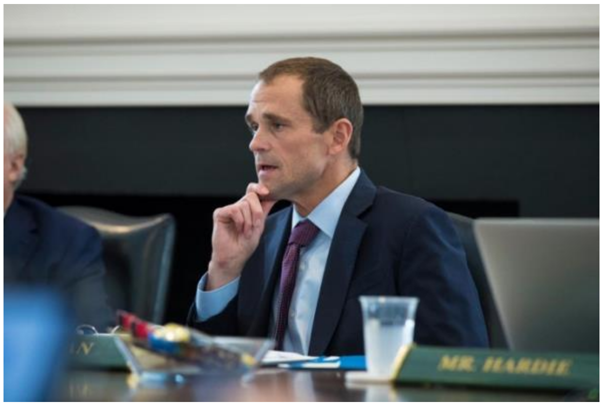
CS4740 Spring 2021 Cloud Computing PA#5

Name: Victoria Wang

UVa User ID: vxw6ta

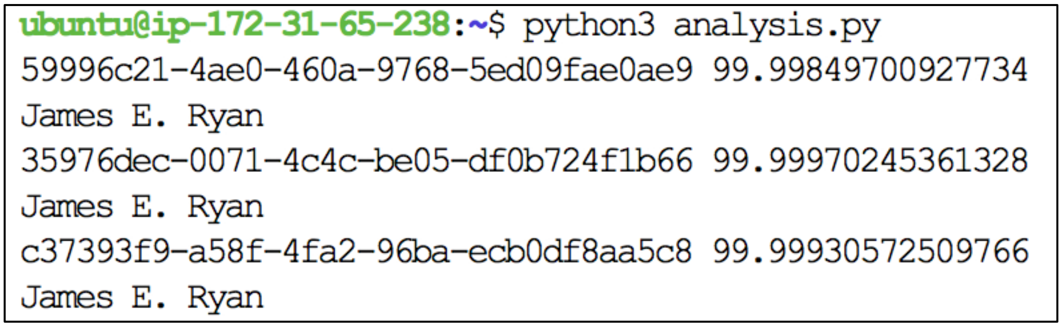
a. [Positive test case] A test picture (which you have not trained your system on) of one of the presidents AND a screenshot of your terminal showing your system processing the picture (showing that the system is invoked via the CLI, there was no run-time error, and the return to the command prompt.)

Test Picture



jim_ryan_at_bov_da_header_3-2.jpg

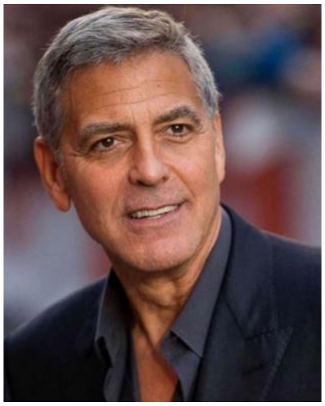
Terminal



```
ubuntu@ip-172-31-65-238:~$ python3 analysis.py
59996c21-4ae0-460a-9768-5ed09fae0ae9 99.99849700927734
James E. Ryan
35976dec-0071-4c4c-be05-df0b724f1b66 99.99970245361328
James E. Ryan
c37393f9-a58f-4fa2-96ba-ecb0df8aa5c8 99.99930572509766
James E. Ryan
```

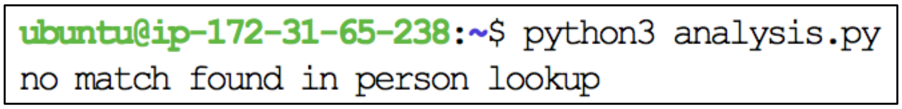
b. [Negative test case(s)] The test picture AND a screenshot of your terminal showing your system processing the picture. Note: the intent of this is to show your boss that your system responds correctly when presented with a picture that it should NOT claim is one of the two people it SHOULD recognize. It is up to you how many pictures, what each picture is, etc., is warranted so that you can convince your boss.

Test Picture 1



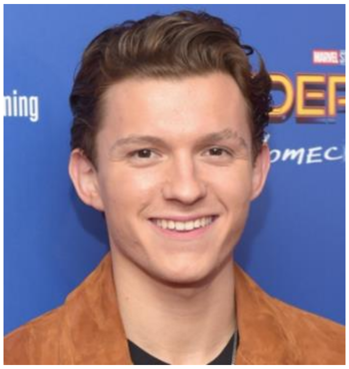
latest?cb=20191217011451

Terminal 1



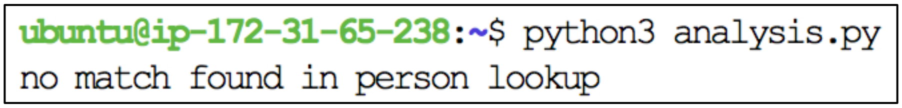
```
ubuntu@ip-172-31-65-238:~$ python3 analysis.py
no match found in person lookup
```

Test Picture 2



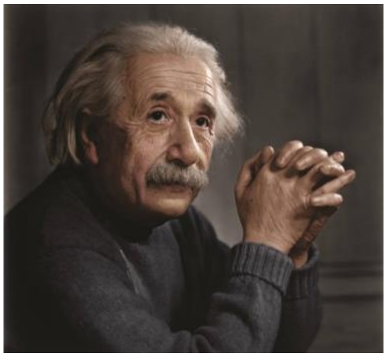
tom-holland-photo-jason-kempin-getty-images-801510482-profile.jpg

Terminal 2



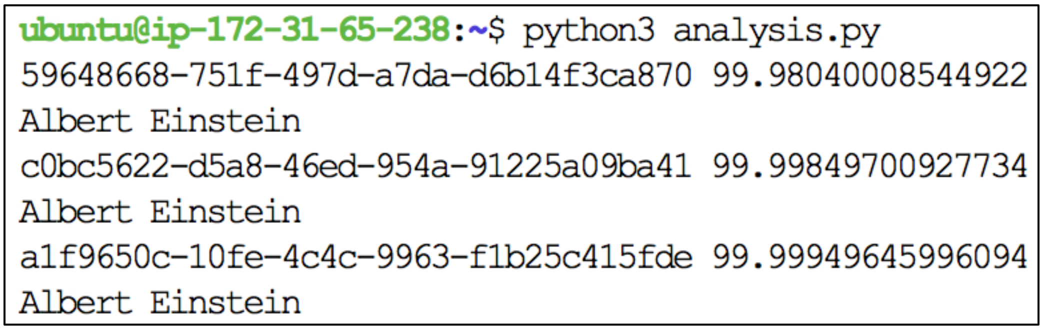
```
ubuntu@ip-172-31-65-238:~$ python3 analysis.py
no match found in person lookup
```

Test Picture 3



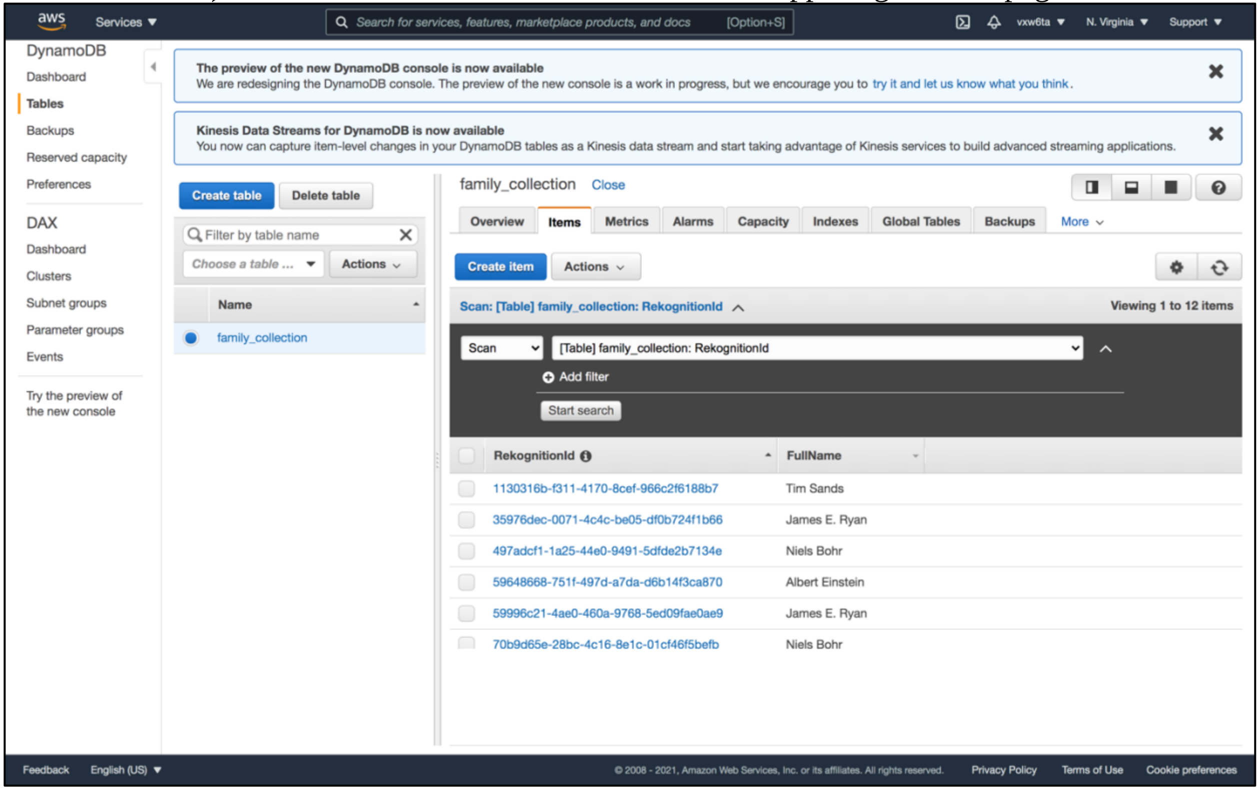
einstein-laurencelivermorenl.jpg

Terminal 3



```
ubuntu@ip-172-31-65-238:~$ python3 analysis.py
59648668-751f-497d-a7da-d6b14f3ca870 99.98040008544922
Albert Einstein
c0bc5622-d5a8-46ed-954a-91225a09ba41 99.99849700927734
Albert Einstein
a1f9650c-10fe-4c4c-9963-f1b25c415fde 99.99949645996094
Albert Einstein
```

c. A screenshot of https://console.aws.amazon.com/dynamodb/home?p=ddb&cp=bn&ad=c&region=us-east1#tables:selected=family_collection;tab=items . To receive full credit, your Name (or UVA ID) must be shown in the AWS console in the upper right of the page.



The preview of the new DynamoDB console is now available
We are redesigning the DynamoDB console. The preview of the new console is a work in progress, but we encourage you to try it and let us know what you think.

Kinesis Data Streams for DynamoDB is now available
You now can capture item-level changes in your DynamoDB tables as a Kinesis data stream and start taking advantage of Kinesis services to build advanced streaming applications.

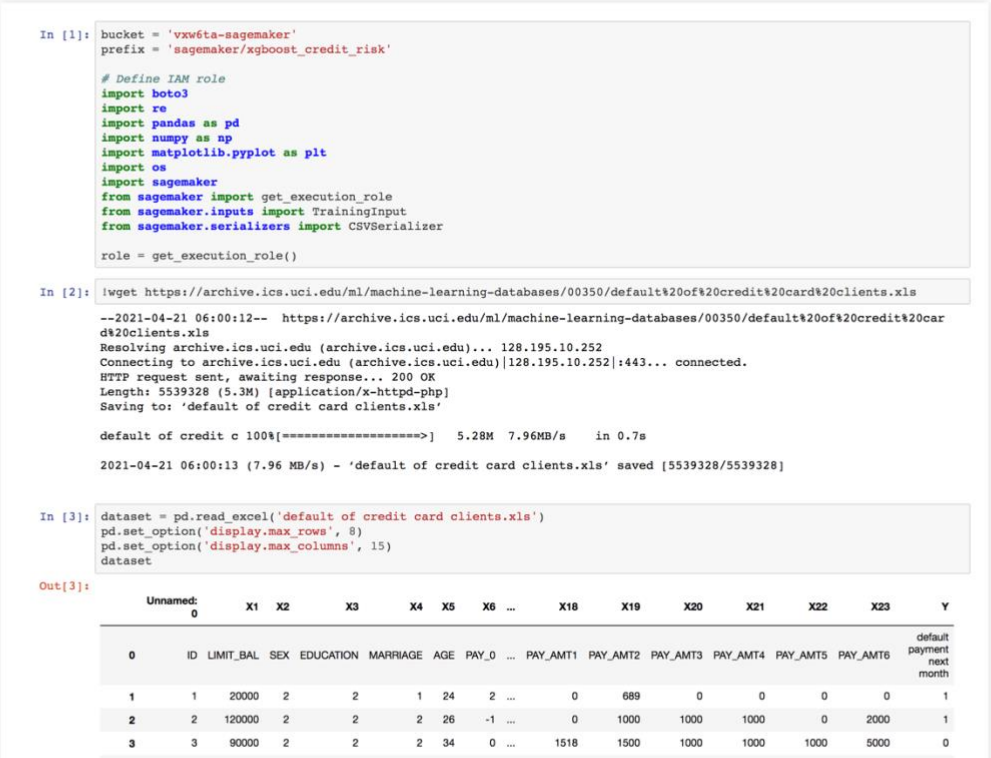
family_collection

Scan: [Table] family_collection: RekognitionId   Viewing 1 to 12 items

Scan ▾ [Table] family_collection: RekognitionId ▾

RekognitionId	FullName
1130316b-f311-4170-8cef-966c2f6188b7	Tim Sands
35976dec-0071-4c4c-be05-df0b724f1b66	James E. Ryan
497adcf1-1a25-44e0-9491-5dfde2b7134e	Niels Bohr
59648668-751f-497d-a7da-d6b14f3ca870	Albert Einstein
59996c21-4ae0-460a-9768-5ed09fae0ae9	James E. Ryan
70b9d65e-28bc-4c16-8e1c-01cf46f5befb	Niels Bohr

d. Open your Jupyter notebook as HTML, take a screen shot (showing the beginning of the notebook – e.g., step [1], step [2], and part of step[3]) and paste into your submission



In [1]:
```
bucket = 'vxw6ta-sagemaker'
prefix = 'sagemaker/xgboost_credit_risk'

# Define IAM role
import boto3
import re
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os
import sagemaker
from sagemaker import get_execution_role
from sagemaker.inputs import TrainingInput
from sagemaker.serializers import CSVSerializer

role = get_execution_role()
```

In [2]:
```
!wget https://archive.ics.uci.edu/ml/machine-learning-databases/00350/default%20of%20credit%20card%20clients.xls
```

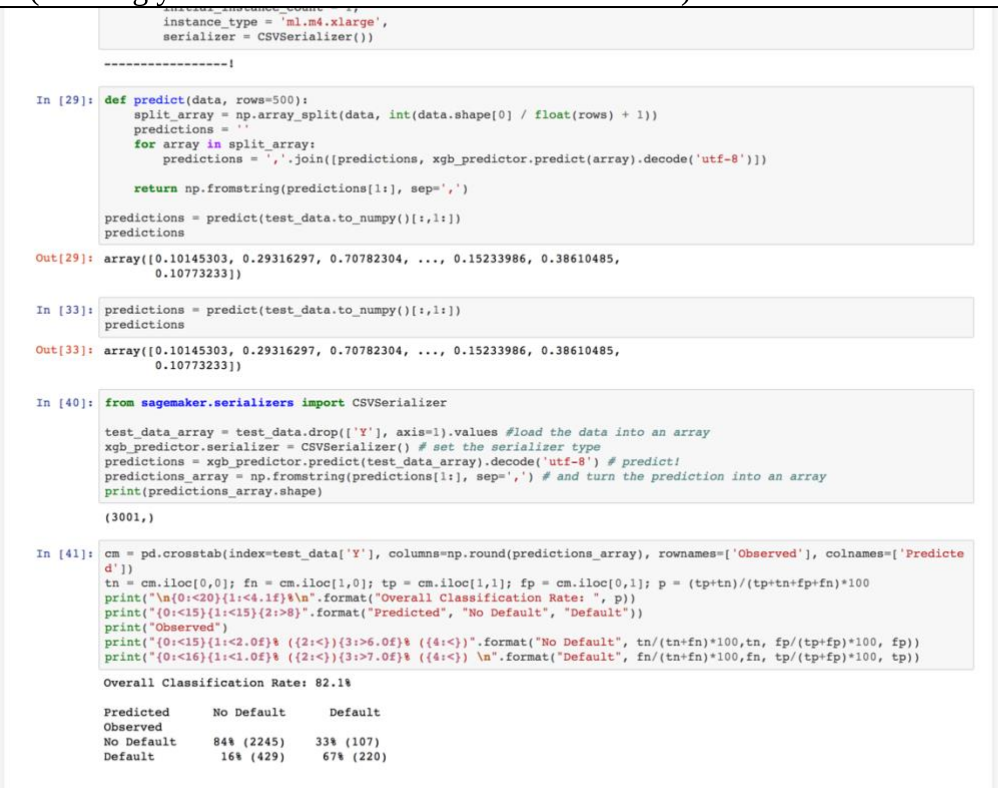
```
--2021-04-21 06:00:12--  https://archive.ics.uci.edu/ml/machine-learning-databases/00350/default%20of%20credit%20car
d%20clients.xls
Resolving archive.ics.uci.edu (archive.ics.uci.edu)... 128.195.10.252
Connecting to archive.ics.uci.edu (archive.ics.uci.edu)|128.195.10.252|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5539328 (5.3M) [application/x-httpd-php]
Saving to: ‘default of credit card clients.xls’

default of credit c 100%[===================>]   5.28M  7.96MB/s    in 0.7s

2021-04-21 06:00:13 (7.96 MB/s) - ‘default of credit card clients.xls’ saved [5539328/5539328]
```

In [3]:
```
dataset = pd.read_excel('default of credit card clients.xls')
pd.set_option('display.max_rows', 8)
pd.set_option('display.max_columns', 15)
dataset
```

Out[3]:

	Unnamed: 0	X1	X2	X3	X4	X5	X6	...	X18	X19	X20	X21	X22	X23	Y
0	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	...	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4	PAY_AMT5	PAY_AMT6	default payment next month
1	1	20000	2	2	1	24	2	...	0	689	0	0	0	0	1
2	2	120000	2	2	2	26	-1	...	0	1000	1000	1000	0	2000	1
3	3	90000	2	2	2	34	0	...	1518	1500	1000	1000	1000	5000	0

e. Open your Jupyter notebook as HTML, scroll down to the very end and take a screen shot (showing your calculation for the confusion matrix).



```
        instance_type = 'ml.m4.xlarge',
        serializer = CSVSerializer())
```

```
-----------------!
```

In [29]:
```
def predict(data, rows=500):
    split_array = np.array_split(data, int(data.shape[0] / float(rows) + 1))
    predictions = ''
    for array in split_array:
        predictions = ','.join([predictions, xgb_predictor.predict(array).decode('utf-8')])

    return np.fromstring(predictions[1:], sep=',')

predictions = predict(test_data.to_numpy()[:,1:])
predictions
```

Out[29]:
```
array([0.10145303, 0.29316297, 0.70782304, ..., 0.15233986, 0.38610485,
       0.10773233])
```

In [33]:
```
predictions = predict(test_data.to_numpy()[:,1:])
predictions
```

Out[33]:
```
array([0.10145303, 0.29316297, 0.70782304, ..., 0.15233986, 0.38610485,
       0.10773233])
```

In [40]:
```
from sagemaker.serializers import CSVSerializer

test_data_array = test_data.drop(['Y'], 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)
```

```
(3001,)
```

In [41]:
```
cm = pd.crosstab(index=test_data['Y'], columns=np.round(predictions_array), rownames=['Observed'], colnames=['Predicte
d'])
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 Default", "Default"))
print("Observed")
print("{0:<15}{1:<2.0f}% ({2:<}){3:>6.0f}% ({4:<})".format("No Default", tn/(tn+fn)*100,tn, fp/(tp+fp)*100, fp))
print("{0:<16}{1:<1.0f}% ({2:<}){3:>7.0f}% ({4:<}) \n".format("Default", fn/(tn+fn)*100,fn, tp/(tp+fp)*100, tp))
```

```
Overall Classification Rate: 82.1%

Predicted      No Default      Default
Observed
No Default     84% (2245)    33% (107)
Default         16% (429)     67% (220)
```

f. Your answers to the two questions from Step 9 above.
   a. What percent of the time did you predict a person would default on their credit card payment and they actually did default?
      67%
   b. What percent of the time did you predict a person would NOT default on their credit card and they actually did NOT default?
      84%