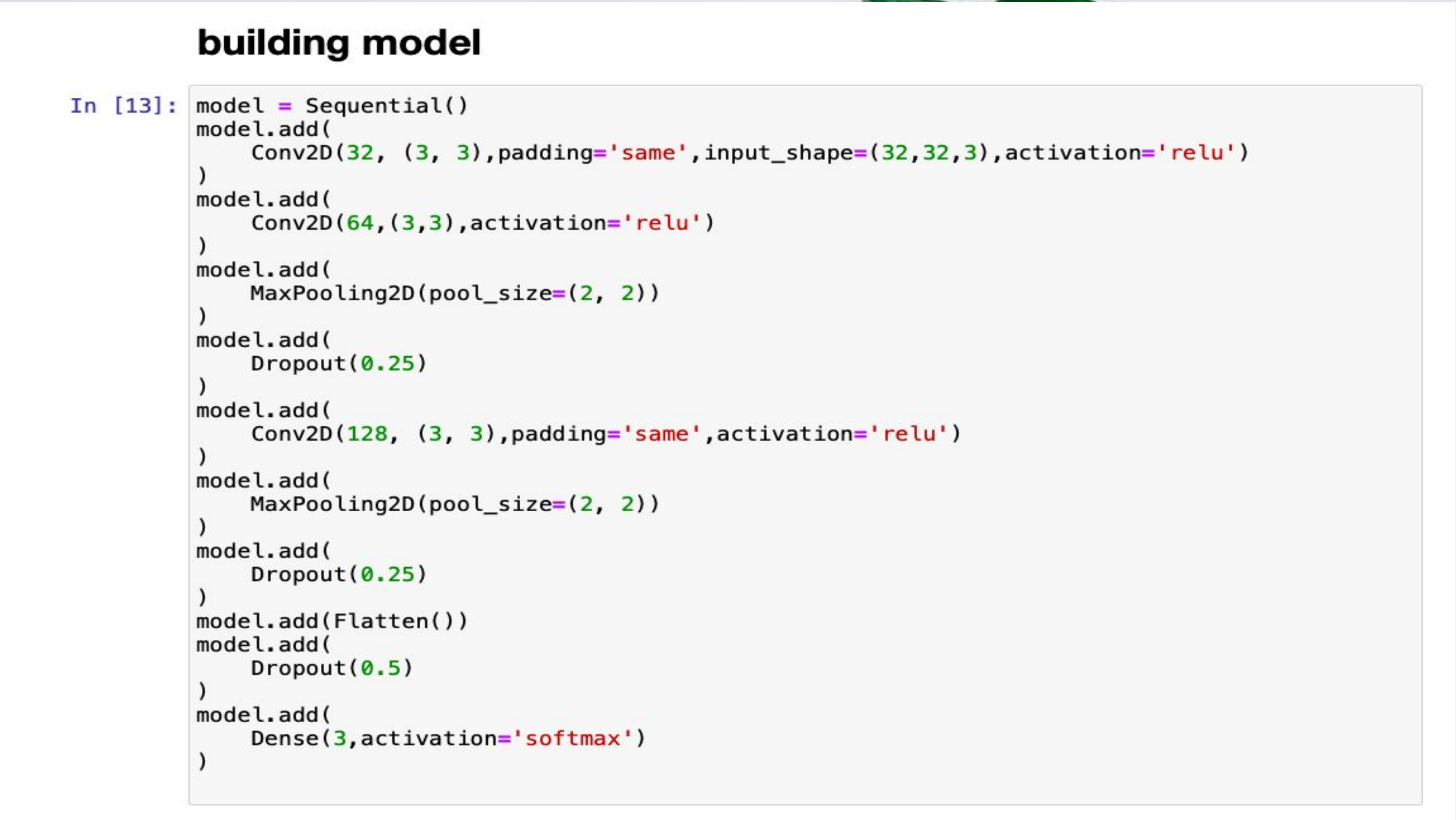


Generating validate generator iterator

In [12]:

```
valid_generator=datagen.flow_from_dataframe(
dataframe=traindf[400:562],
directory=r"/Users/ranjeetkhinchar/Downloads/data/train/train/All traini
x_col="Image",
y_col=columns,
batch_size=8,
seed=42,
shuffle=True,
class_mode="raw",
target_size=(32,32)
)
```

SEQUENTIAL MODEL

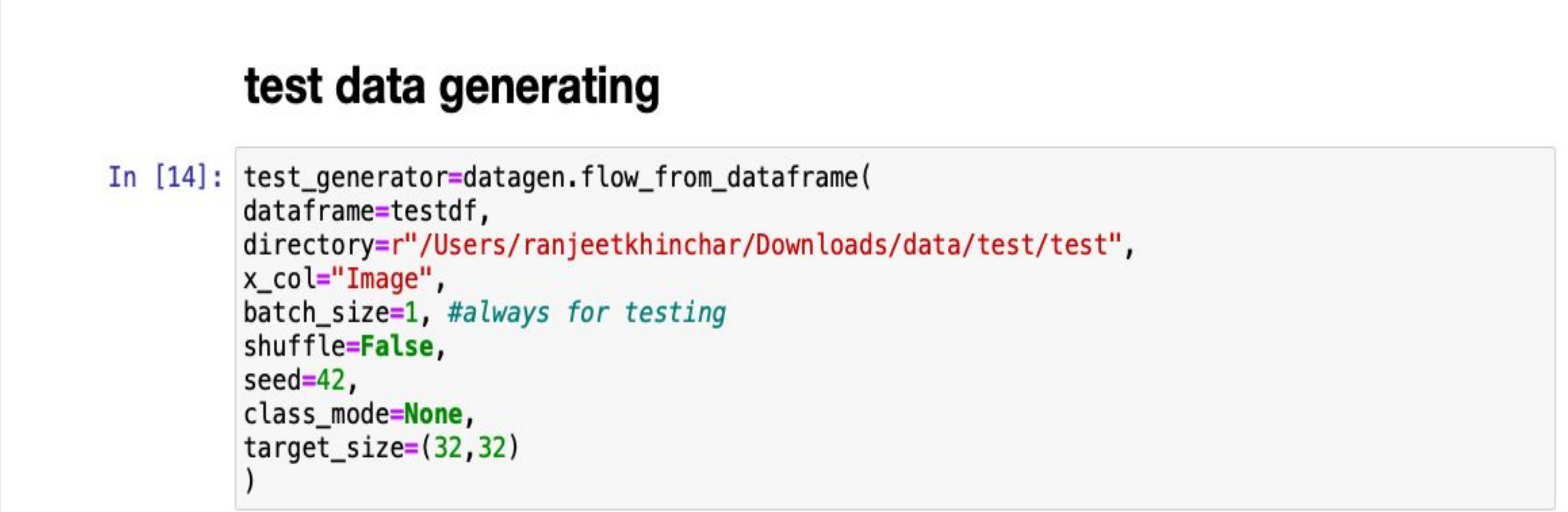


building model

In [13]:

```
model = Sequential()
model.add(
    Conv2D(32, (3, 3),padding='same',input_shape=(32,32,3),activation='relu')
)
model.add(
    Conv2D(64,(3,3),activation='relu')
)
model.add(
    MaxPooling2D(pool_size=(2, 2))
)
model.add(
    Dropout(0.25)
)
model.add(
    Conv2D(128, (3, 3),padding='same',activation='relu')
)
model.add(
    MaxPooling2D(pool_size=(2, 2))
)
model.add(
    Dropout(0.25)
)
model.add(Flatten())
model.add(
    Dropout(0.5)
)
model.add(
    Dense(3,activation='softmax')
)
```

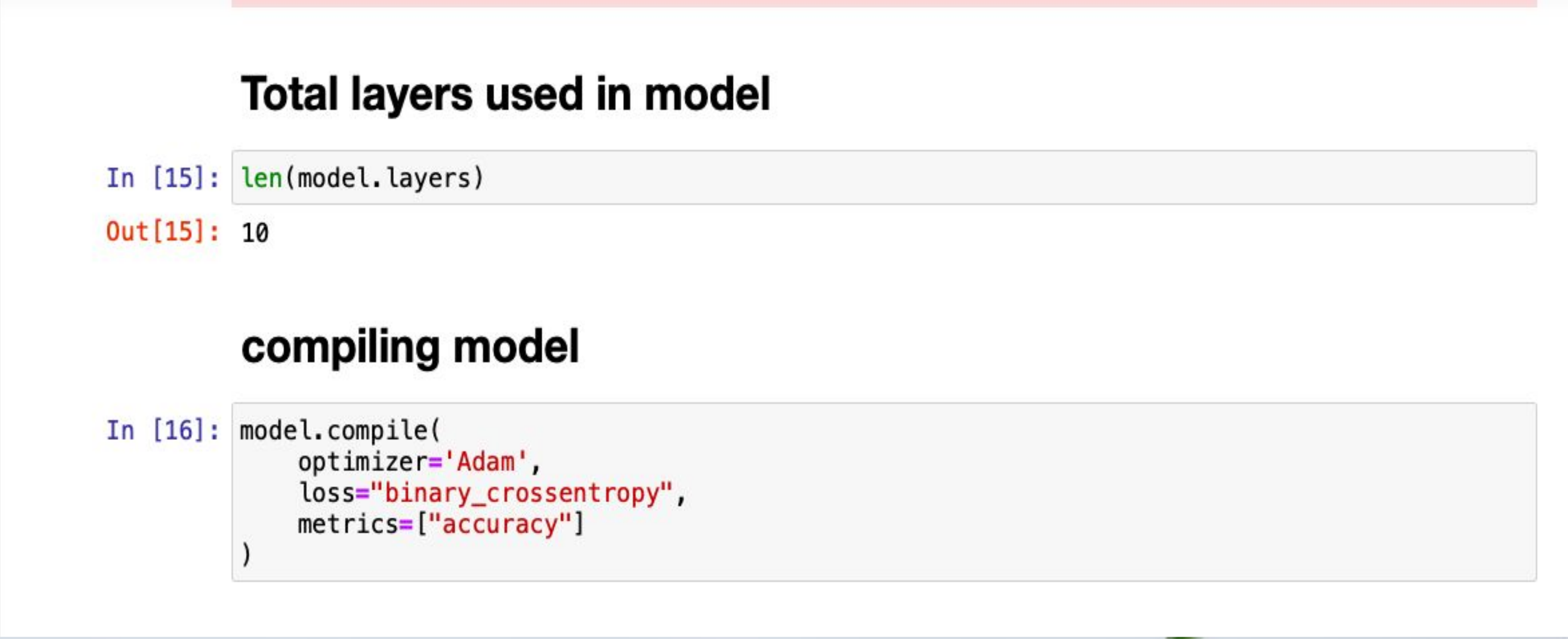
Generate test data



test data generating

```
In [14]: test_generator=datagen.flow_from_dataframe(
         dataframe=testdf,
         directory=r"/Users/ranjeetkhinchar/Downloads/data/test/test",
         x_col="Image",
         batch_size=1, #always for testing
         shuffle=False,
         seed=42,
         class_mode=None,
         target_size=(32,32)
         )
```

Total no of layers and Compiling model

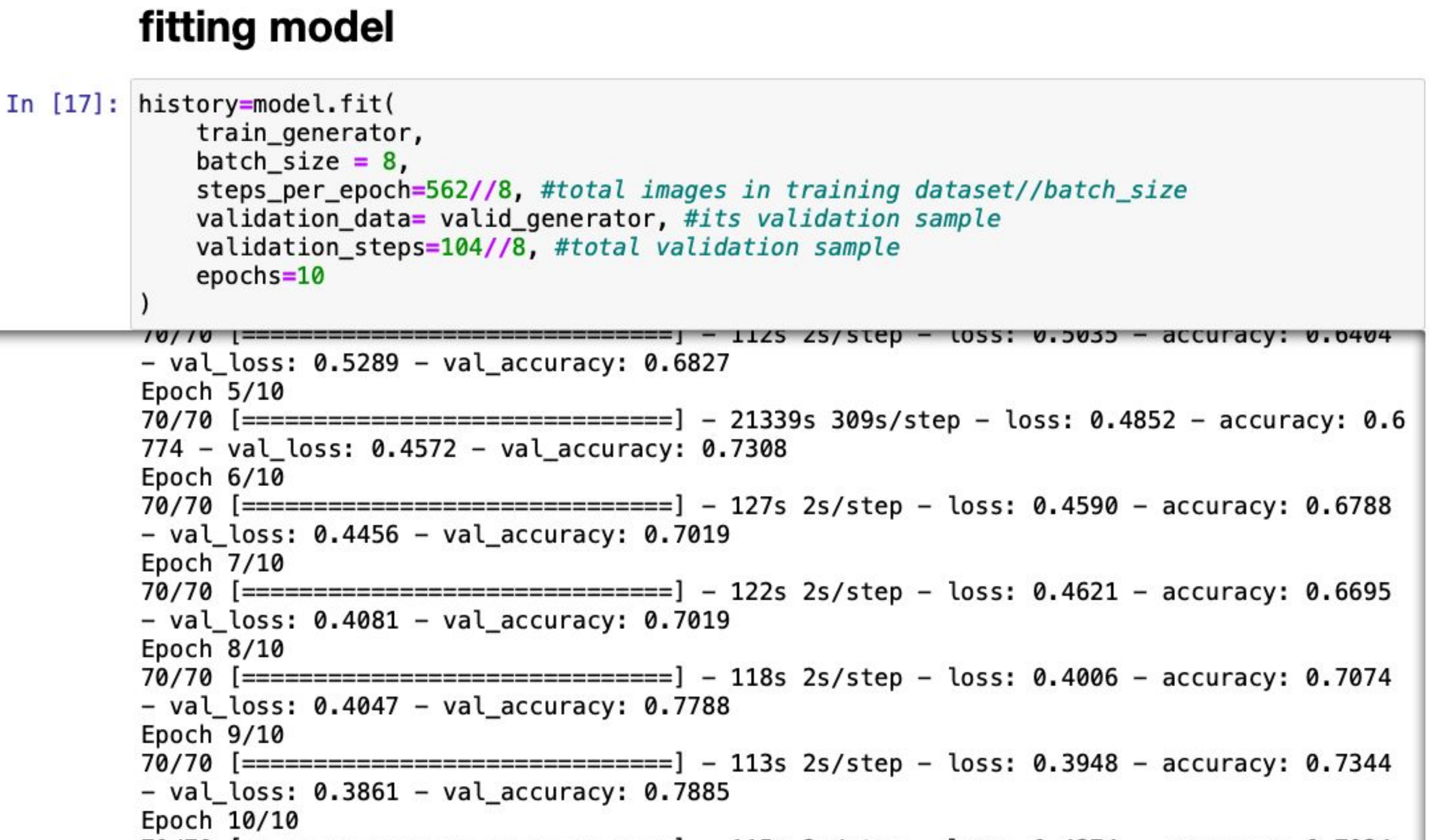


Total layers used in model

```
In [15]: len(model.layers)

Out[15]: 10
```

compiling model

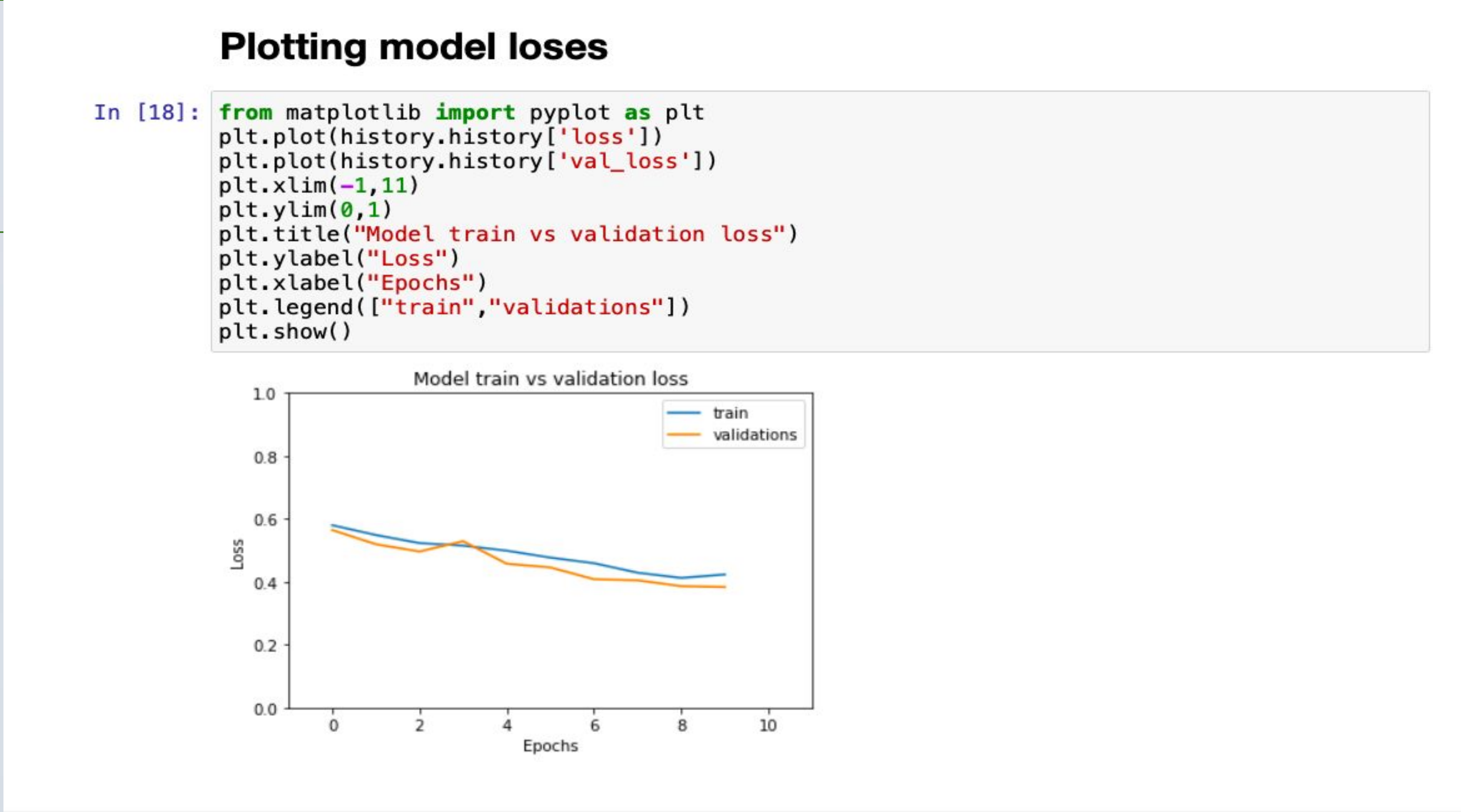
```
In [16]: model.compile(
             optimizer='Adam',
             loss="binary_crossentropy",
             metrics=["accuracy"]
         )
```



fitting model

In [17]:
```
history=model.fit(
    train_generator,
    batch_size = 8,
    steps_per_epoch=562//8, #total images in training dataset//batch_size
    validation_data= valid_generator, #its validation sample
    validation_steps=104//8, #total validation sample
    epochs=10
)
```

```
70/70 [==============================] - 112s 2s/step - loss: 0.5035 - accuracy: 0.6404
- val_loss: 0.5289 - val_accuracy: 0.6827
Epoch 5/10
70/70 [==============================] - 21339s 309s/step - loss: 0.4852 - accuracy: 0.6
774 - val_loss: 0.4572 - val_accuracy: 0.7308
Epoch 6/10
70/70 [==============================] - 127s 2s/step - loss: 0.4590 - accuracy: 0.6788
- val_loss: 0.4456 - val_accuracy: 0.7019
Epoch 7/10
70/70 [==============================] - 122s 2s/step - loss: 0.4621 - accuracy: 0.6695
- val_loss: 0.4081 - val_accuracy: 0.7019
Epoch 8/10
70/70 [==============================] - 118s 2s/step - loss: 0.4006 - accuracy: 0.7074
- val_loss: 0.4047 - val_accuracy: 0.7788
Epoch 9/10
70/70 [==============================] - 113s 2s/step - loss: 0.3948 - accuracy: 0.7344
- val_loss: 0.3861 - val_accuracy: 0.7885
Epoch 10/10
```



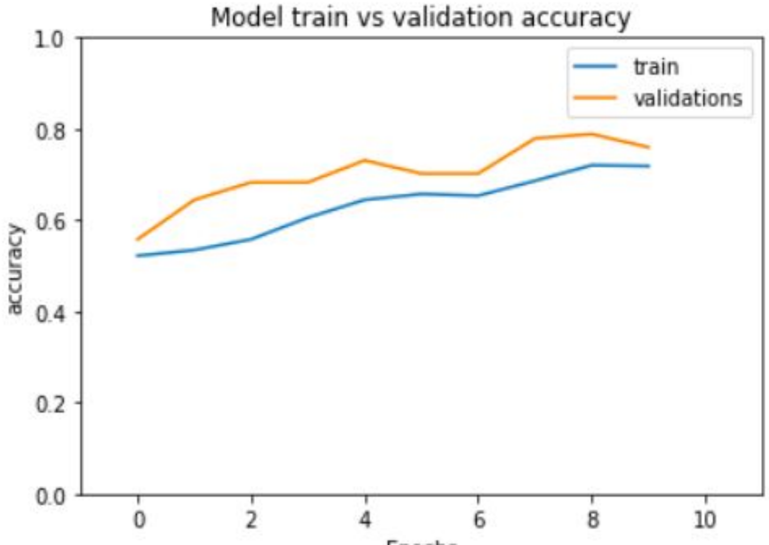
Plotting model loses

In [18]:

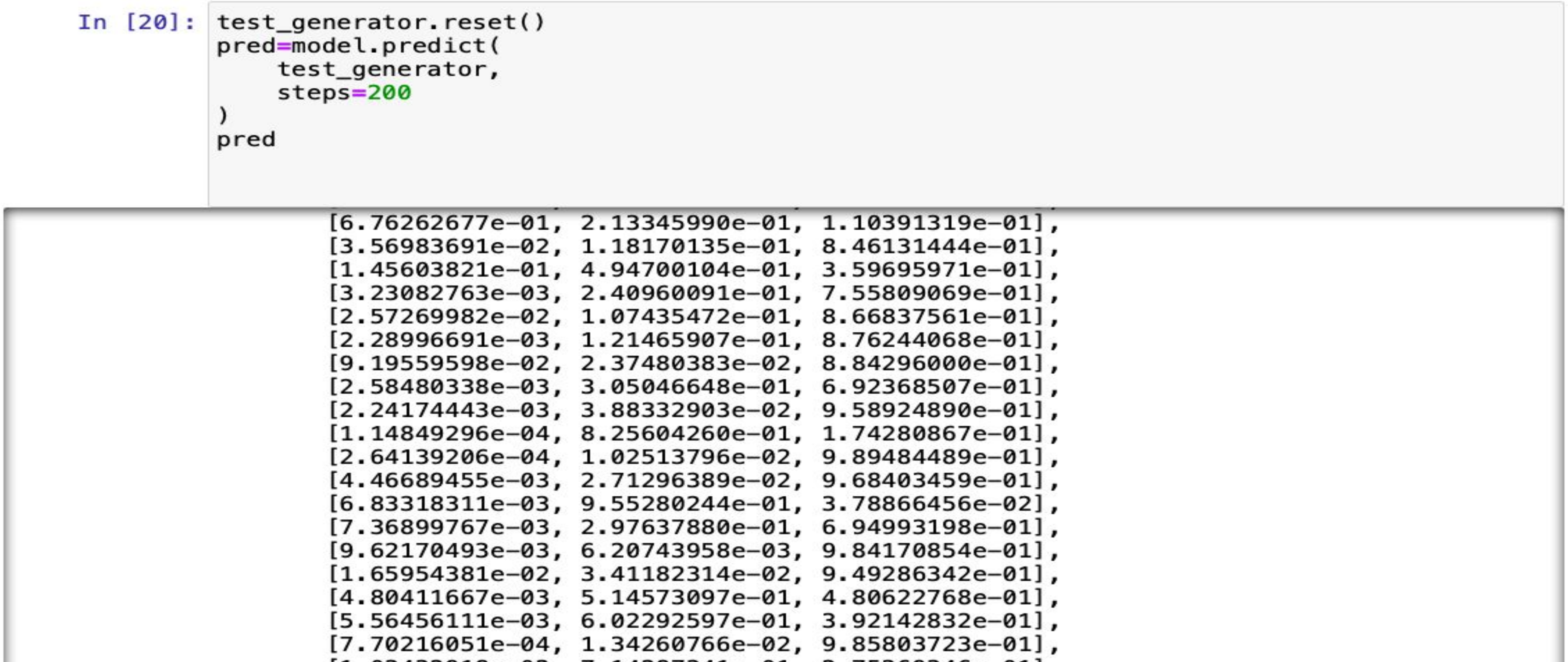
```
from matplotlib import pyplot as plt
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.xlim(-1,11)
plt.ylim(0,1)
plt.title("Model train vs validation loss")
plt.ylabel("Loss")
plt.xlabel("Epochs")
plt.legend(["train","validations"])
plt.show()
```

Plotting model Accuracy

In [19]:
```
from matplotlib import pyplot as plt
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.xlim(-1,11)
plt.ylim(0,1)
plt.title("Model train vs validation accuracy")
plt.ylabel("accuracy")
plt.xlabel("Epochs")
plt.legend(["train","validations"])
plt.show()
```



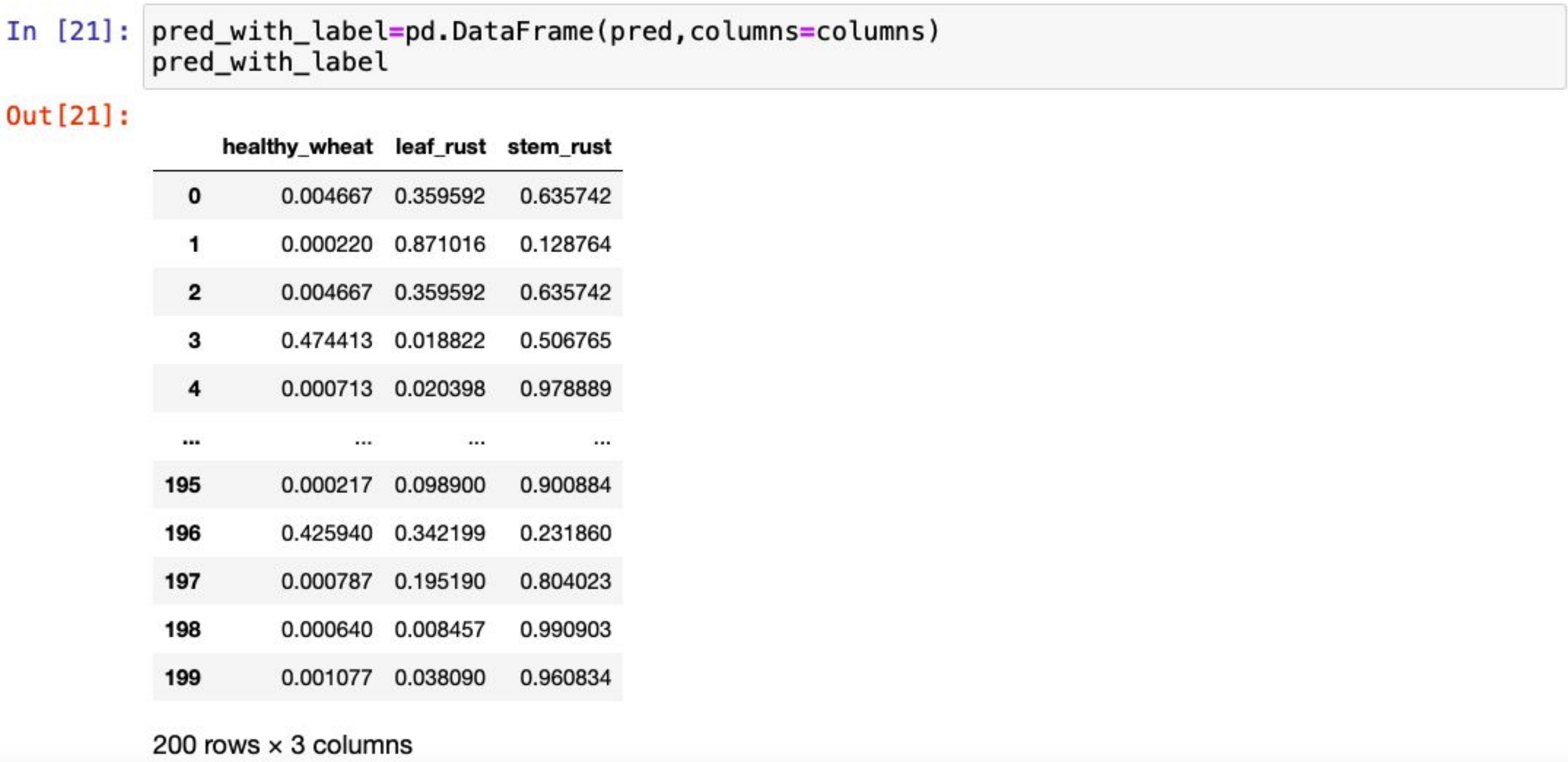
Prediction probabilities



```
In [20]: test_generator.reset()
         pred=model.predict(
              test_generator,
              steps=200
         )
         pred
```

```
[6.76262677e-01, 2.13345990e-01, 1.10391319e-01],
[3.56983691e-02, 1.18170135e-01, 8.46131444e-01],
[1.45603821e-01, 4.94700104e-01, 3.59695971e-01],
[3.23082763e-03, 2.40960091e-01, 7.55809069e-01],
[2.57269982e-02, 1.07435472e-01, 8.66837561e-01],
[2.28996691e-03, 1.21465907e-01, 8.76244068e-01],
[9.19559598e-02, 2.37480383e-02, 8.84296000e-01],
[2.58480338e-03, 3.05046648e-01, 6.92368507e-01],
[2.24174443e-03, 3.88332903e-02, 9.58924890e-01],
[1.14849296e-04, 8.25604260e-01, 1.74280867e-01],
[2.64139206e-04, 1.02513796e-02, 9.89484489e-01],
[4.46689455e-03, 2.71296389e-02, 9.68403459e-01],
[6.83318311e-03, 9.55280244e-01, 3.78866456e-02],
[7.36899767e-03, 2.97637880e-01, 6.94993198e-01],
[9.62170493e-03, 6.20743958e-03, 9.84170854e-01],
[1.65954381e-02, 3.41182314e-02, 9.49286342e-01],
[4.80411667e-03, 5.14573097e-01, 4.80622768e-01],
[5.56456111e-03, 6.02292597e-01, 3.92142832e-01],
[7.70216051e-04, 1.34260766e-02, 9.85803723e-01],
```

Prediction with labels



```
In [21]: pred_with_label=pd.DataFrame(pred,columns=columns)
         pred_with_label
```

Out[21]:

	healthy_wheat	leaf_rust	stem_rust
0	0.004667	0.359592	0.635742
1	0.000220	0.871016	0.128764
2	0.004667	0.359592	0.635742
3	0.474413	0.018822	0.506765
4	0.000713	0.020398	0.978889
...	...	...	...
195	0.000217	0.098900	0.900884
196	0.425940	0.342199	0.231860
197	0.000787	0.195190	0.804023
198	0.000640	0.008457	0.990903
199	0.001077	0.038090	0.960834

200 rows × 3 columns

SOURCE OF DATA :

LINK :

https://www.kaggle.com/shadabhussain/cgiar-computer-vision-for-crop-disease

RESEARCH PAPER :

LINK : Jaware, T., Badgujar, R., Patil, G.: Crop disease detection using image segmentation2, 190–194 (April 2012)

THANK YOU