AI System Design

Mini 2 Lab 4
Andrew ID: oadedeji

April 29th, 2022

Overview

In this lab, I was able to load my scores at once to DynamoDB using a driver function. The driver function was updated to follow my API design. Examples of things changed are the parameters into the functions, the HTTP status code and more. I also scanned my retraining table and appended to a copy of the credit_train file. A new training file is formed, and this file is used in the do_model_update to perform the retraining. The lab1 code is edited such that the first 20 percent of data are used for validation, while the 80 percent are used for training (which includes the retraining data).

Questions

1. Listing of your retraining python application showing code to update training data and REST API calls to redo preprocessing and training

Observations and updates

- The starter code provided was updated with the access_key, secret_key, and the table.
- The line of code that appends the label was uncommented.
- Shutil.copyfile() is used to make a copy of the credit_score_clean file in that same directory.
- Under the do_model_update, similar code to what was used in model_drive is used for preprocessing and training.
- In the lab1 code, the splitting of train and validation data is changed. Since the retrained data is appended to a copy of the credit_train data, the first 20 percent of the data is used for validation, while the last 80 percent of the data is used for training.

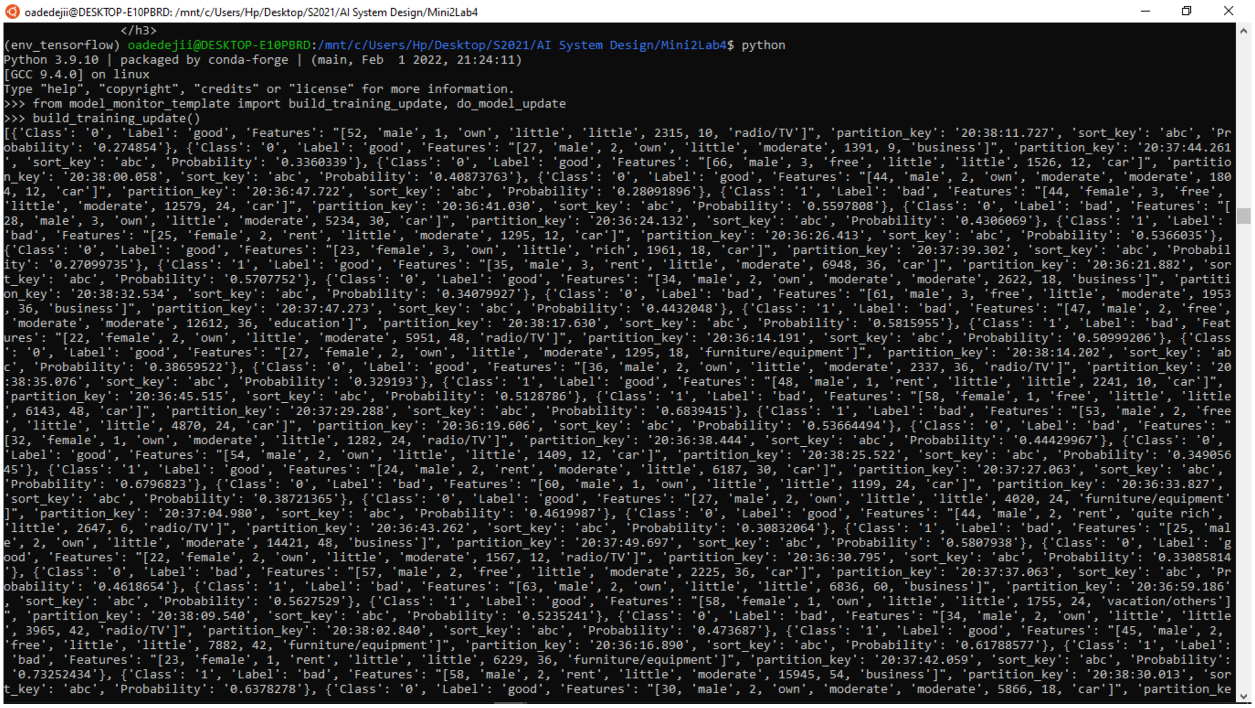
Retraining.py

```
import boto3
from botocore.config import Config
from boto3.dynamodb.conditions import Key, Attr
import time
import csv
from datetime import datetime
import requests
import sys
import ast
import shutil



my_config = Config(
    region_name = 'us-west-2'
)
```

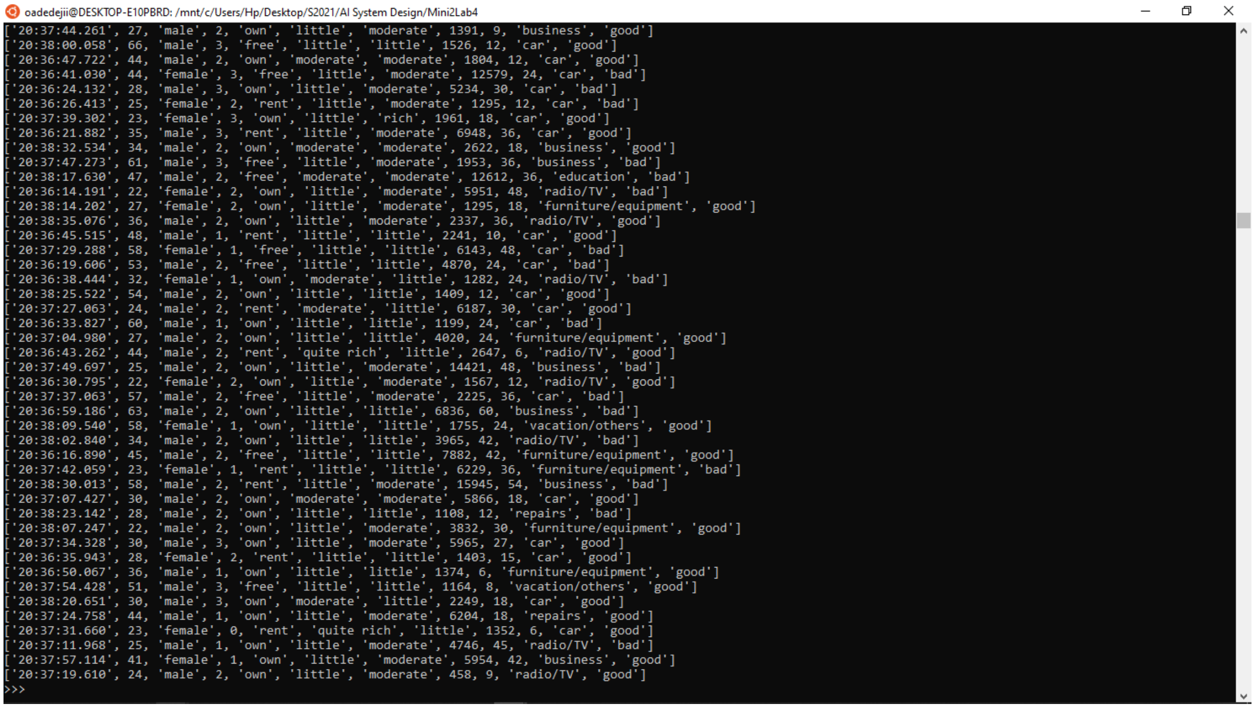
```
# Get the service resource.

session = boto3.Session(
    aws_access_key_id='AKIAVFPETSJYIUA66ASG',
    aws_secret_access_key='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'
)

dynamodb = session.resource('dynamodb', config=my_config)
update_table = dynamodb.Table('Lab3retraining')


def build_training_update():
    list_of_lists = []
    response = update_table.scan()
    items = response['Items']
    print(items)
    for item in items:
        # build the training feature set
        features_str = item['Features']
        features = ast.literal_eval(features_str)
        features.append(item['Label'])
        features.insert(0, item['partition_key'])
        print(features)
        list_of_lists.append( features )

    # copy original training data to new training_file_name.csv
    # check https://docs.python.org/3/library/shutil.html for info on how to do
the file system copy!

    shutil.copyfile("credit_train.csv","new_training_file.csv")

    with open("new_training_file.csv", "a") as f:
        wr = csv.writer(f)
        wr.writerows( list_of_lists )

    return

# use the example REST invocations in the model driver python script to then
reprocess your updated training data.
# be sure to do the "context" step as well as the retraining step
# then run a set of scoring tests to check the service is still operational

def do_model_update():
    # use the pattern from model_drive.py to pre-process and retrain you model,
calling the credit service using the REST API


    train_data = { 'training_data': 'new_training_file.csv' }
    r = requests.put("http://localhost:4000/credit/context", params=train_data)
    print(r.text)
    if ( r.status_code != 201 ):
        print("Exiting")
        sys.exit()

    r = requests.post("http://localhost:4000/credit/model")
    print(r.text)


    train_type = {"type":"whole"}
    r = requests.put("http://localhost:4000/credit/model", params= train_type)
    print(r.text)

    return
```

Results

Build_training_update()


oadedejii@DESKTOP-E10PBRD: /mnt/c/Users/Hp/Desktop/S2021/AI System Design/Mini2Lab4

```
                </h3>
(env_tensorflow) oadedejii@DESKTOP-E10PBRD:/mnt/c/Users/Hp/Desktop/S2021/AI System Design/Mini2Lab4$ python
Python 3.9.10 | packaged by conda-forge | (main, Feb  1 2022, 21:24:11)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from model_monitor_template import build_training_update, do_model_update
>>> build_training_update()
[{'Class': '0', 'Label': 'good', 'Features': "[52, 'male', 1, 'own', 'little', 'little', 2315, 10, 'radio/TV']", 'partition_key': '20:38:11.727', 'sort_key': 'abc', 'Pr
obability': '0.274854'}, {'Class': '0', 'Label': 'good', 'Features': "[27, 'male', 2, 'own', 'little', 'moderate', 1391, 9, 'business']", 'partition_key': '20:37:44.261
', 'sort_key': 'abc', 'Probability': '0.3360339'}, {'Class': '0', 'Label': 'good', 'Features': "[66, 'male', 3, 'free', 'little', 'little', 1526, 12, 'car']", 'partitio
n_key': '20:38:00.058', 'sort_key': 'abc', 'Probability': '0.40873763'}, {'Class': '0', 'Label': 'good', 'Features': "[44, 'male', 2, 'own', 'moderate', 'moderate', 180
4, 12, 'car']", 'partition_key': '20:36:47.722', 'sort_key': 'abc', 'Probability': '0.28091896'}, {'Class': '1', 'Label': 'bad', 'Features': "[44, 'female', 3, 'free',
'little', 'moderate', 12579, 24, 'car']", 'partition_key': '20:36:41.030', 'sort_key': 'abc', 'Probability': '0.5597808'}, {'Class': '0', 'Label': 'bad', 'Features': "[
28, 'male', 3, 'own', 'little', 'moderate', 5234, 30, 'car']", 'partition_key': '20:36:24.132', 'sort_key': 'abc', 'Probability': '0.4306069'}, {'Class': '1', 'Label':
'bad', 'Features': "[25, 'female', 2, 'rent', 'little', 'moderate', 1295, 12, 'car']", 'partition_key': '20:36:26.413', 'sort_key': 'abc', 'Probability': '0.5366035'},
{'Class': '0', 'Label': 'good', 'Features': "[23, 'female', 3, 'own', 'little', 'rich', 1961, 18, 'car']", 'partition_key': '20:37:39.302', 'sort_key': 'abc', 'Probabil
ity': '0.27099735'}, {'Class': '1', 'Label': 'good', 'Features': "[35, 'male', 3, 'rent', 'little', 'moderate', 6948, 36, 'car']", 'partition_key': '20:36:21.882', 'sor
t_key': 'abc', 'Probability': '0.5707752'}, {'Class': '0', 'Label': 'good', 'Features': "[34, 'male', 2, 'own', 'moderate', 'moderate', 2622, 18, 'business']", 'partiti
on_key': '20:38:32.534', 'sort_key': 'abc', 'Probability': '0.34079927'}, {'Class': '0', 'Label': 'bad', 'Features': "[61, 'male', 3, 'free', 'little', 'moderate', 1953
, 36, 'business']", 'partition_key': '20:37:47.273', 'sort_key': 'abc', 'Probability': '0.4432048'}, {'Class': '1', 'Label': 'bad', 'Features': "[47, 'male', 2, 'free',
 'moderate', 'moderate', 12612, 36, 'education']", 'partition_key': '20:38:17.630', 'sort_key': 'abc', 'Probability': '0.5815955'}, {'Class': '1', 'Label': 'bad', 'Feat
ures': "[22, 'female', 2, 'own', 'little', 'moderate', 5951, 48, 'radio/TV']", 'partition_key': '20:36:14.191', 'sort_key': 'abc', 'Probability': '0.50999206'}, {'Class
': '0', 'Label': 'good', 'Features': "[27, 'female', 2, 'own', 'little', 'moderate', 1295, 18, 'furniture/equipment']", 'partition_key': '20:38:14.202', 'sort_key': 'ab
c', 'Probability': '0.38659522'}, {'Class': '0', 'Label': 'good', 'Features': "[36, 'male', 2, 'own', 'little', 'moderate', 2337, 36, 'radio/TV']", 'partition_key': '20
:38:35.076', 'sort_key': 'abc', 'Probability': '0.329193'}, {'Class': '1', 'Label': 'good', 'Features': "[48, 'male', 1, 'rent', 'little', 'little', 2241, 10, 'car']",
'partition_key': '20:36:45.515', 'sort_key': 'abc', 'Probability': '0.5128786'}, {'Class': '1', 'Label': 'bad', 'Features': "[58, 'female', 1, 'free', 'little', 'little
', 6143, 48, 'car']", 'partition_key': '20:37:29.288', 'sort_key': 'abc', 'Probability': '0.6839415'}, {'Class': '1', 'Label': 'bad', 'Features': "[53, 'male', 2, 'free
', 'little', 'little', 4870, 24, 'car']", 'partition_key': '20:36:19.606', 'sort_key': 'abc', 'Probability': '0.53664494'}, {'Class': '0', 'Label': 'bad', 'Features': "
[32, 'female', 1, 'own', 'moderate', 'little', 1282, 24, 'radio/TV']", 'partition_key': '20:36:38.444', 'sort_key': 'abc', 'Probability': '0.44429967'}, {'Class': '0',
'Label': 'good', 'Features': "[54, 'male', 2, 'own', 'little', 'little', 1409, 12, 'car']", 'partition_key': '20:38:25.522', 'sort_key': 'abc', 'Probability': '0.349056
45'}, {'Class': '1', 'Label': 'good', 'Features': "[24, 'male', 2, 'rent', 'moderate', 'little', 6187, 30, 'car']", 'partition_key': '20:37:27.063', 'sort_key': 'abc',
'Probability': '0.6796823'}, {'Class': '0', 'Label': 'bad', 'Features': "[60, 'male', 1, 'own', 'little', 'little', 1199, 24, 'car']", 'partition_key': '20:36:33.827',
'sort_key': 'abc', 'Probability': '0.38721365'}, {'Class': '0', 'Label': 'good', 'Features': "[27, 'male', 2, 'own', 'little', 'little', 4020, 24, 'furniture/equipment'
]", 'partition_key': '20:37:04.980', 'sort_key': 'abc', 'Probability': '0.4619987'}, {'Class': '0', 'Label': 'good', 'Features': "[44, 'male', 2, 'rent', 'quite rich',
'little', 2647, 6, 'radio/TV']", 'partition_key': '20:36:43.262', 'sort_key': 'abc', 'Probability': '0.30832064'}, {'Class': '1', 'Label': 'bad', 'Features': "[25, 'mal
e', 2, 'own', 'little', 'moderate', 14421, 48, 'business']", 'partition_key': '20:37:49.697', 'sort_key': 'abc', 'Probability': '0.5807938'}, {'Class': '0', 'Label': 'g
ood', 'Features': "[22, 'female', 2, 'own', 'little', 'moderate', 1567, 12, 'radio/TV']", 'partition_key': '20:36:30.795', 'sort_key': 'abc', 'Probability': '0.33085814
'}, {'Class': '0', 'Label': 'bad', 'Features': "[57, 'male', 2, 'free', 'little', 'moderate', 2225, 36, 'car']", 'partition_key': '20:37:37.063', 'sort_key': 'abc', 'Pr
obability': '0.4618654'}, {'Class': '1', 'Label': 'bad', 'Features': "[63, 'male', 2, 'own', 'little', 'little', 6836, 60, 'business']", 'partition_key': '20:36:59.186'
, 'sort_key': 'abc', 'Probability': '0.5627529'}, {'Class': '1', 'Label': 'good', 'Features': "[58, 'female', 1, 'own', 'little', 'little', 1755, 24, 'vacation/others']
", 'partition_key': '20:38:09.540', 'sort_key': 'abc', 'Probability': '0.5235241'}, {'Class': '0', 'Label': 'bad', 'Features': "[34, 'male', 2, 'own', 'little', 'little
', 3965, 42, 'radio/TV']", 'partition_key': '20:38:02.840', 'sort_key': 'abc', 'Probability': '0.473687'}, {'Class': '1', 'Label': 'good', 'Features': "[45, 'male', 2,
'free', 'little', 'little', 7882, 42, 'furniture/equipment']", 'partition_key': '20:36:16.890', 'sort_key': 'abc', 'Probability': '0.61788577'}, {'Class': '1', 'Label':
 'bad', 'Features': "[23, 'female', 1, 'rent', 'little', 'little', 6229, 36, 'furniture/equipment']", 'partition_key': '20:37:42.059', 'sort_key': 'abc', 'Probability':
 '0.73252434'}, {'Class': '1', 'Label': 'bad', 'Features': "[58, 'male', 2, 'rent', 'little', 'moderate', 15945, 54, 'business']", 'partition_key': '20:38:30.013', 'sor
t_key': 'abc', 'Probability': '0.6378278'}, {'Class': '0', 'Label': 'good', 'Features': "[30, 'male', 2, 'own', 'moderate', 'moderate', 5866, 18, 'car']", 'partition_ke
```



oadedejii@DESKTOP-E10PBRD: /mnt/c/Users/Hp/Desktop/S2021/AI System Design/Mini2Lab4

```
['20:37:44.261', 27, 'male', 2, 'own', 'little', 'moderate', 1391, 9, 'business', 'good']
['20:38:00.058', 66, 'male', 3, 'free', 'little', 'little', 1526, 12, 'car', 'good']
['20:36:47.722', 44, 'male', 2, 'own', 'moderate', 'moderate', 1804, 12, 'car', 'good']
['20:36:41.030', 44, 'female', 3, 'free', 'little', 'moderate', 12579, 24, 'car', 'bad']
['20:36:24.132', 28, 'male', 3, 'own', 'little', 'moderate', 5234, 30, 'car', 'bad']
['20:36:26.413', 25, 'female', 2, 'rent', 'little', 'moderate', 1295, 12, 'car', 'bad']
['20:37:39.302', 23, 'female', 3, 'own', 'little', 'rich', 1961, 18, 'car', 'good']
['20:36:21.882', 35, 'male', 3, 'rent', 'little', 'moderate', 6948, 36, 'car', 'good']
['20:38:32.534', 34, 'male', 2, 'own', 'moderate', 'moderate', 2622, 18, 'business', 'good']
['20:37:47.273', 61, 'male', 3, 'free', 'little', 'moderate', 1953, 36, 'business', 'bad']
['20:38:17.630', 47, 'male', 2, 'free', 'moderate', 'moderate', 12612, 36, 'education', 'bad']
['20:36:14.191', 22, 'female', 2, 'own', 'little', 'moderate', 5951, 48, 'radio/TV', 'bad']
['20:38:14.202', 27, 'female', 2, 'own', 'little', 'moderate', 1295, 18, 'furniture/equipment', 'good']
['20:38:35.076', 36, 'male', 2, 'own', 'little', 'moderate', 2337, 36, 'radio/TV', 'good']
['20:36:45.515', 48, 'male', 1, 'rent', 'little', 'little', 2241, 10, 'car', 'good']
['20:37:29.288', 58, 'female', 1, 'free', 'little', 'little', 6143, 48, 'car', 'bad']
['20:36:19.606', 53, 'male', 2, 'free', 'little', 'little', 4870, 24, 'car', 'bad']
['20:36:38.444', 32, 'female', 1, 'own', 'moderate', 'little', 1282, 24, 'radio/TV', 'bad']
['20:38:25.522', 54, 'male', 2, 'own', 'little', 'little', 1409, 12, 'car', 'good']
['20:37:27.063', 24, 'male', 2, 'rent', 'moderate', 'little', 6187, 30, 'car', 'good']
['20:36:33.827', 60, 'male', 1, 'own', 'little', 'little', 1199, 24, 'car', 'bad']
['20:37:04.980', 27, 'male', 2, 'own', 'little', 'little', 4020, 24, 'furniture/equipment', 'good']
['20:36:43.262', 44, 'male', 2, 'rent', 'quite rich', 'little', 2647, 6, 'radio/TV', 'good']
['20:37:49.697', 25, 'male', 2, 'own', 'little', 'moderate', 14421, 48, 'business', 'bad']
['20:36:30.795', 22, 'female', 2, 'own', 'little', 'moderate', 1567, 12, 'radio/TV', 'good']
['20:37:37.063', 57, 'male', 2, 'free', 'little', 'moderate', 2225, 36, 'car', 'bad']
['20:36:59.186', 63, 'male', 2, 'own', 'little', 'little', 6836, 60, 'business', 'bad']
['20:38:09.540', 58, 'female', 1, 'own', 'little', 'little', 1755, 24, 'vacation/others', 'good']
['20:38:02.840', 34, 'male', 2, 'own', 'little', 'little', 3965, 42, 'radio/TV', 'bad']
['20:36:16.890', 45, 'male', 2, 'free', 'little', 'little', 7882, 42, 'furniture/equipment', 'good']
['20:37:42.059', 23, 'female', 1, 'rent', 'little', 'little', 6229, 36, 'furniture/equipment', 'bad']
['20:38:30.013', 58, 'male', 2, 'rent', 'little', 'moderate', 15945, 54, 'business', 'bad']
['20:37:07.427', 30, 'male', 2, 'own', 'moderate', 'moderate', 5866, 18, 'car', 'good']
['20:38:23.142', 28, 'male', 2, 'own', 'little', 'little', 1108, 12, 'repairs', 'bad']
['20:38:07.247', 22, 'male', 2, 'own', 'little', 'moderate', 3832, 30, 'furniture/equipment', 'good']
['20:37:34.328', 30, 'male', 3, 'own', 'little', 'moderate', 5965, 27, 'car', 'good']
['20:36:35.943', 28, 'female', 2, 'rent', 'little', 'little', 1403, 15, 'car', 'good']
['20:36:50.067', 36, 'male', 1, 'own', 'little', 'little', 1374, 6, 'furniture/equipment', 'good']
['20:37:54.428', 51, 'male', 3, 'free', 'little', 'little', 1164, 8, 'vacation/others', 'good']
['20:38:20.651', 30, 'male', 3, 'own', 'moderate', 'little', 2249, 18, 'car', 'good']
['20:37:24.758', 44, 'male', 1, 'own', 'little', 'moderate', 6204, 18, 'repairs', 'good']
['20:37:31.660', 23, 'female', 0, 'rent', 'quite rich', 'little', 1352, 6, 'car', 'good']
['20:37:11.968', 25, 'male', 1, 'own', 'little', 'moderate', 4746, 45, 'radio/TV', 'bad']
['20:37:57.114', 41, 'female', 1, 'own', 'little', 'moderate', 5954, 42, 'business', 'good']
['20:37:19.610', 24, 'male', 2, 'own', 'little', 'moderate', 458, 9, 'radio/TV', 'good']
>>>
```

As it can be seen, the new features are added to the end of the file.

Part of the lab1 code edited.

First 20 percent are used for validation, and the other 80 percent are used for training

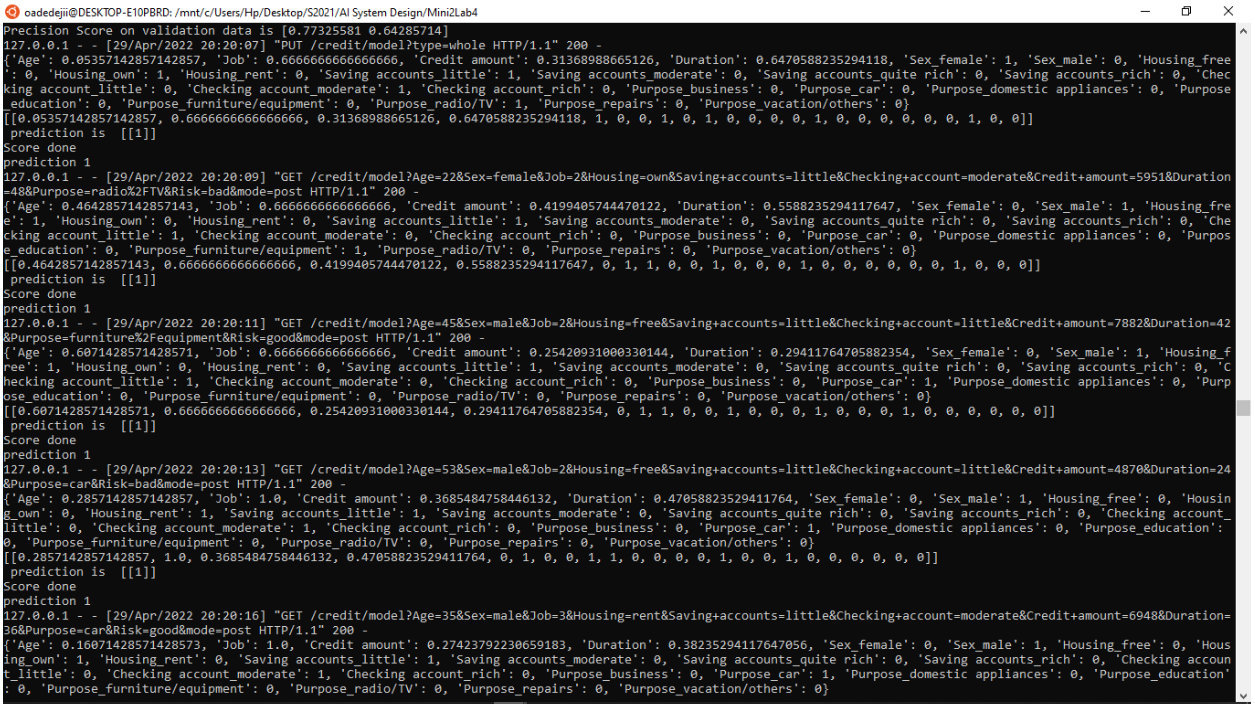
```
data_train = data.iloc[int(0.2*(len(data))):]
data_valid = data.iloc[:int(0.2*(len(data)))]
```

2. Copy and paste of Flask console showing invocation of initial setup, build, train, using automated application driver code.

   <u>Observations and updates</u>

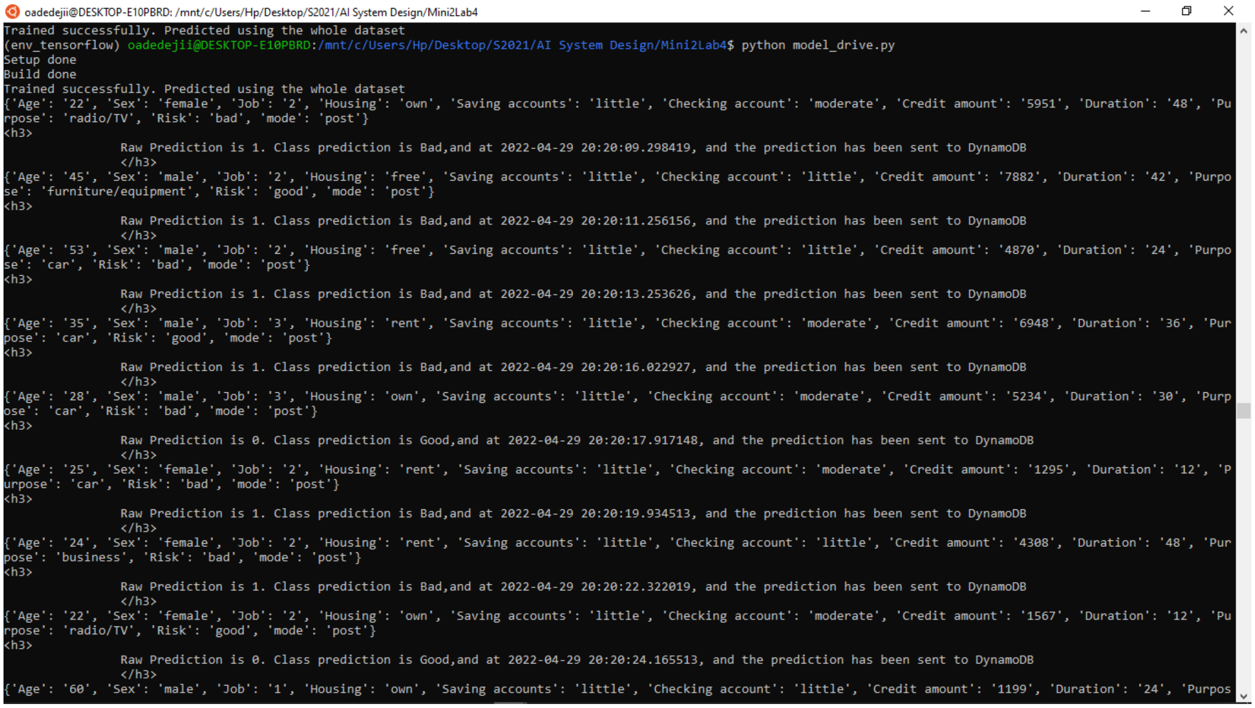
- I didn't need to change much here because I had similar model implementations
- Nevertheless, the changes made are related to the parameters passed in. For example, for training, another argument to state what type of prediction is done is passed. Also, when posting records, an extra parameter to denote whether to post or not is passed.

Flask invocation


oadedejii@DESKTOP-E10PBRD: /mnt/c/Users/Hp/Desktop/S2021/AI System Design/Mini2Lab4

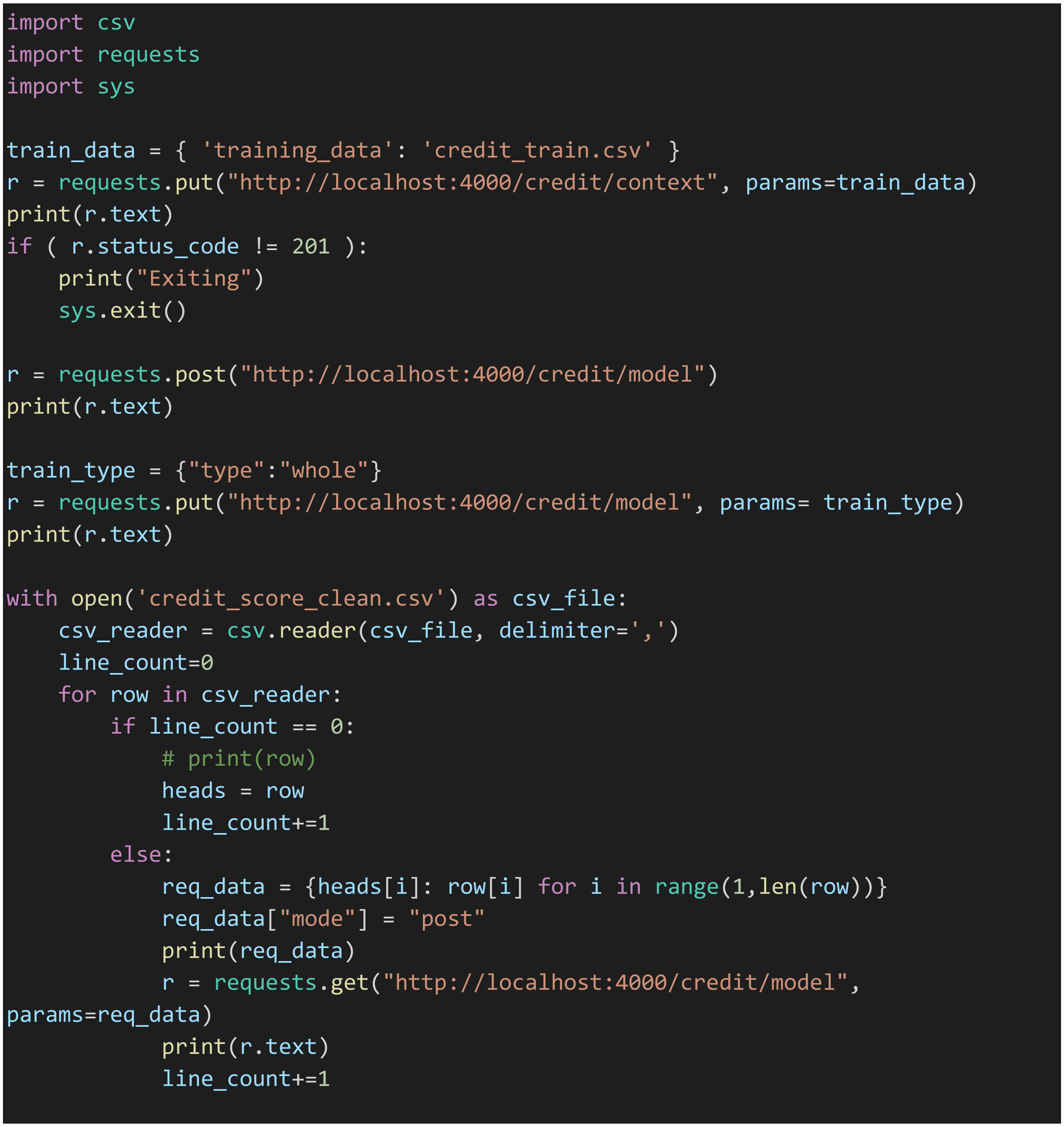
```
Precision Score on validation data is [0.77325581 0.64285714]
127.0.0.1 - - [29/Apr/2022 20:20:07] "PUT /credit/model?type=whole HTTP/1.1" 200 -
{'Age': 0.05357142857142857, 'Job': 0.6666666666666666, 'Credit amount': 0.31368988665126, 'Duration': 0.6470588235294118, 'Sex_female': 1, 'Sex_male': 0, 'Housing_free
': 0, 'Housing_own': 1, 'Housing_rent': 0, 'Saving accounts_little': 1, 'Saving accounts_moderate': 0, 'Saving accounts_quite rich': 0, 'Saving accounts_rich': 0, 'Chec
king account_little': 0, 'Checking account_moderate': 1, 'Checking account_rich': 0, 'Purpose_business': 0, 'Purpose_car': 0, 'Purpose_domestic appliances': 0, 'Purpose
_education': 0, 'Purpose_furniture/equipment': 0, 'Purpose_radio/TV': 1, 'Purpose_repairs': 0, 'Purpose_vacation/others': 0}
[[0.05357142857142857, 0.6666666666666666, 0.31368988665126, 0.6470588235294118, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0]]
 prediction is  [[1]]
Score done
prediction 1
127.0.0.1 - - [29/Apr/2022 20:20:09] "GET /credit/model?Age=22&Sex=female&Job=2&Housing=own&Saving+accounts=little&Checking+account=moderate&Credit+amount=5951&Duration
=48&Purpose=radio%2FTV&Risk=bad&mode=post HTTP/1.1" 200 -
{'Age': 0.4642857142857143, 'Job': 0.6666666666666666, 'Credit amount': 0.4199405744470122, 'Duration': 0.5588235294117647, 'Sex_female': 0, 'Sex_male': 1, 'Housing_fre
e': 1, 'Housing_own': 0, 'Housing_rent': 0, 'Saving accounts_little': 1, 'Saving accounts_moderate': 0, 'Saving accounts_quite rich': 0, 'Saving accounts_rich': 0, 'Che
cking account_little': 1, 'Checking account_moderate': 0, 'Checking account_rich': 0, 'Purpose_business': 0, 'Purpose_car': 0, 'Purpose_domestic appliances': 0, 'Purpos
e_education': 0, 'Purpose_furniture/equipment': 1, 'Purpose_radio/TV': 0, 'Purpose_repairs': 0, 'Purpose_vacation/others': 0}
[[0.4642857142857143, 0.6666666666666666, 0.4199405744470122, 0.5588235294117647, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0]]
 prediction is  [[1]]
Score done
prediction 1
127.0.0.1 - - [29/Apr/2022 20:20:11] "GET /credit/model?Age=45&Sex=male&Job=2&Housing=free&Saving+accounts=little&Checking+account=little&Credit+amount=7882&Duration=42
&Purpose=furniture%2Fequipment&Risk=good&mode=post HTTP/1.1" 200 -
{'Age': 0.6071428571428571, 'Job': 0.6666666666666666, 'Credit amount': 0.25420931000330144, 'Duration': 0.29411764705882354, 'Sex_female': 0, 'Sex_male': 1, 'Housing_f
ree': 1, 'Housing_own': 0, 'Housing_rent': 0, 'Saving accounts_little': 1, 'Saving accounts_moderate': 0, 'Saving accounts_quite rich': 0, 'Saving accounts_rich': 0, 'C
hecking account_little': 1, 'Checking account_moderate': 0, 'Checking account_rich': 0, 'Purpose_business': 0, 'Purpose_car': 1, 'Purpose_domestic appliances': 0, 'Purp
ose_education': 0, 'Purpose_furniture/equipment': 0, 'Purpose_radio/TV': 0, 'Purpose_repairs': 0, 'Purpose_vacation/others': 0}
[[0.6071428571428571, 0.6666666666666666, 0.25420931000330144, 0.29411764705882354, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]]
 prediction is  [[1]]
Score done
prediction 1
127.0.0.1 - - [29/Apr/2022 20:20:13] "GET /credit/model?Age=53&Sex=male&Job=2&Housing=free&Saving+accounts=little&Checking+account=little&Credit+amount=4870&Duration=24
&Purpose=car&Risk=bad&mode=post HTTP/1.1" 200 -
{'Age': 0.2857142857142857, 'Job': 1.0, 'Credit amount': 0.3685484758446132, 'Duration': 0.47058823529411764, 'Sex_female': 0, 'Sex_male': 1, 'Housing_free': 0, 'Housin
g_own': 0, 'Housing_rent': 1, 'Saving accounts_little': 1, 'Saving accounts_moderate': 0, 'Saving accounts_quite rich': 0, 'Saving accounts_rich': 0, 'Checking account_
little': 0, 'Checking account_moderate': 1, 'Checking account_rich': 0, 'Purpose_business': 0, 'Purpose_car': 1, 'Purpose_domestic appliances': 0, 'Purpose_education':
0, 'Purpose_furniture/equipment': 0, 'Purpose_radio/TV': 0, 'Purpose_repairs': 0, 'Purpose_vacation/others': 0}
[[0.2857142857142857, 1.0, 0.3685484758446132, 0.47058823529411764, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0]]
 prediction is  [[1]]
Score done
prediction 1
127.0.0.1 - - [29/Apr/2022 20:20:16] "GET /credit/model?Age=35&Sex=male&Job=3&Housing=rent&Saving+accounts=little&Checking+account=moderate&Credit+amount=6948&Duration=
36&Purpose=car&Risk=good&mode=post HTTP/1.1" 200 -
{'Age': 0.16071428571428573, 'Job': 1.0, 'Credit amount': 0.27423792230659183, 'Duration': 0.38235294117647056, 'Sex_female': 0, 'Sex_male': 1, 'Housing_free': 0, 'Hous
ing_own': 1, 'Housing_rent': 0, 'Saving accounts_little': 1, 'Saving accounts_moderate': 0, 'Saving accounts_quite rich': 0, 'Saving accounts_rich': 0, 'Checking accoun
t_little': 0, 'Checking account_moderate': 1, 'Checking account_rich': 0, 'Purpose_business': 0, 'Purpose_car': 1, 'Purpose_domestic appliances': 0, 'Purpose_education'
: 0, 'Purpose_furniture/equipment': 0, 'Purpose_radio/TV': 0, 'Purpose_repairs': 0, 'Purpose_vacation/others': 0}
```

Driver model output


oadedejii@DESKTOP-E10PBRD: /mnt/c/Users/Hp/Desktop/S2021/AI System Design/Mini2Lab4

```
Trained successfully. Predicted using the whole dataset
(env_tensorflow) oadedejii@DESKTOP-E10PBRD:/mnt/c/Users/Hp/Desktop/S2021/AI System Design/Mini2Lab4$ python model_drive.py
Setup done
Build done
Trained successfully. Predicted using the whole dataset
{'Age': '22', 'Sex': 'female', 'Job': '2', 'Housing': 'own', 'Saving accounts': 'little', 'Checking account': 'moderate', 'Credit amount': '5951', 'Duration': '48', 'Pu
rpose': 'radio/TV', 'Risk': 'bad', 'mode': 'post'}
<h3>
                Raw Prediction is 1. Class prediction is Bad,and at 2022-04-29 20:20:09.298419, and the prediction has been sent to DynamoDB
                </h3>
{'Age': '45', 'Sex': 'male', 'Job': '2', 'Housing': 'free', 'Saving accounts': 'little', 'Checking account': 'little', 'Credit amount': '7882', 'Duration': '42', 'Purpo
se': 'furniture/equipment', 'Risk': 'good', 'mode': 'post'}
<h3>
                Raw Prediction is 1. Class prediction is Bad,and at 2022-04-29 20:20:11.256156, and the prediction has been sent to DynamoDB
                </h3>
{'Age': '53', 'Sex': 'male', 'Job': '2', 'Housing': 'free', 'Saving accounts': 'little', 'Checking account': 'little', 'Credit amount': '4870', 'Duration': '24', 'Purpo
se': 'car', 'Risk': 'bad', 'mode': 'post'}
<h3>
                Raw Prediction is 1. Class prediction is Bad,and at 2022-04-29 20:20:13.253626, and the prediction has been sent to DynamoDB
                </h3>
{'Age': '35', 'Sex': 'male', 'Job': '3', 'Housing': 'rent', 'Saving accounts': 'little', 'Checking account': 'moderate', 'Credit amount': '6948', 'Duration': '36', 'Pur
pose': 'car', 'Risk': 'good', 'mode': 'post'}
<h3>
                Raw Prediction is 1. Class prediction is Bad,and at 2022-04-29 20:20:16.022927, and the prediction has been sent to DynamoDB
                </h3>
{'Age': '28', 'Sex': 'male', 'Job': '3', 'Housing': 'own', 'Saving accounts': 'little', 'Checking account': 'moderate', 'Credit amount': '5234', 'Duration': '30', 'Purp
ose': 'car', 'Risk': 'bad', 'mode': 'post'}
<h3>
                Raw Prediction is 0. Class prediction is Good,and at 2022-04-29 20:20:17.917148, and the prediction has been sent to DynamoDB
                </h3>
{'Age': '25', 'Sex': 'female', 'Job': '2', 'Housing': 'rent', 'Saving accounts': 'little', 'Checking account': 'moderate', 'Credit amount': '1295', 'Duration': '12', 'P
urpose': 'car', 'Risk': 'bad', 'mode': 'post'}
<h3>
                Raw Prediction is 1. Class prediction is Bad,and at 2022-04-29 20:20:19.934513, and the prediction has been sent to DynamoDB
                </h3>
{'Age': '24', 'Sex': 'female', 'Job': '2', 'Housing': 'rent', 'Saving accounts': 'little', 'Checking account': 'little', 'Credit amount': '4308', 'Duration': '48', 'Pur
pose': 'business', 'Risk': 'bad', 'mode': 'post'}
<h3>
                Raw Prediction is 1. Class prediction is Bad,and at 2022-04-29 20:20:22.322019, and the prediction has been sent to DynamoDB
                </h3>
{'Age': '22', 'Sex': 'female', 'Job': '2', 'Housing': 'own', 'Saving accounts': 'little', 'Checking account': 'moderate', 'Credit amount': '1567', 'Duration': '12', 'Pu
rpose': 'radio/TV', 'Risk': 'good', 'mode': 'post'}
<h3>
                Raw Prediction is 0. Class prediction is Good,and at 2022-04-29 20:20:24.165513, and the prediction has been sent to DynamoDB
                </h3>
{'Age': '60', 'Sex': 'male', 'Job': '1', 'Housing': 'own', 'Saving accounts': 'little', 'Checking account': 'little', 'Credit amount': '1199', 'Duration': '24', 'Purpos
```

Model_driver.py code

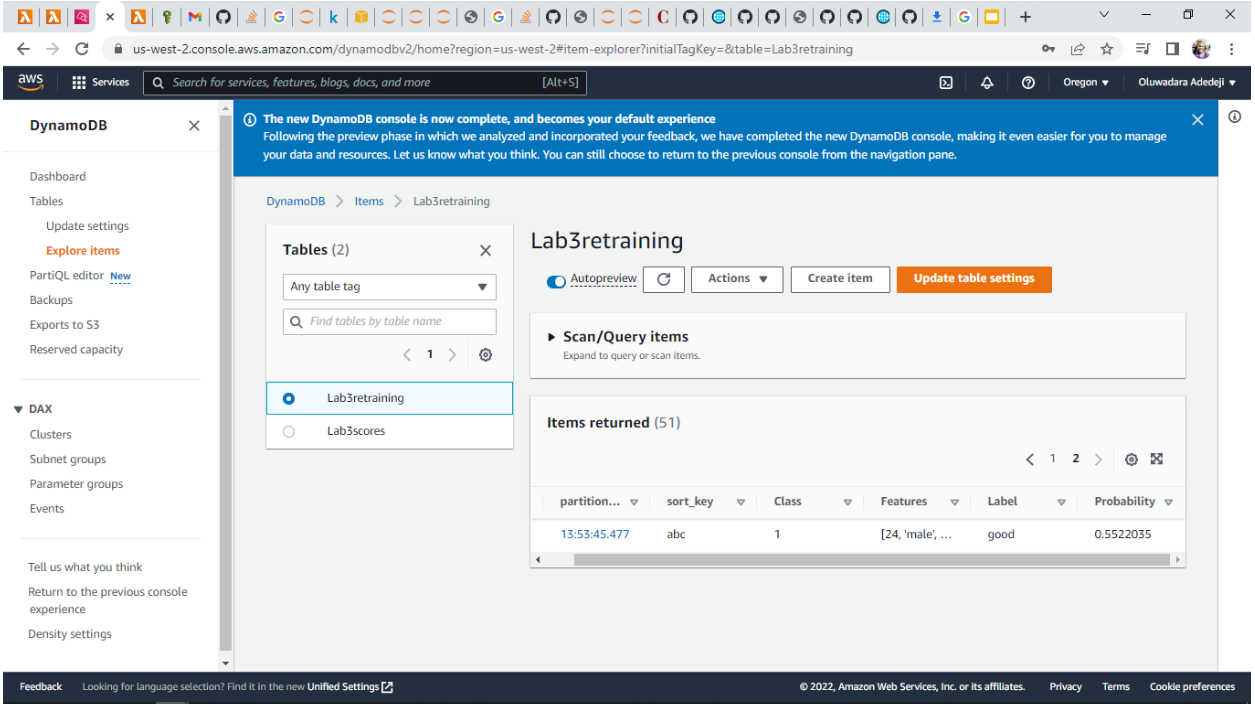


```
import csv
import requests
import sys

train_data = { 'training_data': 'credit_train.csv' }
r = requests.put("http://localhost:4000/credit/context", params=train_data)
print(r.text)
if ( r.status_code != 201 ):
    print("Exiting")
    sys.exit()

r = requests.post("http://localhost:4000/credit/model")
print(r.text)

train_type = {"type":"whole"}
r = requests.put("http://localhost:4000/credit/model", params= train_type)
print(r.text)

with open('credit_score_clean.csv') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    line_count=0
    for row in csv_reader:
        if line_count == 0:
            # print(row)
            heads = row
            line_count+=1
        else:
            req_data = {heads[i]: row[i] for i in range(1,len(row))}
            req_data["mode"] = "post"
            print(req_data)
            r = requests.get("http://localhost:4000/credit/model",
params=req_data)
            print(r.text)
            line_count+=1
```

3. Screenshot showing how many items appeared in retraining table in AWS after running bulk scoring operation.

<u>Observation and updates</u>

- 57 records are posted to the scoring table, and 51 of these entered the retraining table. The screenshots only show the second view of the tables, which show how many items are in each table.



us-west-2.console.aws.amazon.com/dynamodbv2/home?region=us-west-2#item-explorer?initialTagKey=&table=Lab3retraining

The new DynamoDB console is now complete, and becomes your default experience
Following the preview phase in which we analyzed and incorporated your feedback, we have completed the new DynamoDB console, making it even easier for you to manage your data and resources. Let us know what you think. You can still choose to return to the previous console from the navigation pane.

DynamoDB > Items > Lab3retraining

Tables (2)

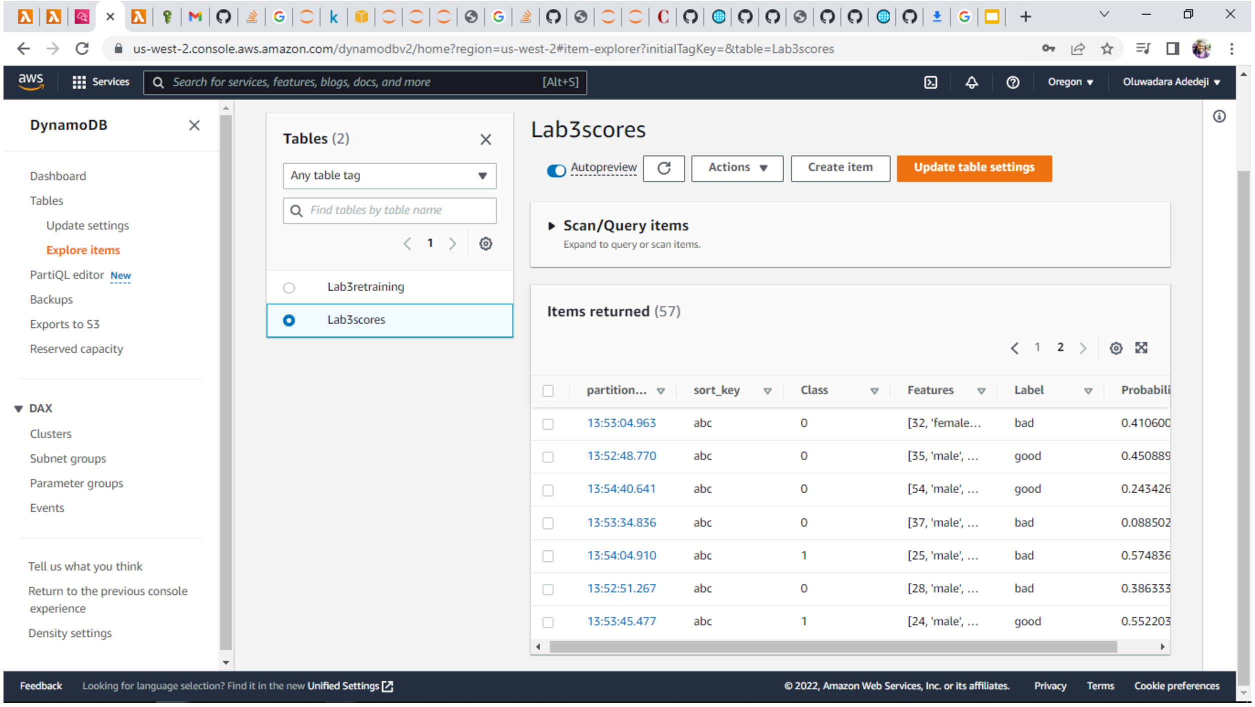
- ● Lab3retraining
- ○ Lab3scores

Lab3retraining

Scan/Query items
Expand to query or scan items.

Items returned (51)

partition...	sort_key	Class	Features	Label	Probability
13:53:45.477	abc	1	[24, 'male', ...	good	0.5522035



us-west-2.console.aws.amazon.com/dynamodbv2/home?region=us-west-2#item-explorer?initialTagKey=&table=Lab3scores

Tables (2)

- ○ Lab3retraining
- ● Lab3scores

Lab3scores

Scan/Query items
Expand to query or scan items.

Items returned (57)

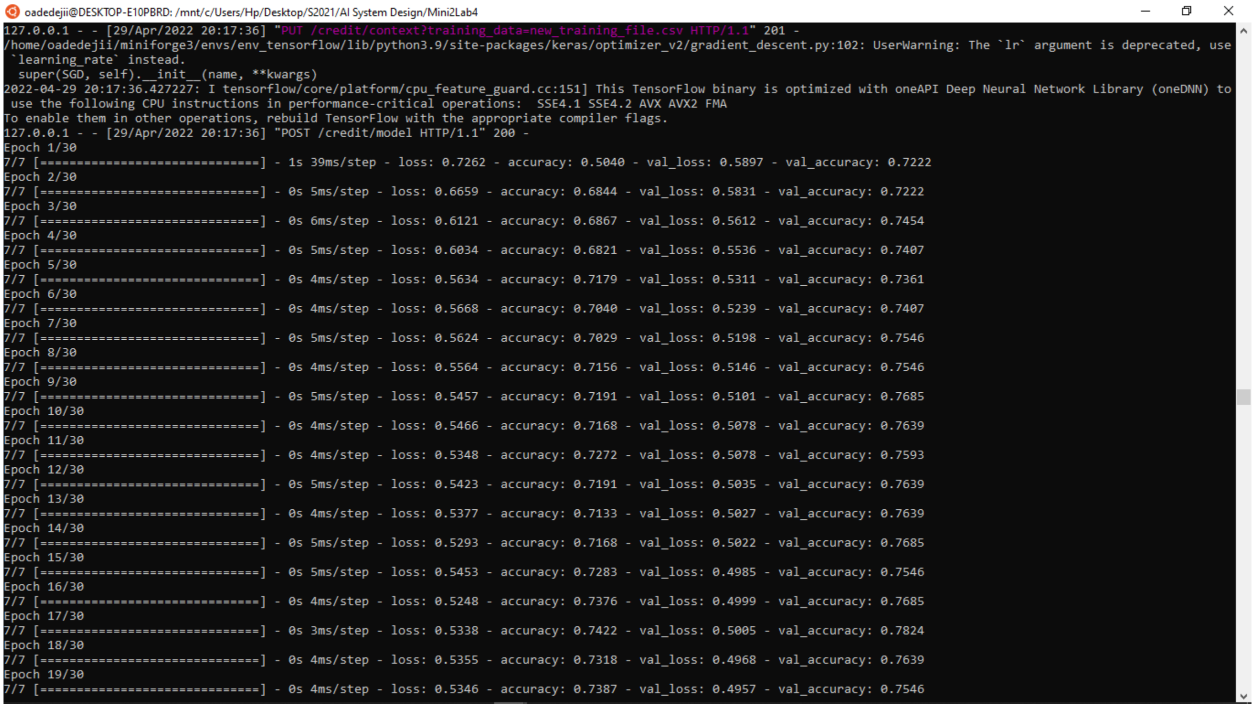
□	partition...	sort_key	Class	Features	Label	Probabili
□	13:53:04.963	abc	0	[32, 'female...	bad	0.410600
□	13:52:48.770	abc	0	[35, 'male', ...	good	0.450889
□	13:54:40.641	abc	0	[54, 'male', ...	good	0.243426
□	13:53:34.836	abc	0	[37, 'male', ...	bad	0.088502
□	13:54:04.910	abc	1	[25, 'male', ...	bad	0.574836
□	13:52:51.267	abc	0	[28, 'male', ...	bad	0.386333
□	13:53:45.477	abc	1	[24, 'male', ...	good	0.552203

4. Copy and paste of Flask console showing execution of retraining application including setup and training REST API calls again

Observation and updates

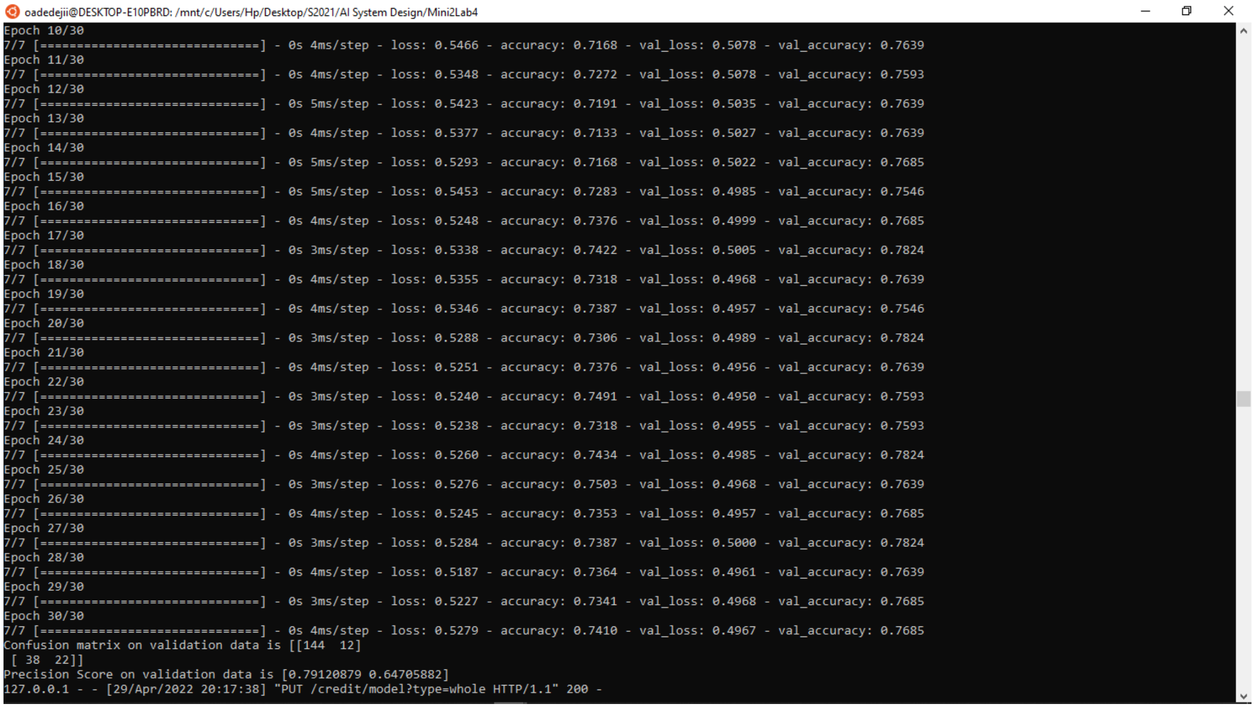
- The console shows that the setup was done (with the new training file), the build and also train are successfully done. The data is also checked as seen in the screenshot below.
- The updated do_model_update code is also shown.

Flask invocation



oadedejii@DESKTOP-E10PBRD: /mnt/c/Users/Hp/Desktop/S2021/AI System Design/Mini2Lab4

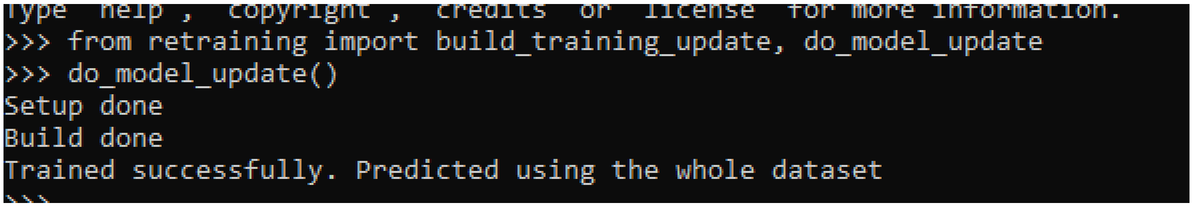
```
127.0.0.1 - - [29/Apr/2022 20:17:36] "PUT /credit/context?training_data=new_training_file.csv HTTP/1.1" 201 -
/home/oadedejii/miniforge3/envs/env_tensorflow/lib/python3.9/site-packages/keras/optimizer_v2/gradient_descent.py:102: UserWarning: The `lr` argument is deprecated, use
 `learning_rate` instead.
  super(SGD, self).__init__(name, **kwargs)
2022-04-29 20:17:36.427227: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to
 use the following CPU instructions in performance-critical operations:  SSE4.1 SSE4.2 AVX AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
127.0.0.1 - - [29/Apr/2022 20:17:36] "POST /credit/model HTTP/1.1" 200 -
Epoch 1/30
7/7 [==============================] - 1s 39ms/step - loss: 0.7262 - accuracy: 0.5040 - val_loss: 0.5897 - val_accuracy: 0.7222
Epoch 2/30
7/7 [==============================] - 0s 5ms/step - loss: 0.6659 - accuracy: 0.6844 - val_loss: 0.5831 - val_accuracy: 0.7222
Epoch 3/30
7/7 [==============================] - 0s 6ms/step - loss: 0.6121 - accuracy: 0.6867 - val_loss: 0.5612 - val_accuracy: 0.7454
Epoch 4/30
7/7 [==============================] - 0s 5ms/step - loss: 0.6034 - accuracy: 0.6821 - val_loss: 0.5536 - val_accuracy: 0.7407
Epoch 5/30
7/7 [==============================] - 0s 4ms/step - loss: 0.5634 - accuracy: 0.7179 - val_loss: 0.5311 - val_accuracy: 0.7361
Epoch 6/30
7/7 [==============================] - 0s 4ms/step - loss: 0.5668 - accuracy: 0.7040 - val_loss: 0.5239 - val_accuracy: 0.7407
Epoch 7/30
7/7 [==============================] - 0s 5ms/step - loss: 0.5624 - accuracy: 0.7029 - val_loss: 0.5198 - val_accuracy: 0.7546
Epoch 8/30
7/7 [==============================] - 0s 4ms/step - loss: 0.5564 - accuracy: 0.7156 - val_loss: 0.5146 - val_accuracy: 0.7546
Epoch 9/30
7/7 [==============================] - 0s 5ms/step - loss: 0.5457 - accuracy: 0.7191 - val_loss: 0.5101 - val_accuracy: 0.7685
Epoch 10/30
7/7 [==============================] - 0s 4ms/step - loss: 0.5466 - accuracy: 0.7168 - val_loss: 0.5078 - val_accuracy: 0.7639
Epoch 11/30
7/7 [==============================] - 0s 4ms/step - loss: 0.5348 - accuracy: 0.7272 - val_loss: 0.5078 - val_accuracy: 0.7593
Epoch 12/30
7/7 [==============================] - 0s 5ms/step - loss: 0.5423 - accuracy: 0.7191 - val_loss: 0.5035 - val_accuracy: 0.7639
Epoch 13/30
7/7 [==============================] - 0s 4ms/step - loss: 0.5377 - accuracy: 0.7133 - val_loss: 0.5027 - val_accuracy: 0.7639
Epoch 14/30
7/7 [==============================] - 0s 5ms/step - loss: 0.5293 - accuracy: 0.7168 - val_loss: 0.5022 - val_accuracy: 0.7685
Epoch 15/30
7/7 [==============================] - 0s 5ms/step - loss: 0.5453 - accuracy: 0.7283 - val_loss: 0.4985 - val_accuracy: 0.7546
Epoch 16/30
7/7 [==============================] - 0s 4ms/step - loss: 0.5248 - accuracy: 0.7376 - val_loss: 0.4999 - val_accuracy: 0.7685
Epoch 17/30
7/7 [==============================] - 0s 3ms/step - loss: 0.5338 - accuracy: 0.7422 - val_loss: 0.5005 - val_accuracy: 0.7824
Epoch 18/30
7/7 [==============================] - 0s 4ms/step - loss: 0.5355 - accuracy: 0.7318 - val_loss: 0.4968 - val_accuracy: 0.7639
Epoch 19/30
7/7 [==============================] - 0s 4ms/step - loss: 0.5346 - accuracy: 0.7387 - val_loss: 0.4957 - val_accuracy: 0.7546
```



oadedejii@DESKTOP-E10PBRD: /mnt/c/Users/Hp/Desktop/S2021/AI System Design/Mini2Lab4

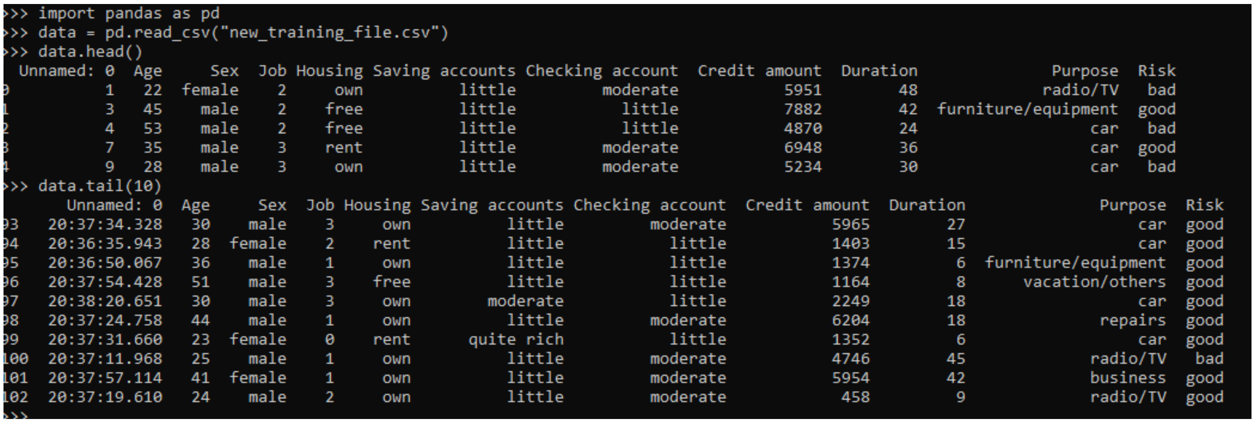
```
Epoch 10/30
7/7 [==============================] - 0s 4ms/step - loss: 0.5466 - accuracy: 0.7168 - val_loss: 0.5078 - val_accuracy: 0.7639
Epoch 11/30
7/7 [==============================] - 0s 4ms/step - loss: 0.5348 - accuracy: 0.7272 - val_loss: 0.5078 - val_accuracy: 0.7593
Epoch 12/30
7/7 [==============================] - 0s 5ms/step - loss: 0.5423 - accuracy: 0.7191 - val_loss: 0.5035 - val_accuracy: 0.7639
Epoch 13/30
7/7 [==============================] - 0s 4ms/step - loss: 0.5377 - accuracy: 0.7133 - val_loss: 0.5027 - val_accuracy: 0.7639
Epoch 14/30
7/7 [==============================] - 0s 5ms/step - loss: 0.5293 - accuracy: 0.7168 - val_loss: 0.5022 - val_accuracy: 0.7685
Epoch 15/30
7/7 [==============================] - 0s 5ms/step - loss: 0.5453 - accuracy: 0.7283 - val_loss: 0.4985 - val_accuracy: 0.7546
Epoch 16/30
7/7 [==============================] - 0s 4ms/step - loss: 0.5248 - accuracy: 0.7376 - val_loss: 0.4999 - val_accuracy: 0.7685
Epoch 17/30
7/7 [==============================] - 0s 3ms/step - loss: 0.5338 - accuracy: 0.7422 - val_loss: 0.5005 - val_accuracy: 0.7824
Epoch 18/30
7/7 [==============================] - 0s 4ms/step - loss: 0.5355 - accuracy: 0.7318 - val_loss: 0.4968 - val_accuracy: 0.7639
Epoch 19/30
7/7 [==============================] - 0s 4ms/step - loss: 0.5346 - accuracy: 0.7387 - val_loss: 0.4957 - val_accuracy: 0.7546
Epoch 20/30
7/7 [==============================] - 0s 3ms/step - loss: 0.5288 - accuracy: 0.7306 - val_loss: 0.4989 - val_accuracy: 0.7824
Epoch 21/30
7/7 [==============================] - 0s 4ms/step - loss: 0.5251 - accuracy: 0.7376 - val_loss: 0.4956 - val_accuracy: 0.7639
Epoch 22/30
7/7 [==============================] - 0s 3ms/step - loss: 0.5240 - accuracy: 0.7491 - val_loss: 0.4950 - val_accuracy: 0.7593
Epoch 23/30
7/7 [==============================] - 0s 3ms/step - loss: 0.5238 - accuracy: 0.7318 - val_loss: 0.4955 - val_accuracy: 0.7593
Epoch 24/30
7/7 [==============================] - 0s 4ms/step - loss: 0.5260 - accuracy: 0.7434 - val_loss: 0.4985 - val_accuracy: 0.7824
Epoch 25/30
7/7 [==============================] - 0s 3ms/step - loss: 0.5276 - accuracy: 0.7503 - val_loss: 0.4968 - val_accuracy: 0.7639
Epoch 26/30
7/7 [==============================] - 0s 4ms/step - loss: 0.5245 - accuracy: 0.7353 - val_loss: 0.4957 - val_accuracy: 0.7685
Epoch 27/30
7/7 [==============================] - 0s 3ms/step - loss: 0.5284 - accuracy: 0.7387 - val_loss: 0.5000 - val_accuracy: 0.7824
Epoch 28/30
7/7 [==============================] - 0s 4ms/step - loss: 0.5187 - accuracy: 0.7364 - val_loss: 0.4961 - val_accuracy: 0.7639
Epoch 29/30
7/7 [==============================] - 0s 3ms/step - loss: 0.5227 - accuracy: 0.7341 - val_loss: 0.4968 - val_accuracy: 0.7685
Epoch 30/30
7/7 [==============================] - 0s 4ms/step - loss: 0.5279 - accuracy: 0.7410 - val_loss: 0.4967 - val_accuracy: 0.7685
Confusion matrix on validation data is [[144  12]
 [ 38  22]]
Precision Score on validation data is [0.79120879 0.64705882]
127.0.0.1 - - [29/Apr/2022 20:17:38] "PUT /credit/model?type=whole HTTP/1.1" 200 -
```

Do_model_update



```
Type "help", "copyright", "credits" or "license" for more information.
>>> from retraining import build_training_update, do_model_update
>>> do_model_update()
Setup done
Build done
Trained successfully. Predicted using the whole dataset
>>>
```

New data



```
>>> import pandas as pd
>>> data = pd.read_csv("new_training_file.csv")
>>> data.head()
  Unnamed: 0  Age     Sex  Job Housing Saving accounts Checking account  Credit amount  Duration              Purpose  Risk
0          1   22  female    2     own          little         moderate           5951        48             radio/TV   bad
1          3   45    male    2    free          little           little           7882        42  furniture/equipment  good
2          4   53    male    2    free          little           little           4870        24                  car   bad
3          7   35    male    3    rent          little         moderate           6948        36                  car  good
4          9   28    male    3     own          little         moderate           5234        30                  car   bad
>>> data.tail(10)
       Unnamed: 0  Age     Sex  Job Housing Saving accounts Checking account  Credit amount  Duration              Purpose  Risk
93   20:37:34.328   30    male    3     own          little         moderate           5965        27                  car  good
94   20:36:35.943   28  female    2    rent          little           little           1403        15                  car  good
95   20:36:50.067   36    male    1     own          little           little           1374         6  furniture/equipment  good
96   20:37:54.428   51    male    3    free          little           little           1164         8       vacation/others  good
97   20:38:20.651   30    male    3     own        moderate           little           2249        18                  car  good
98   20:37:24.758   44    male    1     own          little         moderate           6204        18              repairs  good
99   20:37:31.660   23  female    0    rent      quite rich           little           1352         6                  car  good
100  20:37:11.968   25    male    1     own          little         moderate           4746        45             radio/TV   bad
101  20:37:57.114   41  female    1     own          little         moderate           5954        42             business  good
102  20:37:19.610   24    male    2     own          little         moderate            458         9             radio/TV  good
>>>
```

The records added are seen at the tail end of the data.

Do_model_update code only

```
def do_model_update():
    # use the pattern from model_drive.py to pre-process and retrain you model,
calling the credit service using the REST API


    train_data = { 'training_data': 'new_training_file.csv' }
    r = requests.put("http://localhost:4000/credit/context", params=train_data)
    print(r.text)
    if ( r.status_code != 201 ):
        print("Exiting")
        sys.exit()

    r = requests.post("http://localhost:4000/credit/model")
    print(r.text)


    train_type = {"type":"whole"}
    r = requests.put("http://localhost:4000/credit/model", params= train_type)
    print(r.text)
    return
```

Conclusion

At the end of this lab, I was able to write multiple records to the dynamodb scoring table using a model_drive function. Also, I was able to get records with low confidence to the retraining table and these are scanned, and updated to a copy of the credit_train file. This is then used for retraining. This and other labs in this course have improved greatly my familiarity with REST API, postman, and components of AWS.