Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated with low confidence

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

# import librariesimport boto3, re, sys, math, json, os, sagemaker, urllib.requestfrom sagemaker import get\_execution\_roleimport numpy as npimport pandas as pdimport matplotlib.pyplot as pltfrom IPython.display import Imagefrom IPython.display import display

from time import gmtime, strftime

from sagemaker.predictor import csv\_serializer

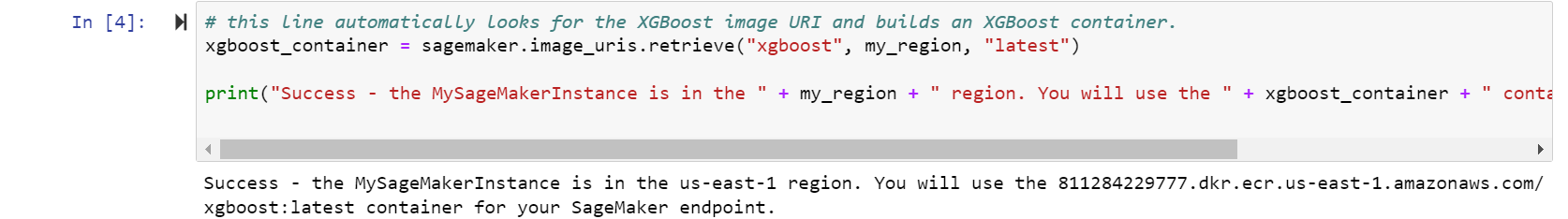


# Define IAM role

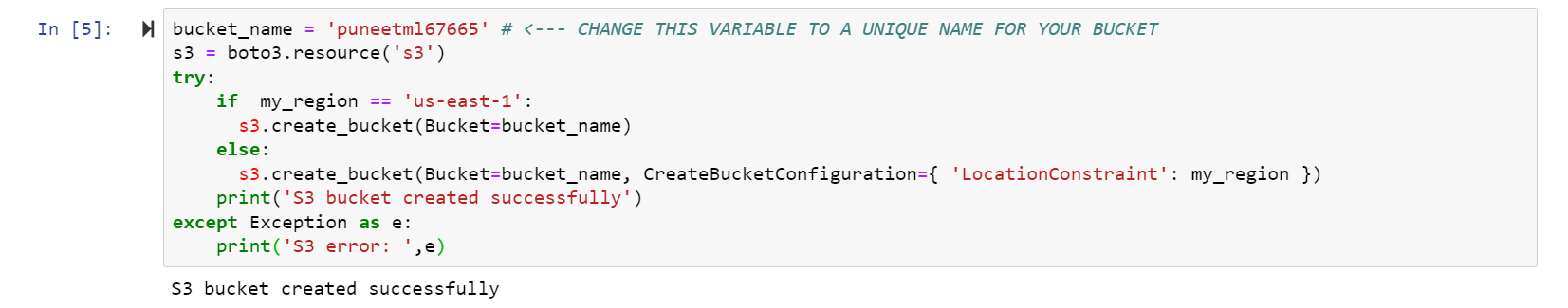
role = get\_execution\_role()

prefix = 'sagemaker/DEMO-xgboost-dm'

my\_region = boto3.session.Session().region\_name # set the region of the instance



# this line automatically looks for the XGBoost image URI and builds an XGBoost container.xgboost\_container = sagemaker.image\_uris.retrieve("xgboost", my\_region, "latest")print("Success - the MySageMakerInstance is in the " + my\_region + " region. You will use the " + xgboost\_container + " container for your SageMaker endpoint.")



bucket\_name = 'puneetml67665' # <--- CHANGE THIS VARIABLE TO A UNIQUE NAME FOR YOUR BUCKET

s3 = boto3.resource('s3')

try:

if my\_region == 'us-east-1':

s3.create\_bucket(Bucket=bucket\_name)

else:

s3.create\_bucket(Bucket=bucket\_name, CreateBucketConfiguration={ 'LocationConstraint': my\_region })

print('S3 bucket created successfully')

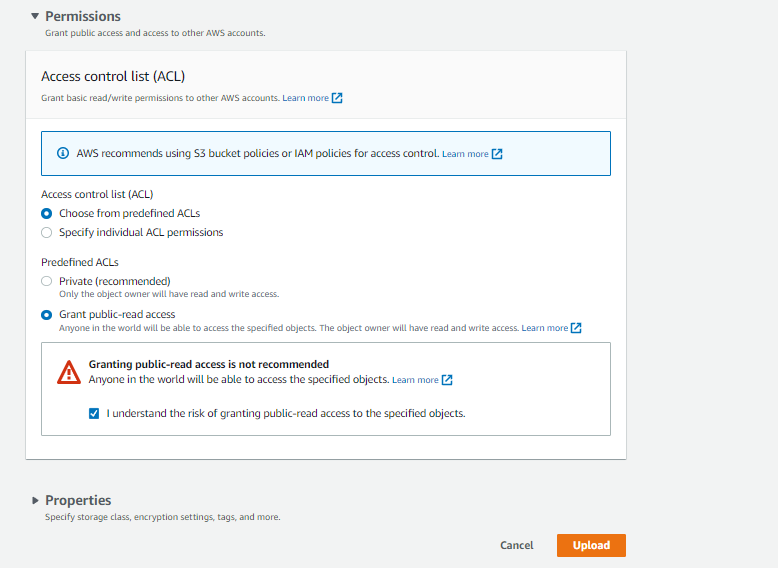
except Exception as e:

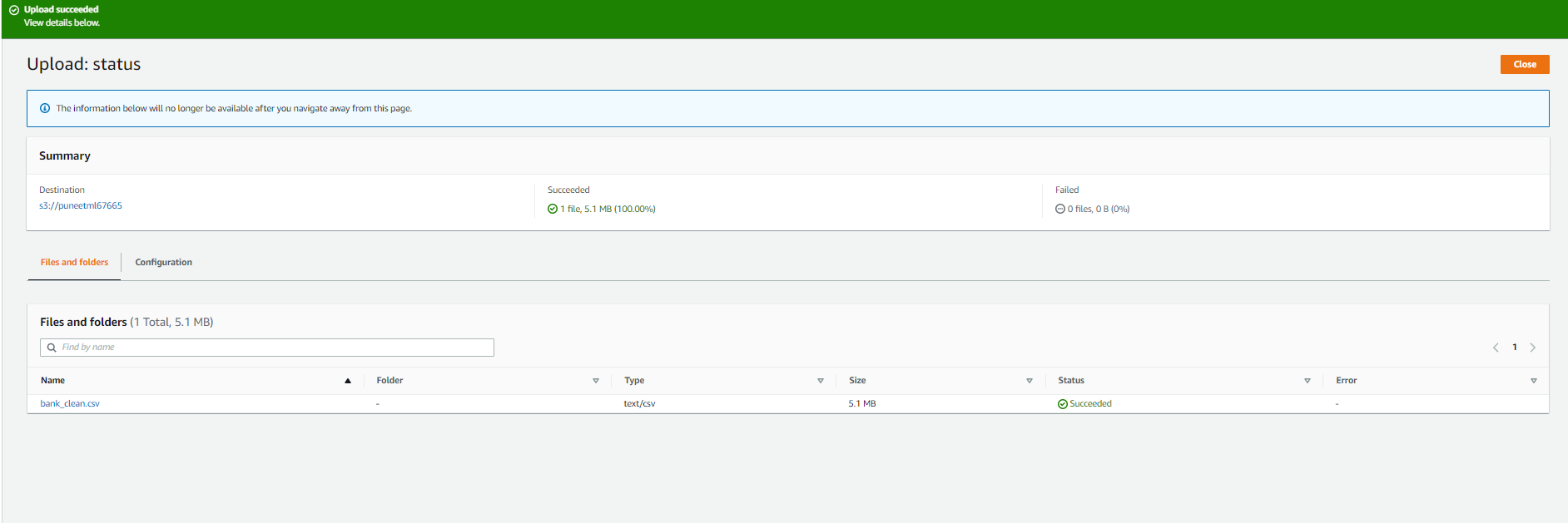
print('S3 error: ',e)

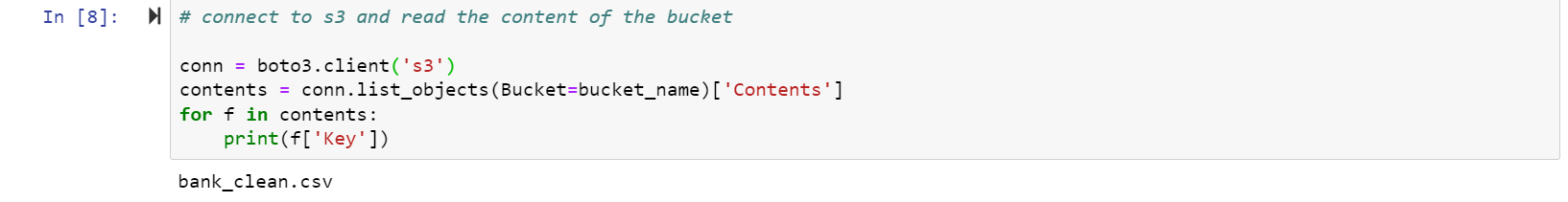


Graphical user interface, text, application, email

Description automatically generated









try: urllib.request.urlretrieve ("https://d1.awsstatic.com/tmt/build-train-deploy-machine-learning-model-sagemaker/bank\_clean.27f01fbbdf43271788427f3682996ae29ceca05d.csv", "bank\_clean.csv") print('Success: downloaded bank\_clean.csv.')except Exception as e:

print('Data load error: ',e)

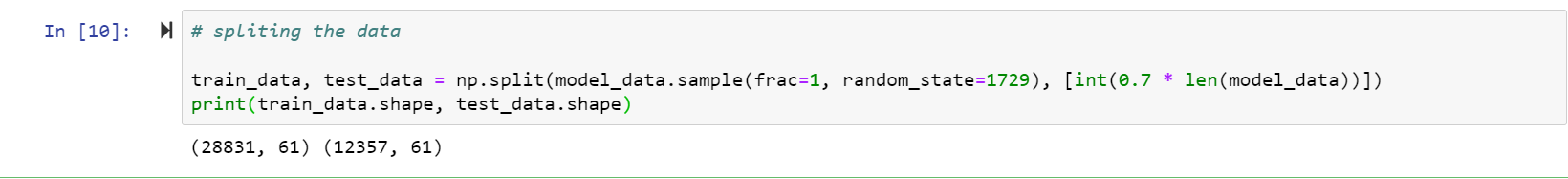
try:

model\_data = pd.read\_csv('./bank\_clean.csv',index\_col=0)

print('Success: Data loaded into dataframe.')

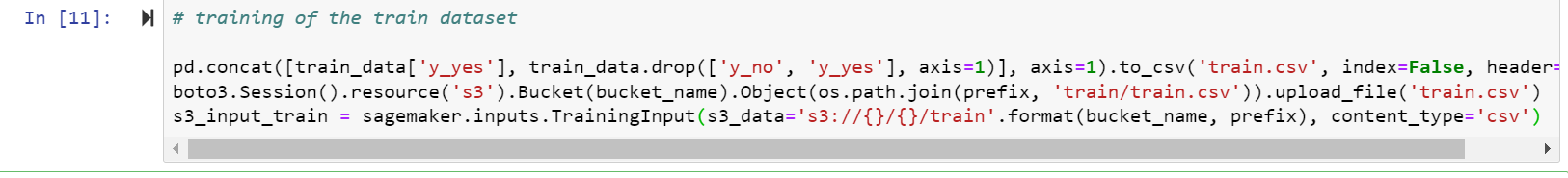
except Exception as e:

print('Data load error: ',e)



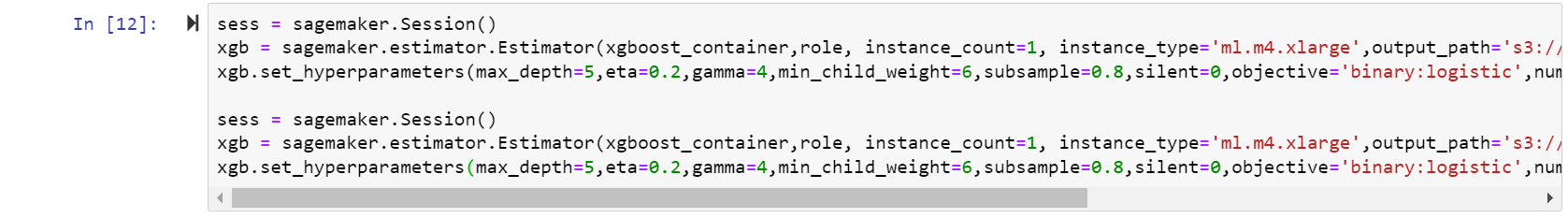
train\_data, test\_data = np.split(model\_data.sample(frac=1, random\_state=1729), [int(0.7 \* len(model\_data))])

print(train\_data.shape, test\_data.shape)



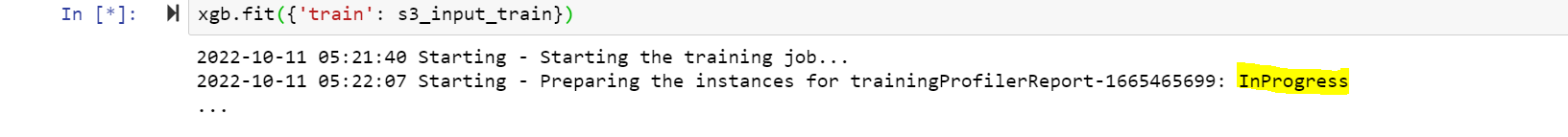
pd.concat([train\_data['y\_yes'], train\_data.drop(['y\_no', 'y\_yes'], axis=1)], axis=1).to\_csv('train.csv', index=False, header=False)boto3.Session().resource('s3').Bucket(bucket\_name).Object(os.path.join(prefix, 'train/train.csv')).upload\_file('train.csv')

s3\_input\_train = sagemaker.inputs.TrainingInput(s3\_data='s3://{}/{}/train'.format(bucket\_name, prefix), content\_type='csv')

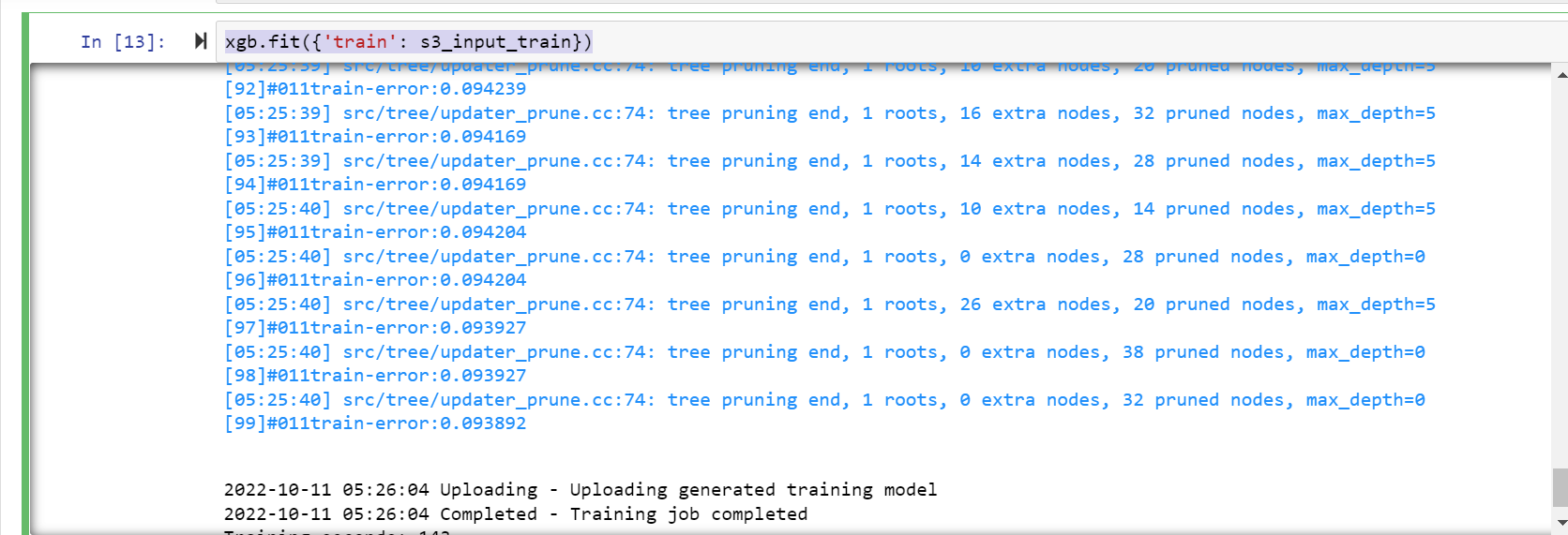


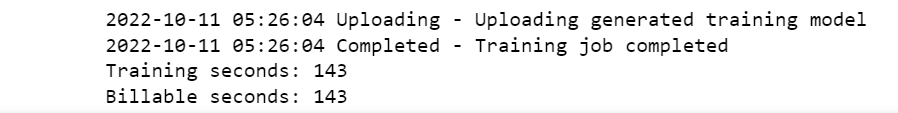
sess = sagemaker.Session()xgb = sagemaker.estimator.Estimator(xgboost\_container,role, instance\_count=1, instance\_type='ml.m4.xlarge',output\_path='s3://{}/{}/output'.format(bucket\_name, prefix),sagemaker\_session=sess)xgb.set\_hyperparameters(max\_depth=5,eta=0.2,gamma=4,min\_child\_weight=6,subsample=0.8,silent=0,objective='binary:logistic',num\_round=100)sess = sagemaker.Session()xgb = sagemaker.estimator.Estimator(xgboost\_container,role, instance\_count=1, instance\_type='ml.m4.xlarge',output\_path='s3://{}/{}/output'.format(bucket\_name, prefix),sagemaker\_session=sess)

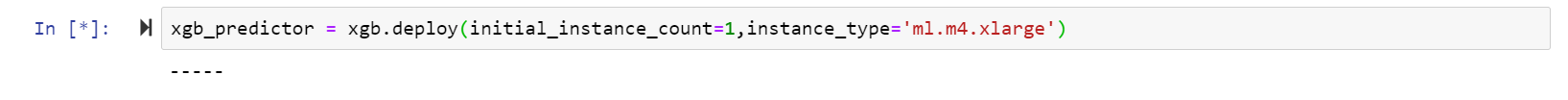
xgb.set\_hyperparameters(max\_depth=5,eta=0.2,gamma=4,min\_child\_weight=6,subsample=0.8,silent=0,objective='binary:logistic',num\_round=100)



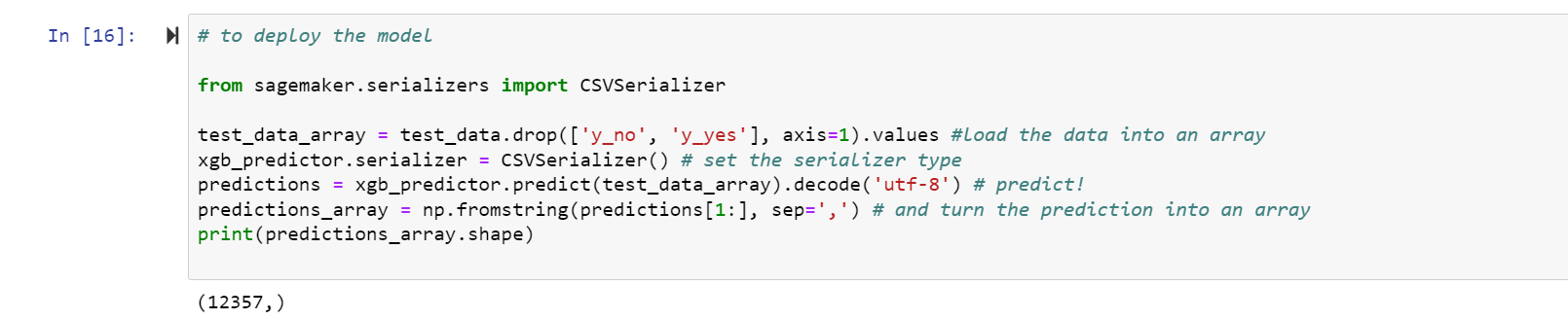
xgb.fit({'train': s3\_input\_train})







xgb\_predictor = xgb.deploy(initial\_instance\_count=1,instance\_type='ml.m4.xlarge')



# to deploy the model

from sagemaker.serializers import CSVSerializer

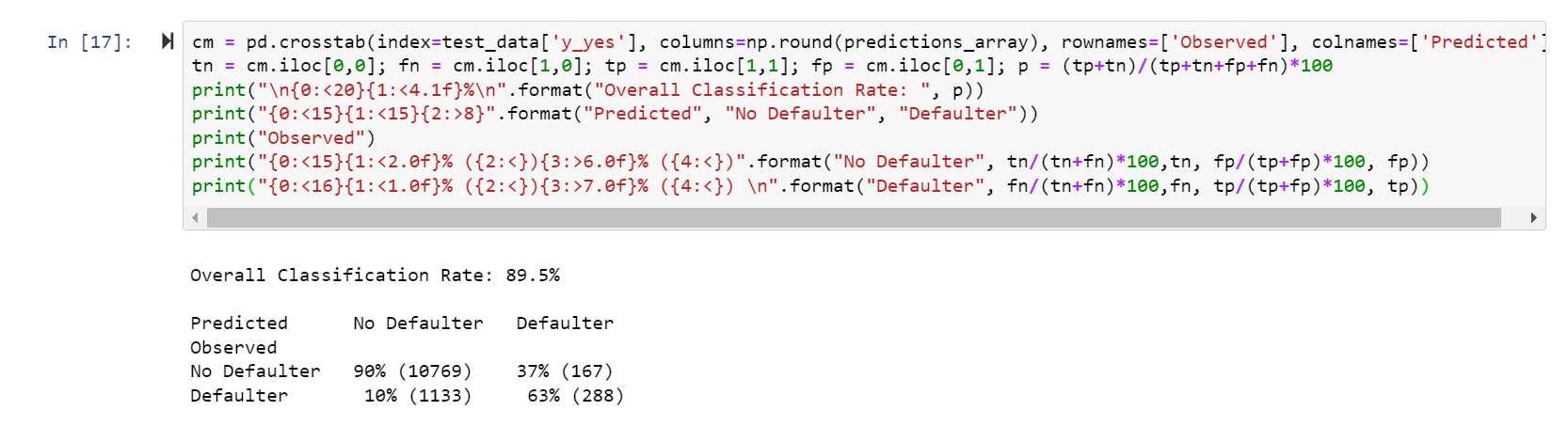
test\_data\_array = test\_data.drop(['y\_no', 'y\_yes'], axis=1).values #load the data into an array

xgb\_predictor.serializer = CSVSerializer() # set the serializer type

predictions = xgb\_predictor.predict(test\_data\_array).decode('utf-8') # predict!

predictions\_array = np.fromstring(predictions[1:], sep=',') # and turn the prediction into an array

print(predictions\_array.shape)



cm = pd.crosstab(index=test\_data['y\_yes'], columns=np.round(predictions\_array), rownames=['Observed'], colnames=['Predicted'])

tn = cm.iloc[0,0]; fn = cm.iloc[1,0]; tp = cm.iloc[1,1]; fp = cm.iloc[0,1]; p = (tp+tn)/(tp+tn+fp+fn)\*100

print("\n{0:<20}{1:<4.1f}%\n".format("Overall Classification Rate: ", p))

print("{0:<15}{1:<15}{2:>8}".format("Predicted", "No Defaulter", "Defaulter"))

print("Observed")

print("{0:<15}{1:<2.0f}% ({2:<}){3:>6.0f}% ({4:<})".format("No Defaulter", tn/(tn+fn)\*100,tn, fp/(tp+fp)\*100, fp))

print("{0:<16}{1:<1.0f}% ({2:<}){3:>7.0f}% ({4:<}) \n".format("Defaulter", fn/(tn+fn)\*100,fn, tp/(tp+fp)\*100, tp))