Web Application for Evaluating Recommender Systems Machine Learning

CS 682 Capstone Project

USER GUIDE

By:

Andrew Hochstetler
University of Massachusetts Boston
Department of Computer Science
andrew.hochstetle001@umb.edu

Ashley Farrell
University of Massachusetts Boston
Department of Computer Science
ashley.farrell001@umb.edu

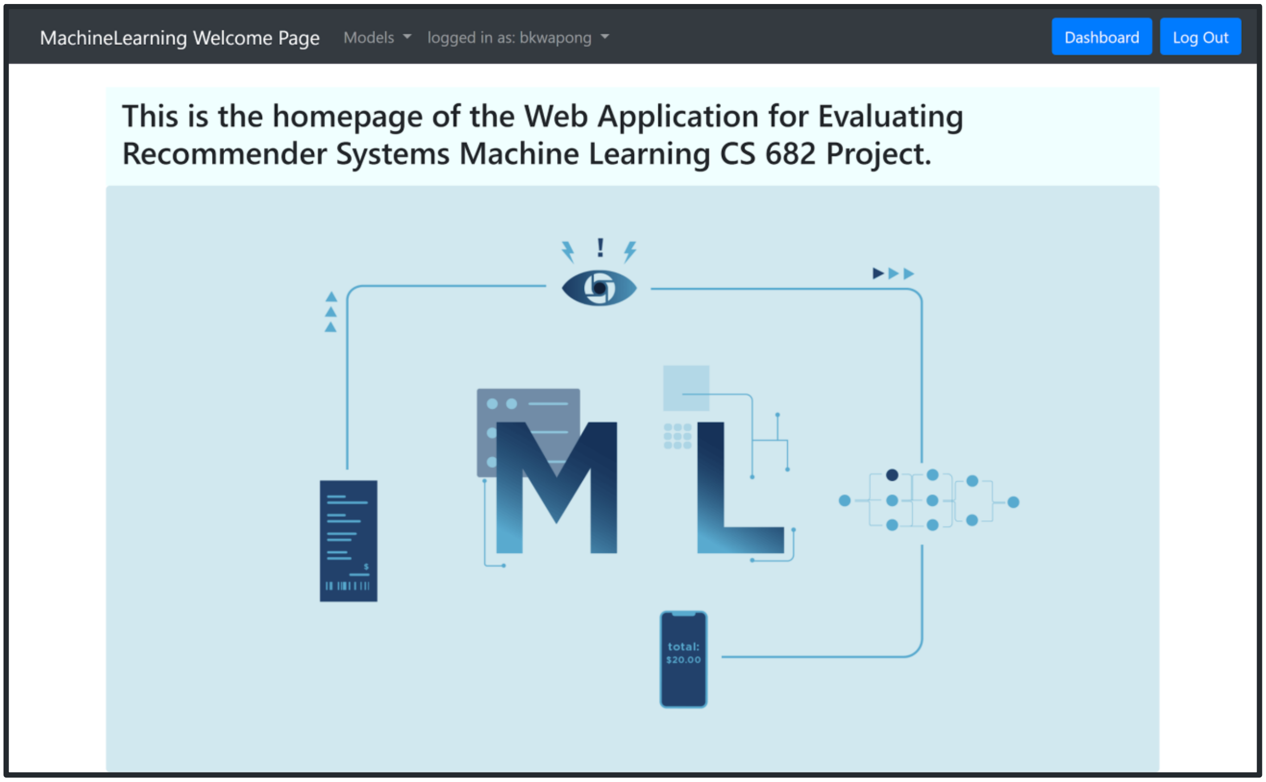
Benjamin Kwapong
University of Massachusetts Boston
Department of Computer Science
benjamin.kwapong001@umb.edu

Richard Anarfi
University of Massachusetts Boston
Department of Computer Science
richard.anarfi001@umb.edu

Admin:
Dr. Kenneth Fletcher
University of Massachusetts Boston
Department of Computer Science
kenneth.fletcher@umb.edu

Table of Contents

Welcome Page

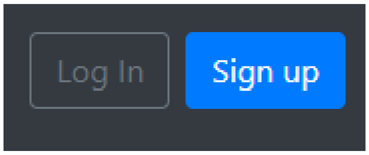


This is the homepage of the Web Application for Evaluating Recommender Systems Machine Learning CS 682 Project.

New account creation

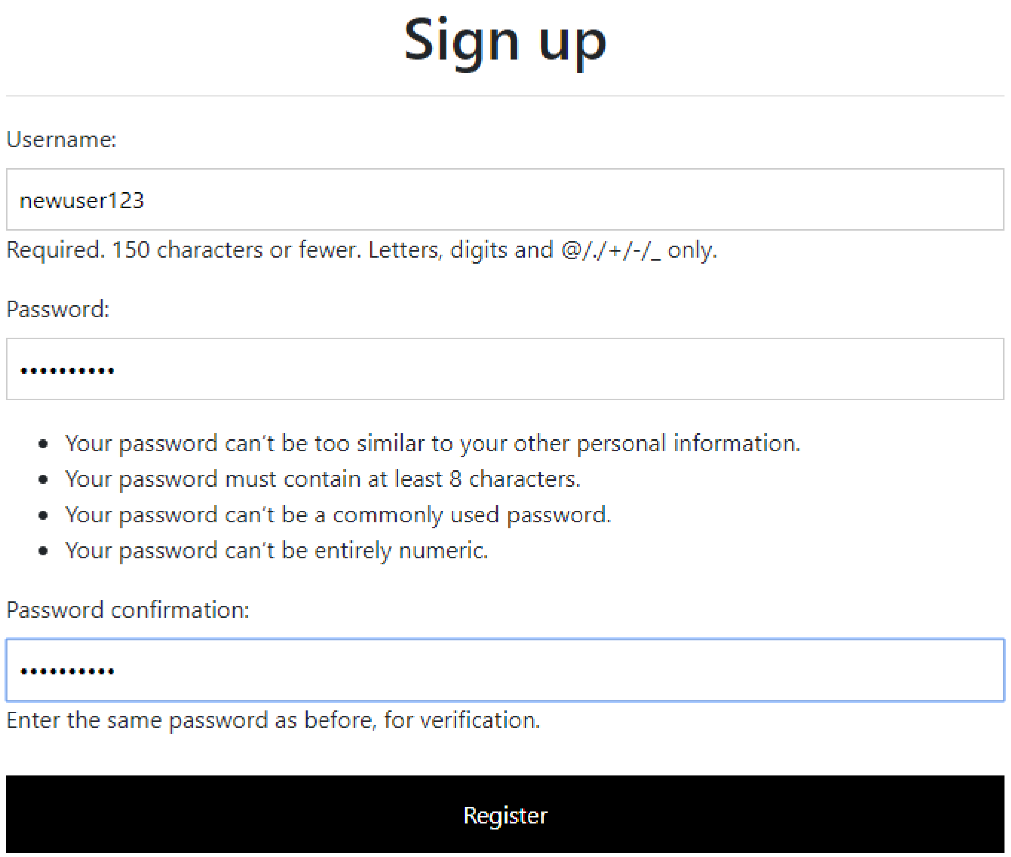
You can create a new account by following the steps outlined below. If you already have an account and password, skip to the “Login process” section for details on logging in.

1. Select “Sign Up” from either the widget on the far right of the title bar:



Or the “Sign Up” button in the welcome text. This will direct you to the registration form.

2. Fill out the form by selecting a user name, password, and confirming your password selection.



Sign up

Username:

newuser123

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Password:

••••••••••

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

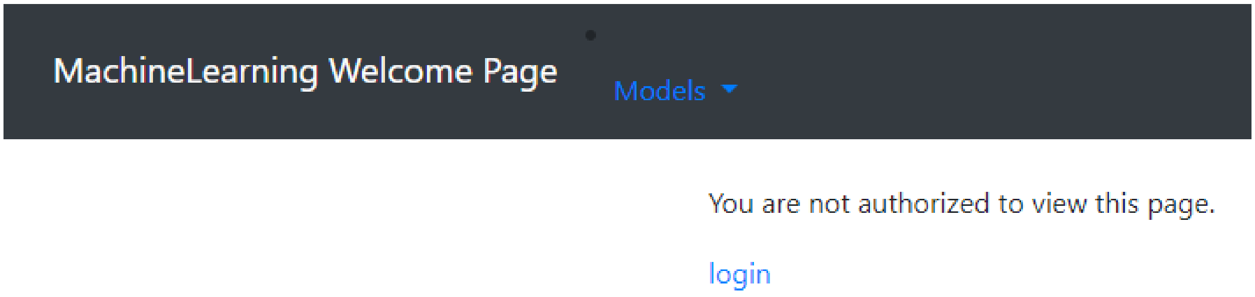
Password confirmation:

••••••••••

Enter the same password as before, for verification.

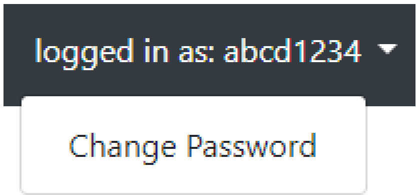
Restrictions:
-User name must be 150 characters or less
-Password must be 8 characters, and cannot be entirely numeric.
-Internal security controls prevent “commonly used” passwords from being accepted, i.e. “password” or “abcd1234”

After a successful submission, you’ll be taken to the dashboard page which will ask you to log in. You can follow the link to log in with your newly created user name.



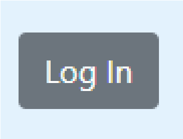
You are not authorized to view this page.

Passwords can be changed by selecting the drop-down under your username on the top navigation bar:



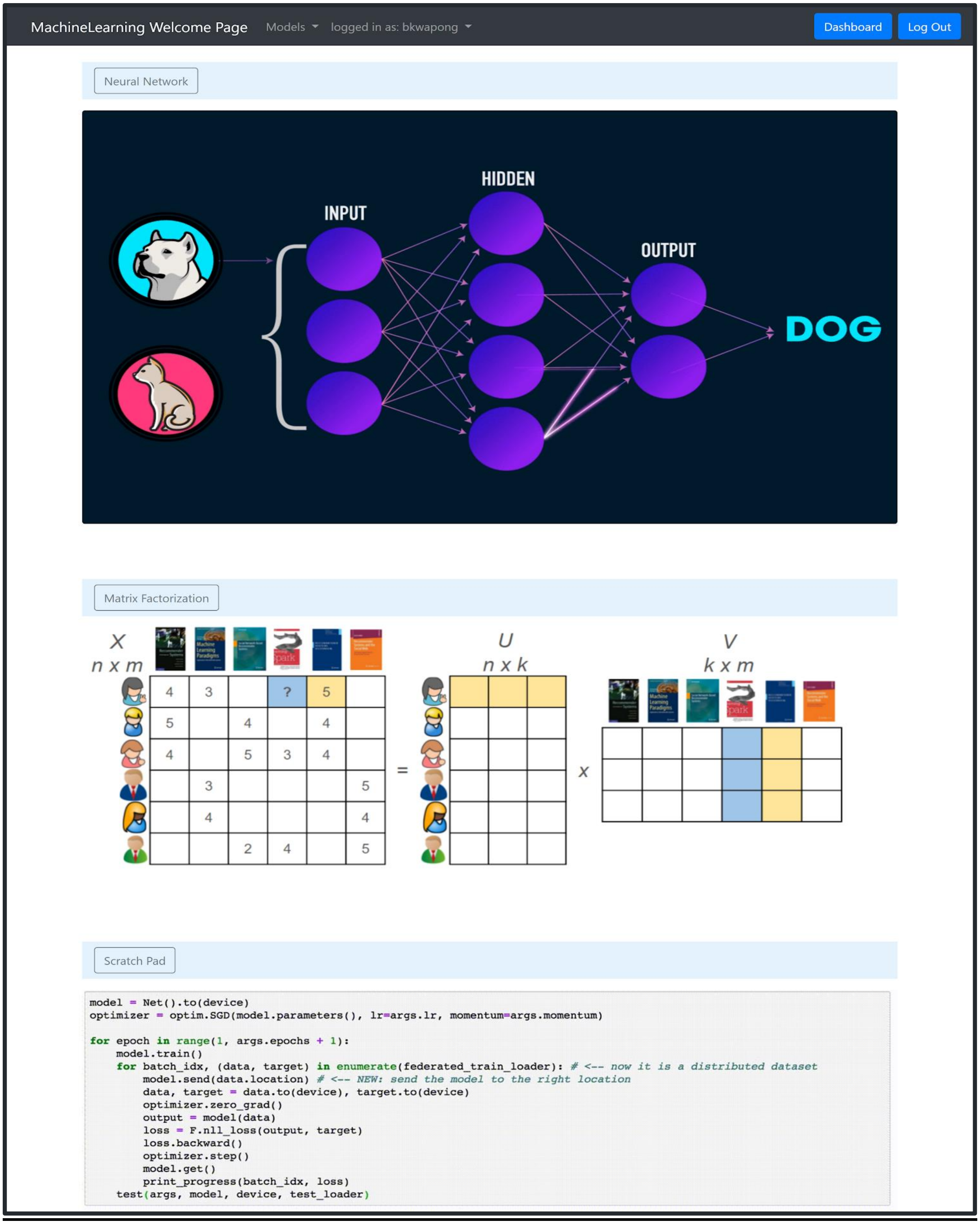
Login process and Navigation

Select a Log In widget on the screen:



Follow on-screen prompt for username and password. You will remain logged in for the lifecycle of your visit.

Dashboard



4	3		?	5	
5		4		4	
4		5	3	4	
	3				5
	4				4
		2	4		5

```
model = Net().to(device)
optimizer = optim.SGD(model.parameters(), lr=args.lr, momentum=args.momentum)

for epoch in range(1, args.epochs + 1):
    model.train()
    for batch_idx, (data, target) in enumerate(federated_train_loader): # <-- now it is a distributed dataset
        model.send(data.location) # <-- NEW: send the model to the right location
        data, target = data.to(device), target.to(device)
        optimizer.zero_grad()
        output = model(data)
        loss = F.nll_loss(output, target)
        loss.backward()
        optimizer.step()
        model.get()
        print_progress(batch_idx, loss)
    test(args, model, device, test_loader)
```

Matrix Factorization Model Backend

File Name: cf.py
Datasets: Movielens rating 100k and movielens 1M
Model: Collaborative user based filtering

Inputs from Views.py:
Train_perc - percentage of data to be used for training
N_sim_users- number of similar users to consider
N_movie_rec- number of movies to recommend to target users

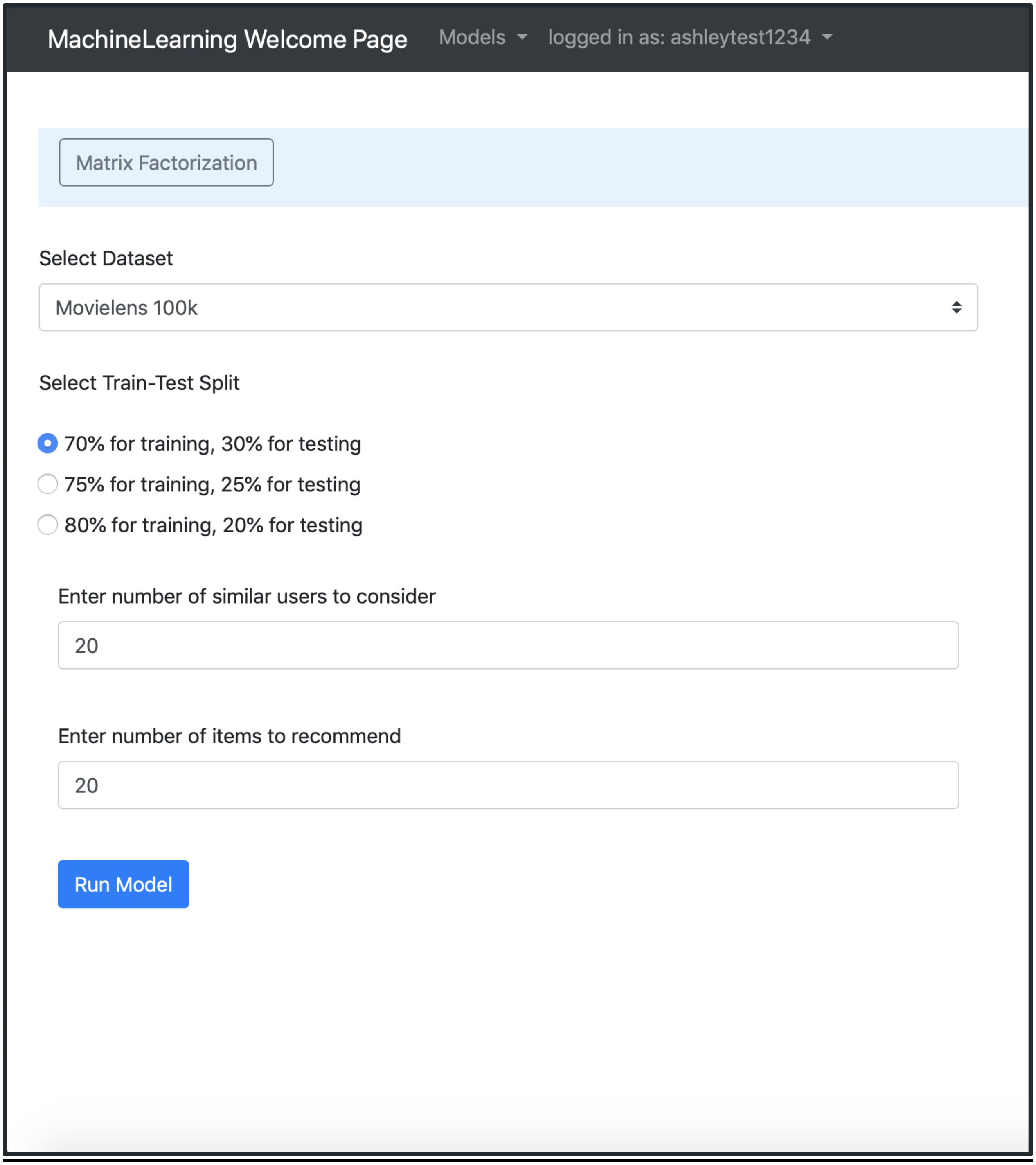
Functions and definitions:
Loadfile- loads datafile from user selection
Generate_dataset- splits file into train/test set based on user selection
Calc_user_sim- calculates user similarity matrix
Recommend- finds k similar users and recommend N movies based on user selection
Evaluate- calculates precision,recall and coverage

Output:
File- precision,recall, and coverage values calculated for n_sim_users to consider 5 +/-

Acknowledgement:
This implementation was adapted from **https://github.com/Lockvictor/MovieLens-RecSys**

Matrix Factorization Webpage

Inputs:



Select Dataset

Movielens 100k

Select Train-Test Split

● 70% for training, 30% for testing
○ 75% for training, 25% for testing
○ 80% for training, 20% for testing

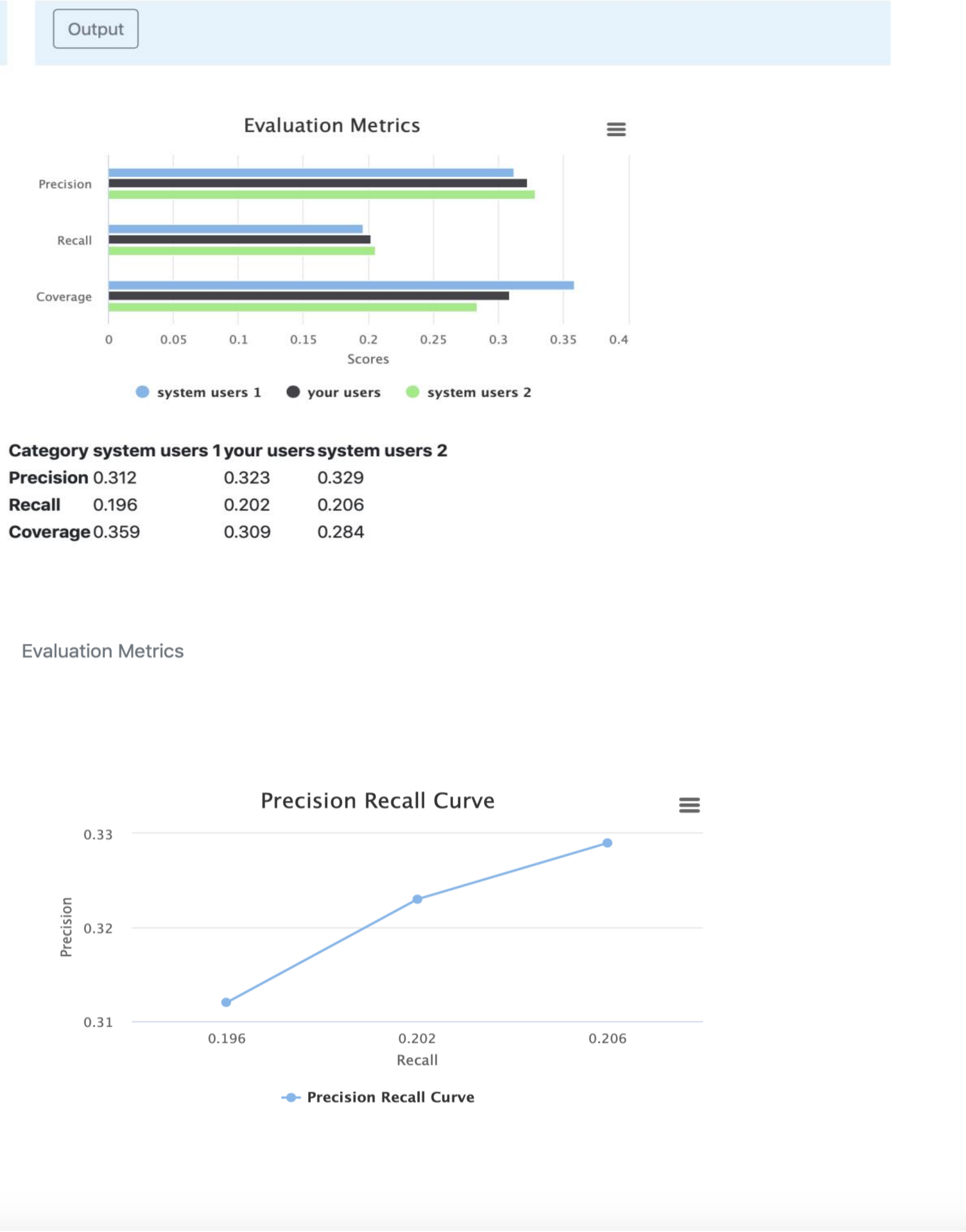
Enter number of similar users to consider

20

Enter number of items to recommend

20

Webpage Output



Category	system users 1	your users	system users 2
Precision	0.312	0.323	0.329
Recall	0.196	0.202	0.206
Coverage	0.359	0.309	0.284

Evaluation Metrics

Matrix Factorization Testing

Backend Output:

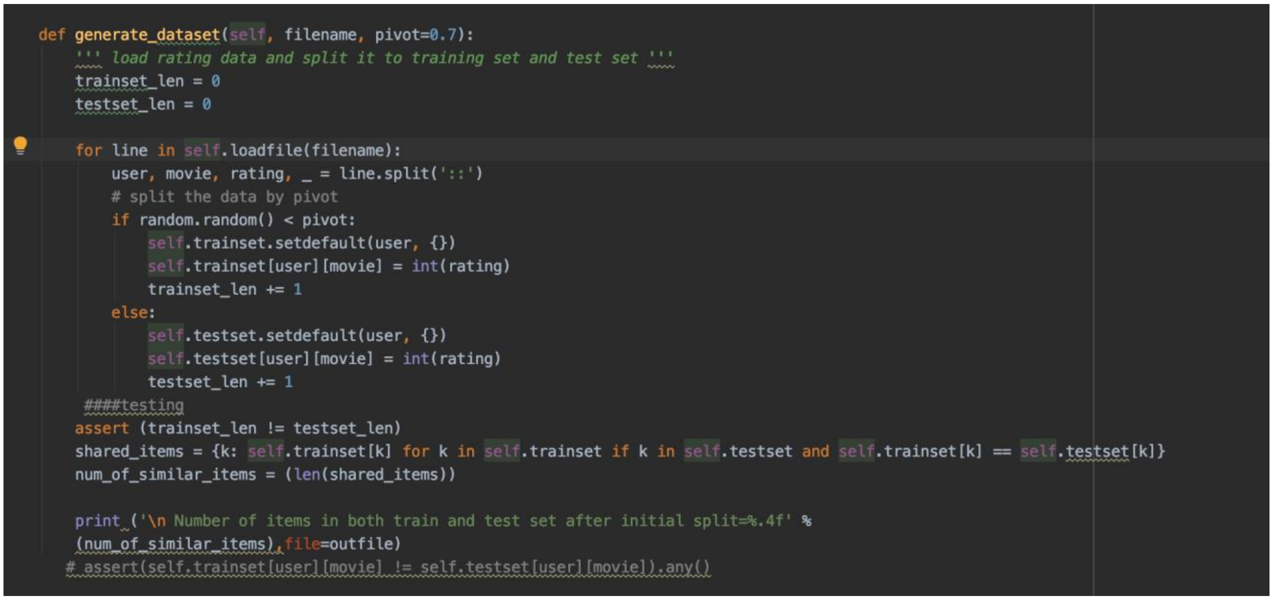
File: testmf.py
Output: unittest_results.txt

Functions:
def generate_dataset:

- **Asserts that test and train set length are not equal**
- **Checks that there are no shared items in train/test set**

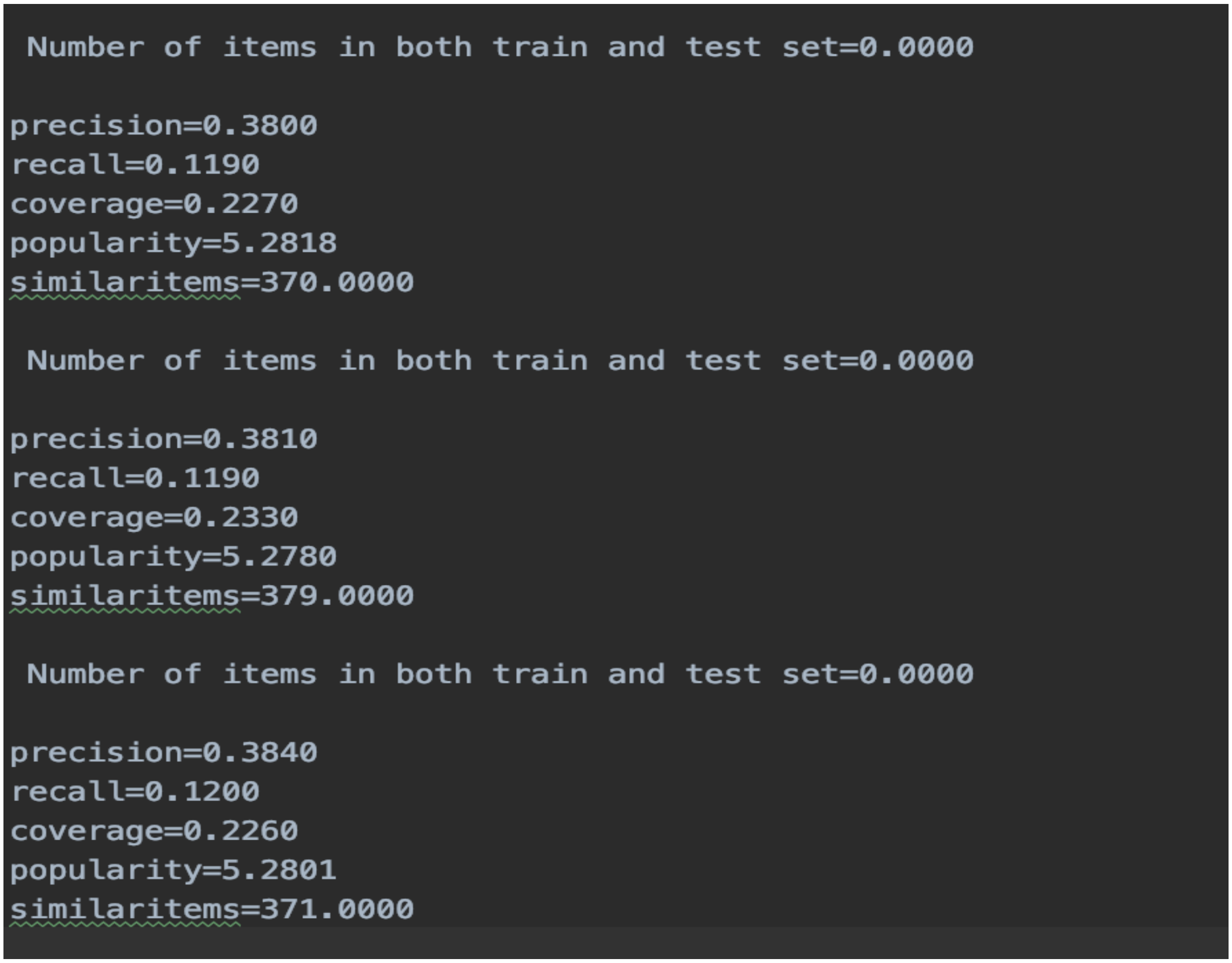
def evaluate:

- **Verifies the number of recommended items that are in testset is changing with each run**



```python
def generate_dataset(self, filename, pivot=0.7):
    ''' load rating data and split it to training set and test set '''
    trainset_len = 0
    testset_len = 0

    for line in self.loadfile(filename):
        user, movie, rating, _ = line.split('::')
        # split the data by pivot
        if random.random() < pivot:
            self.trainset.setdefault(user, {})
            self.trainset[user][movie] = int(rating)
            trainset_len += 1
        else:
            self.testset.setdefault(user, {})
            self.testset[user][movie] = int(rating)
            testset_len += 1
     ####testing
    assert (trainset_len != testset_len)
    shared_items = {k: self.trainset[k] for k in self.trainset if k in self.testset and self.trainset[k] == self.testset[k]}
    num_of_similar_items = (len(shared_items))

    print ('\n Number of items in both train and test set after initial split=%.4f' %
    (num_of_similar_items),file=outfile)
   # assert(self.trainset[user][movie]  != self.testset[user][movie]).any()
```

Backend Test File

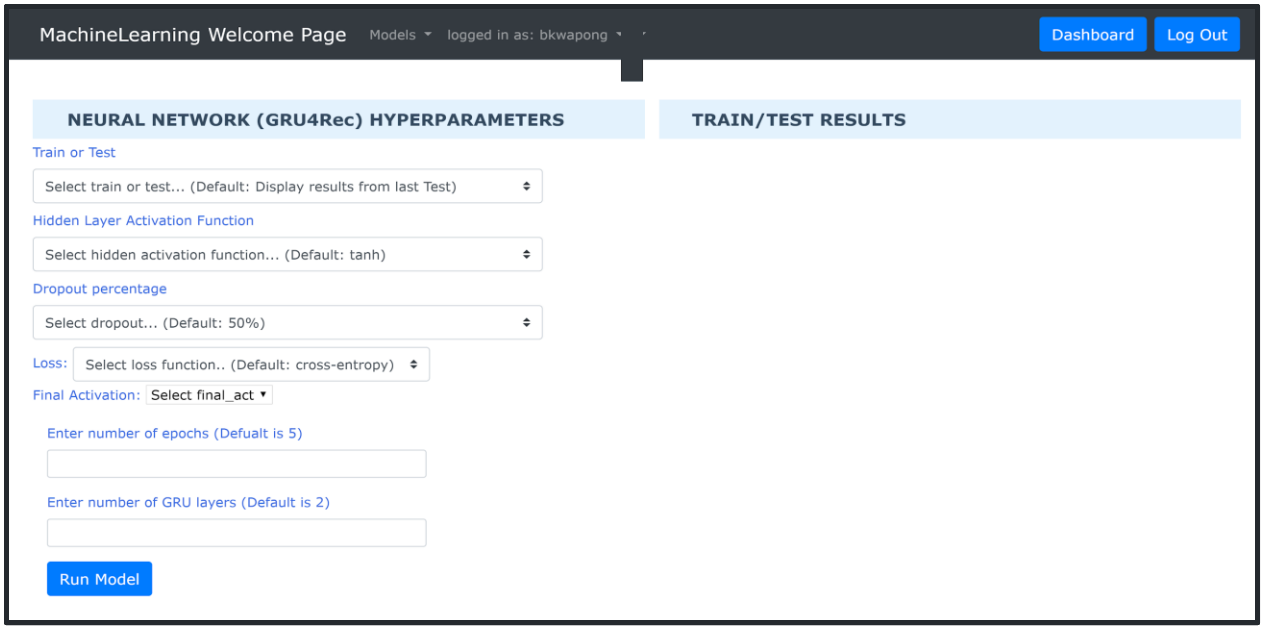


```
 Number of items in both train and test set=0.0000

precision=0.3800
recall=0.1190
coverage=0.2270
popularity=5.2818
similaritems=370.0000

 Number of items in both train and test set=0.0000

precision=0.3810
recall=0.1190
coverage=0.2330
popularity=5.2780
similaritems=379.0000

 Number of items in both train and test set=0.0000

precision=0.3840
recall=0.1200
coverage=0.2260
popularity=5.2801
similaritems=371.0000
```

Frontend Test Output:

Number of Users/ Number of Items/ Training/Testing Split	Precision	Recall	Coverage
Users: 25,20,15 , Items:20 Training/Testing: 70/30	0.312 0.323 0.329	0.196 0.202 0.206	0.358 0.31 0.284
Users: 40,35 30, Items:35, Training/ Testing: 75/25	0.256 0.26 0.264	0.289 0.294 0.299	0.351 0.31 0.309
Users: 95,100,105, Items:100, Training/Testing: 80/20	0.122 0.121 0.123	0.574 0.579 0.575	0.422 0.412 0.41

Neural Network Model



NEURAL NETWORK (GRU4Rec) HYPERPARAMETERS

TRAIN/TEST RESULTS

Train or Test

Select train or test... (Default: Display results from last Test)

Hidden Layer Activation Function

Select hidden activation function... (Default: tanh)

Dropout percentage

Select dropout... (Default: 50%)

Loss: Select loss function.. (Default: cross-entropy)

Final Activation: Select final_act

Enter number of epochs (Defualt is 5)

Enter number of GRU layers (Default is 2)

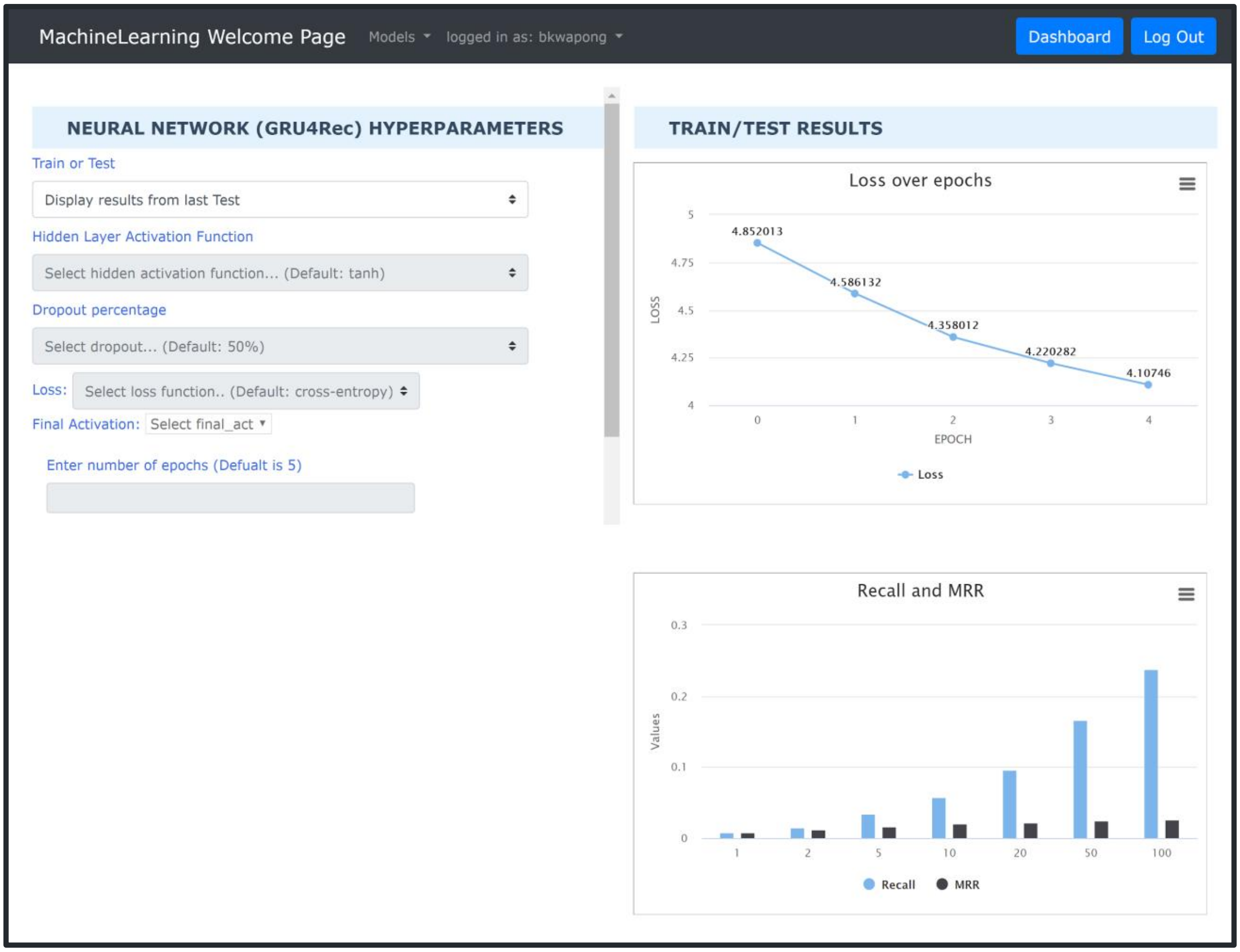
Backend Filenames: main.py, model.py, evaluation.py
Dataset: Movielens rating 100k
Model: GRU4Rec

Inputs from Views.py:
--layer, default=2, type=int
--size, default=128, type=int
--epoch, default=10, type=int
--lr, default=0.001, type=float
--train, default=1, type=int
--test, default=2, type=int
--hidden_act, default='tanh', type=str
--final_act, default='softmax', type=str
--loss, default='cross-entropy', type=str
--dropout, default='0.5', type=float

Output:
File - Precision & MRR @ k = [1, 2, 5, 10, 20, 50,100]

Acknowledgement: This is a keras implementation of *GRU4Rec*, which was described in "Session-based Recommendations With Recurrent Neural Networks". See paper: http://arxiv.org/abs/1511.06939.

Neural Network example (display results from last test)



MachineLearning Welcome Page   Models ▾   logged in as: bkwapong ▾   Dashboard   Log Out

NEURAL NETWORK (GRU4Rec) HYPERPARAMETERS

Train or Test: Display results from last Test

Hidden Layer Activation Function: Select hidden activation function... (Default: tanh)

Dropout percentage: Select dropout... (Default: 50%)

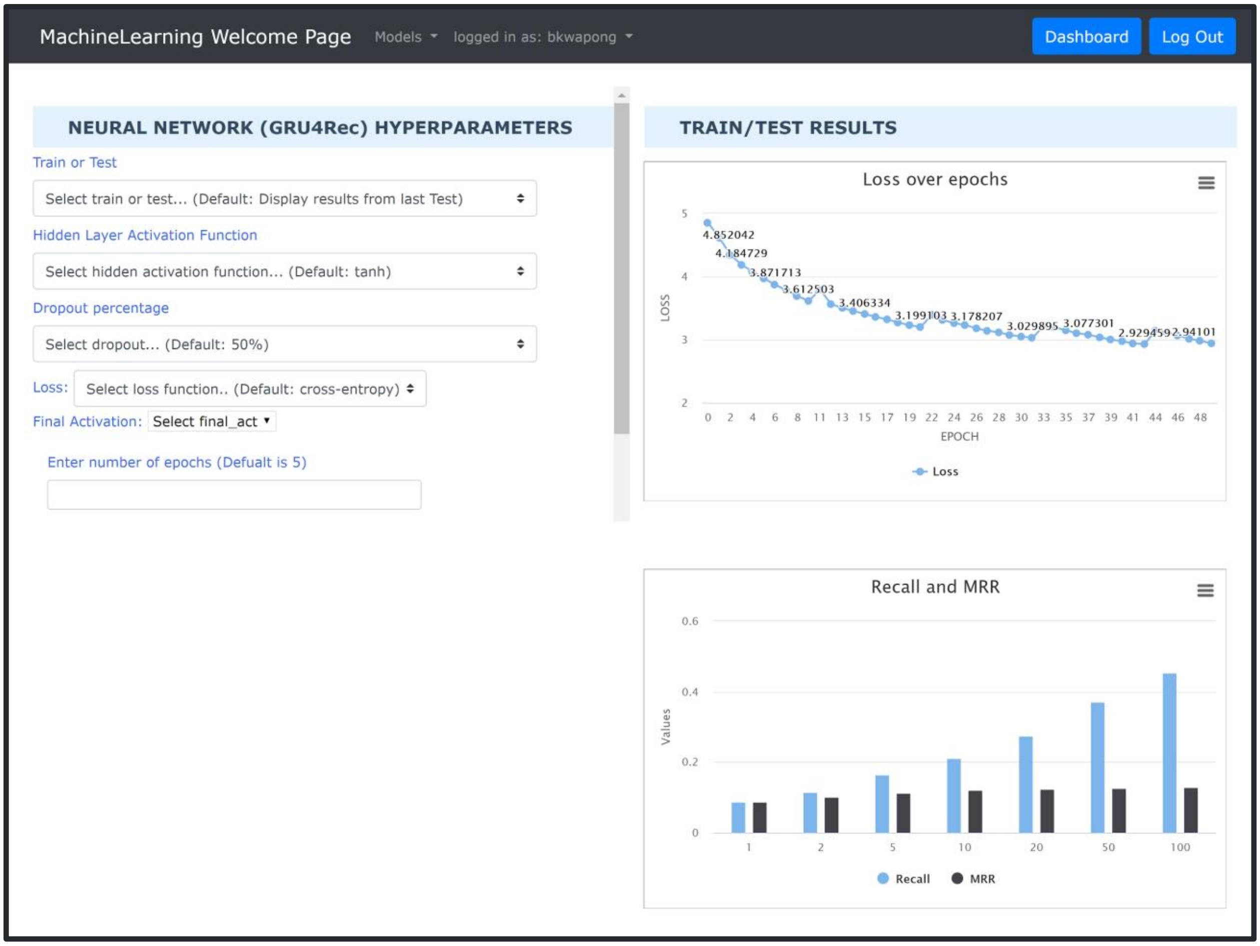
Loss: Select loss function.. (Default: cross-entropy)

Final Activation: Select final_act

Enter number of epochs (Defualt is 5)

TRAIN/TEST RESULTS

Neural Network example (Train and test on 50 epochs**)**



NEURAL NETWORK (GRU4Rec) HYPERPARAMETERS

Train or Test

Select train or test... (Default: Display results from last Test)

Hidden Layer Activation Function

Select hidden activation function... (Default: tanh)

Dropout percentage

Select dropout... (Default: 50%)

Loss: Select loss function.. (Default: cross-entropy)

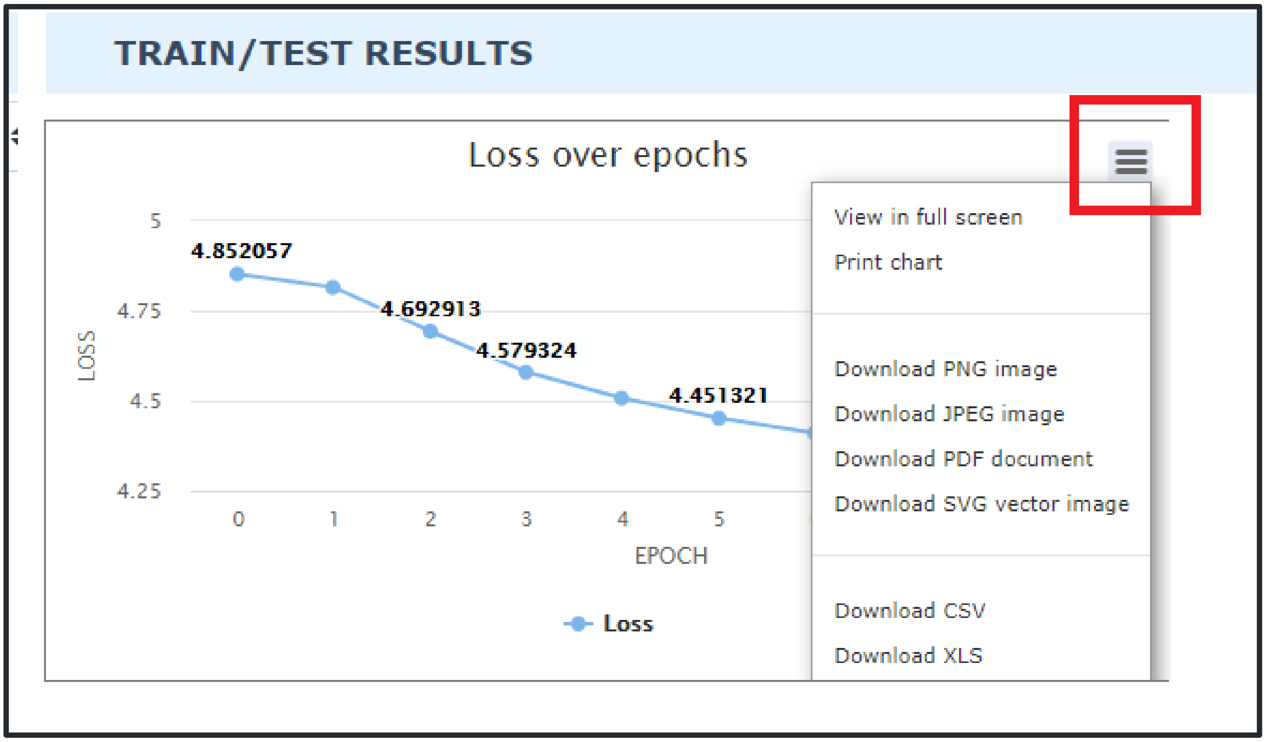
Final Activation: Select final_act

Enter number of epochs (Defualt is 5)

TRAIN/TEST RESULTS

Exporting Results from the chart

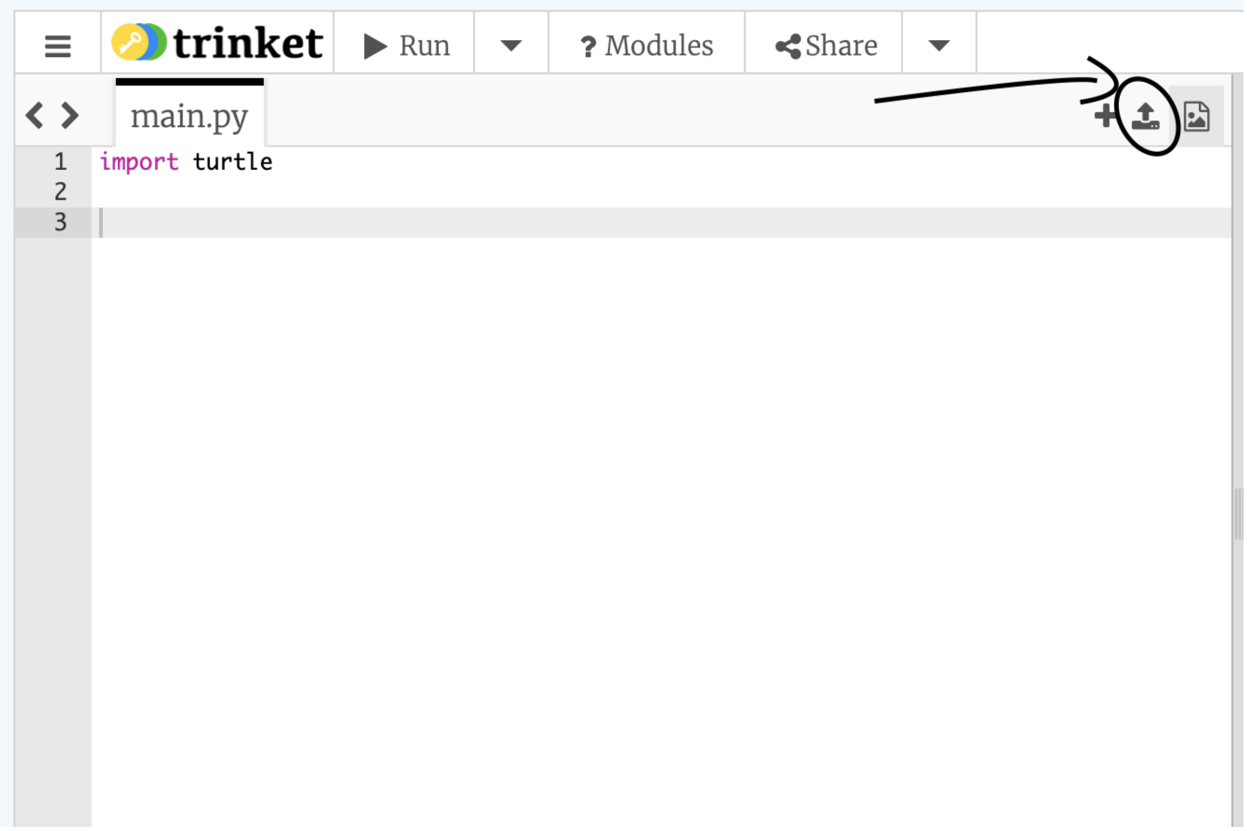
Each of the charts has the ability to export results in both graphical and pure data interfaces. Select the set of lines icon on the top right of the graph, and either download a picture version, or a .csv or .xls export of the raw data.



Scratch Pad

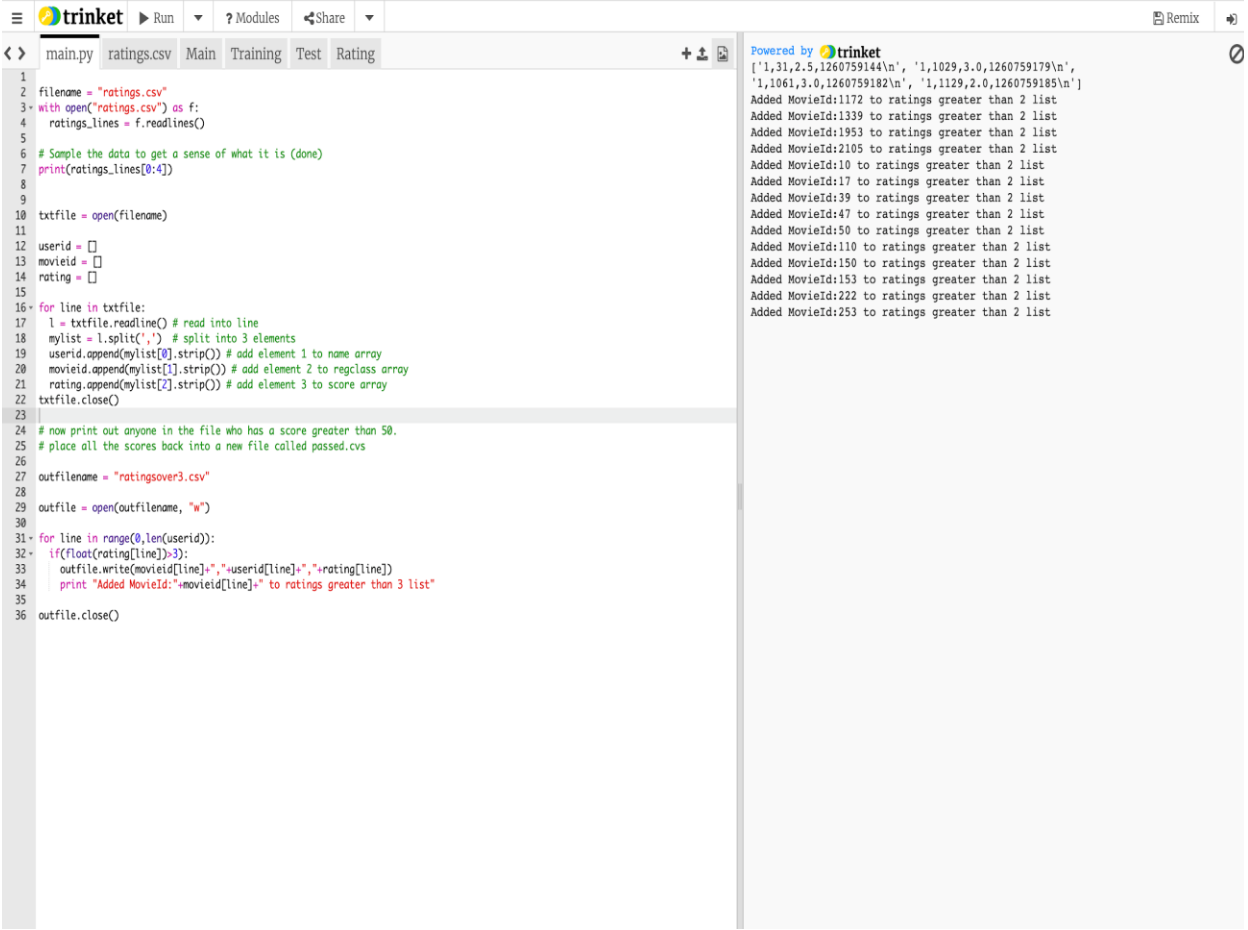
(Powered by Trinket.io)

File Upload:



```
import turtle
```

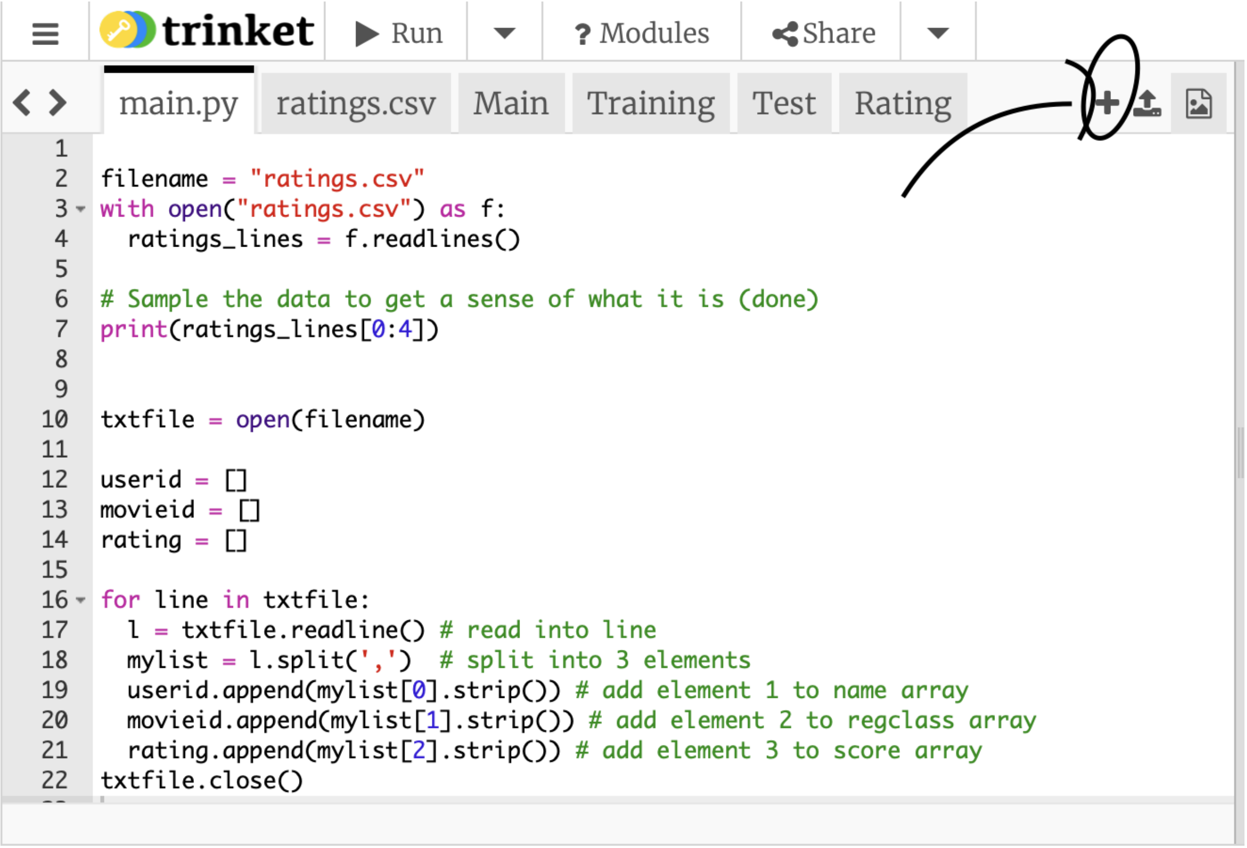
Reading File/ Writing to Output



```
filename = "ratings.csv"
with open("ratings.csv") as f:
  ratings_lines = f.readlines()

# Sample the data to get a sense of what it is (done)
print(ratings_lines[0:4])


txtfile = open(filename)

userid = []
movieid = []
rating = []

for line in txtfile:
  l = txtfile.readline() # read into line
  mylist = l.split(',')  # split into 3 elements
  userid.append(mylist[0].strip()) # add element 1 to name array
  movieid.append(mylist[1].strip()) # add element 2 to regclass array
  rating.append(mylist[2].strip()) # add element 3 to score array
txtfile.close()

# now print out anyone in the file who has a score greater than 50.
# place all the scores back into a new file called passed.cvs

outfilename = "ratingsover3.csv"

outfile = open(outfilename, "w")

for line in range(0,len(userid)):
  if(float(rating[line])>3):
    outfile.write(movieid[line]+","+userid[line]+","+rating[line])
    print "Added MovieId:"+movieid[line]+" to ratings greater than 3 list"

outfile.close()
```

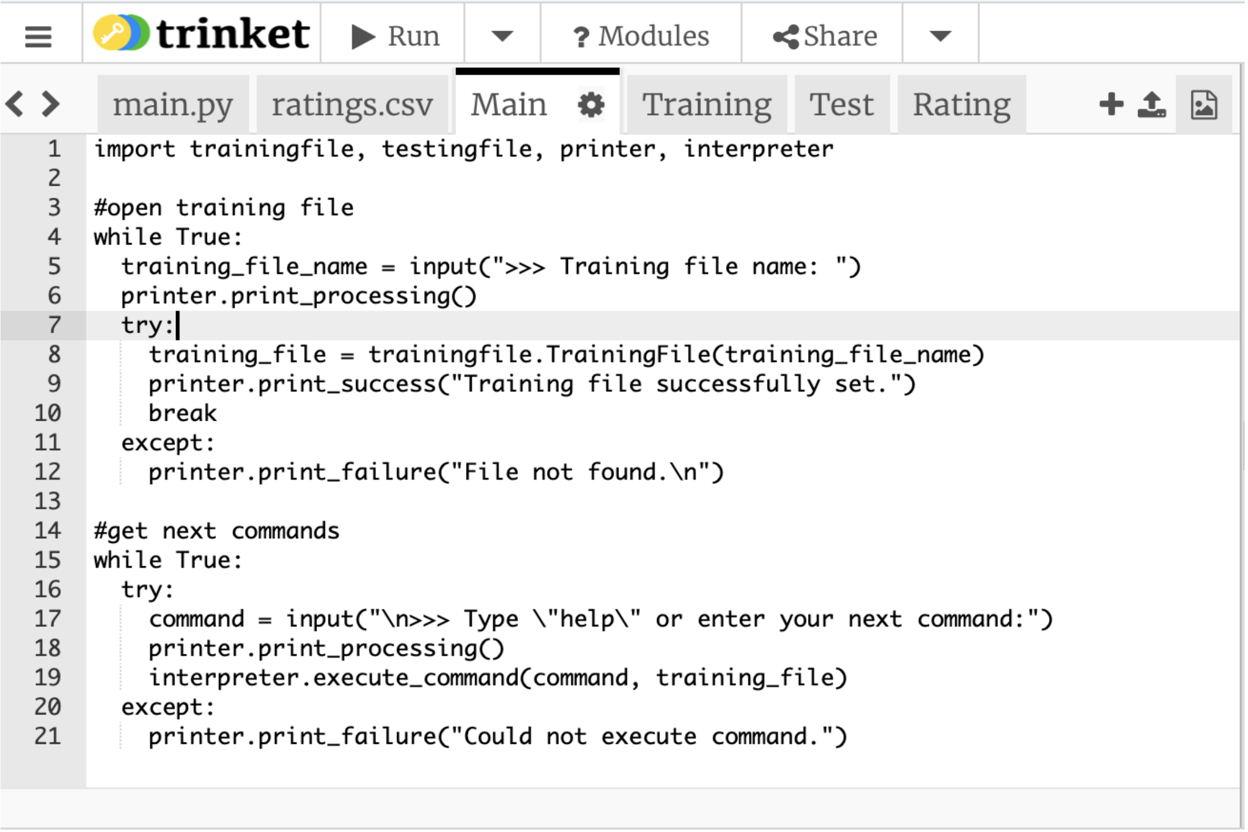
```
['1,31,2.5,1260759144\n', '1,1029,3.0,1260759179\n',
'1,1061,3.0,1260759182\n', '1,1129,2.0,1260759185\n']
Added MovieId:1172 to ratings greater than 2 list
Added MovieId:1339 to ratings greater than 2 list
Added MovieId:1953 to ratings greater than 2 list
Added MovieId:2105 to ratings greater than 2 list
Added MovieId:10 to ratings greater than 2 list
Added MovieId:17 to ratings greater than 2 list
Added MovieId:39 to ratings greater than 2 list
Added MovieId:47 to ratings greater than 2 list
Added MovieId:50 to ratings greater than 2 list
Added MovieId:110 to ratings greater than 2 list
Added MovieId:150 to ratings greater than 2 list
Added MovieId:153 to ratings greater than 2 list
Added MovieId:222 to ratings greater than 2 list
Added MovieId:253 to ratings greater than 2 list
```

Creating Multiple Files



```

filename = "ratings.csv"
with open("ratings.csv") as f:
  ratings_lines = f.readlines()

# Sample the data to get a sense of what it is (done)
print(ratings_lines[0:4])


txtfile = open(filename)

userid = []
movieid = []
rating = []

for line in txtfile:
  l = txtfile.readline() # read into line
  mylist = l.split(',')  # split into 3 elements
  userid.append(mylist[0].strip()) # add element 1 to name array
  movieid.append(mylist[1].strip()) # add element 2 to regclass array
  rating.append(mylist[2].strip()) # add element 3 to score array
txtfile.close()
```

Import Multiple Files

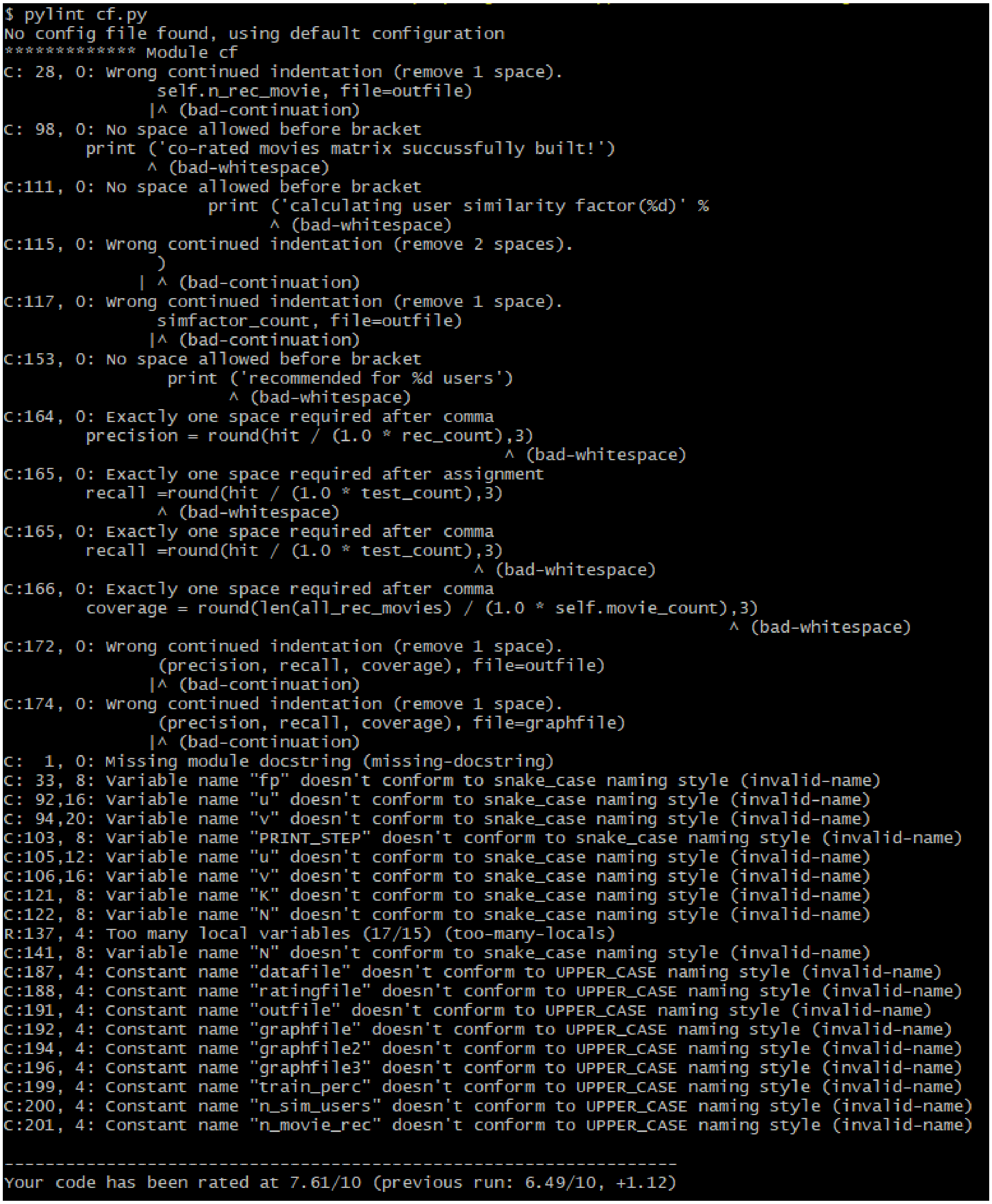


```python
import trainingfile, testingfile, printer, interpreter

#open training file
while True:
  training_file_name = input(">>> Training file name: ")
  printer.print_processing()
  try:
    training_file = trainingfile.TrainingFile(training_file_name)
    printer.print_success("Training file successfully set.")
    break
  except:
    printer.print_failure("File not found.\n")

#get next commands
while True:
  try:
    command = input("\n>>> Type \"help\" or enter your next command:")
    printer.print_processing()
    interpreter.execute_command(command, training_file)
  except:
    printer.print_failure("Could not execute command.")
```

Coding Standard Evaluation

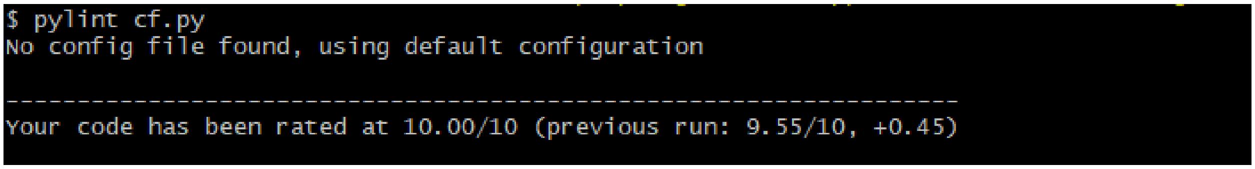
(PYTHON)

Matrix Factorization Code (Conformance Testing)



```
$ pylint cf.py
No config file found, using default configuration
************* Module cf
C: 28, 0: Wrong continued indentation (remove 1 space).
               self.n_rec_movie, file=outfile)
              |^ (bad-continuation)
C: 98, 0: No space allowed before bracket
        print ('co-rated movies matrix succussfully built!')
              ^ (bad-whitespace)
C:111, 0: No space allowed before bracket
                    print ('calculating user similarity factor(%d)' %
                          ^ (bad-whitespace)
C:115, 0: Wrong continued indentation (remove 2 spaces).
               )
             | ^ (bad-continuation)
C:117, 0: Wrong continued indentation (remove 1 space).
               simfactor_count, file=outfile)
              |^ (bad-continuation)
C:153, 0: No space allowed before bracket
                print ('recommended for %d users')
                      ^ (bad-whitespace)
C:164, 0: Exactly one space required after comma
        precision = round(hit / (1.0 * rec_count),3)
                                                 ^ (bad-whitespace)
C:165, 0: Exactly one space required after assignment
        recall =round(hit / (1.0 * test_count),3)
               ^ (bad-whitespace)
C:165, 0: Exactly one space required after comma
        recall =round(hit / (1.0 * test_count),3)
                                              ^ (bad-whitespace)
C:166, 0: Exactly one space required after comma
        coverage = round(len(all_rec_movies) / (1.0 * self.movie_count),3)
                                                                       ^ (bad-whitespace)
C:172, 0: Wrong continued indentation (remove 1 space).
               (precision, recall, coverage), file=outfile)
              |^ (bad-continuation)
C:174, 0: Wrong continued indentation (remove 1 space).
               (precision, recall, coverage), file=graphfile)
              |^ (bad-continuation)
C:  1, 0: Missing module docstring (missing-docstring)
C: 33, 8: Variable name "fp" doesn't conform to snake_case naming style (invalid-name)
C: 92,16: Variable name "u" doesn't conform to snake_case naming style (invalid-name)
C: 94,20: Variable name "v" doesn't conform to snake_case naming style (invalid-name)
C:103, 8: Variable name "PRINT_STEP" doesn't conform to snake_case naming style (invalid-name)
C:105,12: Variable name "u" doesn't conform to snake_case naming style (invalid-name)
C:106,16: Variable name "v" doesn't conform to snake_case naming style (invalid-name)
C:121, 8: Variable name "K" doesn't conform to snake_case naming style (invalid-name)
C:122, 8: Variable name "N" doesn't conform to snake_case naming style (invalid-name)
R:137, 4: Too many local variables (17/15) (too-many-locals)
C:141, 8: Variable name "N" doesn't conform to snake_case naming style (invalid-name)
C:187, 4: Constant name "datafile" doesn't conform to UPPER_CASE naming style (invalid-name)
C:188, 4: Constant name "ratingfile" doesn't conform to UPPER_CASE naming style (invalid-name)
C:191, 4: Constant name "outfile" doesn't conform to UPPER_CASE naming style (invalid-name)
C:192, 4: Constant name "graphfile" doesn't conform to UPPER_CASE naming style (invalid-name)
C:194, 4: Constant name "graphfile2" doesn't conform to UPPER_CASE naming style (invalid-name)
C:196, 4: Constant name "graphfile3" doesn't conform to UPPER_CASE naming style (invalid-name)
C:199, 4: Constant name "train_perc" doesn't conform to UPPER_CASE naming style (invalid-name)
C:200, 4: Constant name "n_sim_users" doesn't conform to UPPER_CASE naming style (invalid-name)
C:201, 4: Constant name "n_movie_rec" doesn't conform to UPPER_CASE naming style (invalid-name)

------------------------------------------------------------------
Your code has been rated at 7.61/10 (previous run: 6.49/10, +1.12)
```

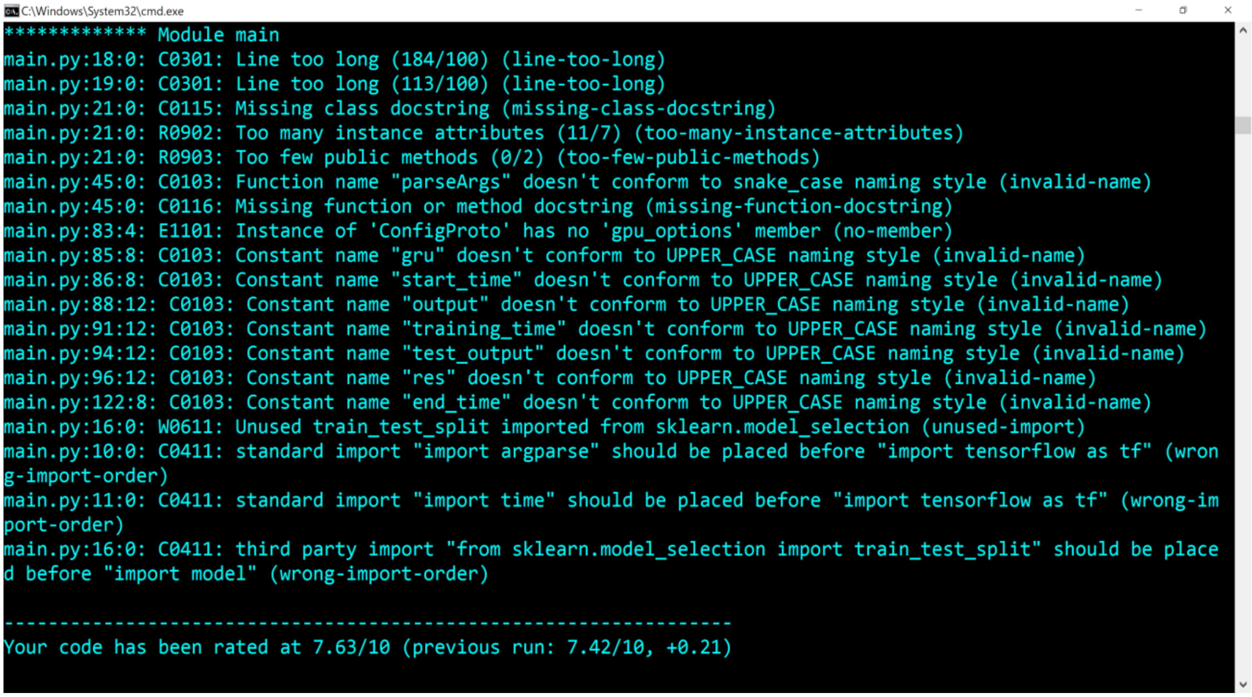
Matrix Factorization Code (After Corrective Actions)



```
$ pylint cf.py
No config file found, using default configuration

------------------------------------------------------------------
Your code has been rated at 10.00/10 (previous run: 9.55/10, +0.45)
```

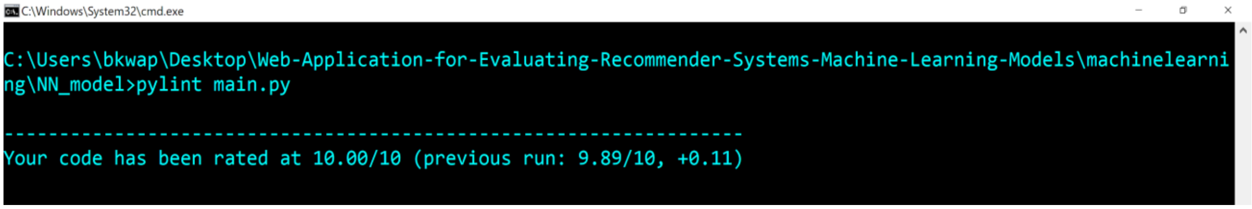
Neural_Network_Code (Conformance Testing)

main.py



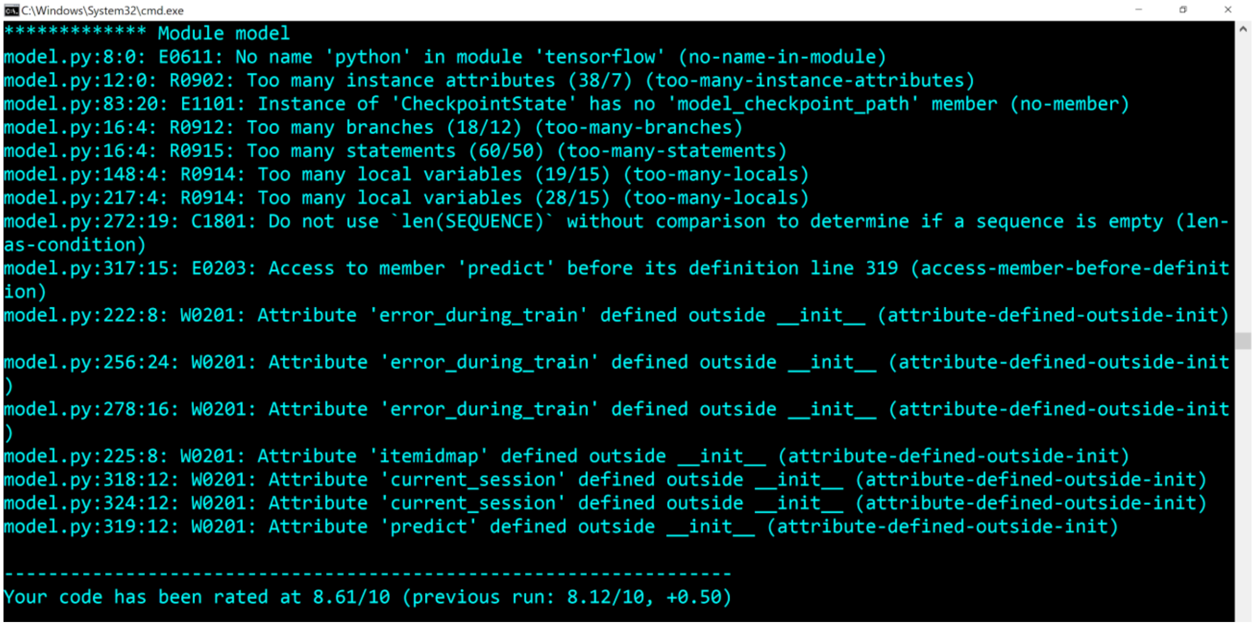
```
************* Module main
main.py:18:0: C0301: Line too long (184/100) (line-too-long)
main.py:19:0: C0301: Line too long (113/100) (line-too-long)
main.py:21:0: C0115: Missing class docstring (missing-class-docstring)
main.py:21:0: R0902: Too many instance attributes (11/7) (too-many-instance-attributes)
main.py:21:0: R0903: Too few public methods (0/2) (too-few-public-methods)
main.py:45:0: C0103: Function name "parseArgs" doesn't conform to snake_case naming style (invalid-name)
main.py:45:0: C0116: Missing function or method docstring (missing-function-docstring)
main.py:83:4: E1101: Instance of 'ConfigProto' has no 'gpu_options' member (no-member)
main.py:85:8: C0103: Constant name "gru" doesn't conform to UPPER_CASE naming style (invalid-name)
main.py:86:8: C0103: Constant name "start_time" doesn't conform to UPPER_CASE naming style (invalid-name)
main.py:88:12: C0103: Constant name "output" doesn't conform to UPPER_CASE naming style (invalid-name)
main.py:91:12: C0103: Constant name "training_time" doesn't conform to UPPER_CASE naming style (invalid-name)
main.py:94:12: C0103: Constant name "test_output" doesn't conform to UPPER_CASE naming style (invalid-name)
main.py:96:12: C0103: Constant name "res" doesn't conform to UPPER_CASE naming style (invalid-name)
main.py:122:8: C0103: Constant name "end_time" doesn't conform to UPPER_CASE naming style (invalid-name)
main.py:16:0: W0611: Unused train_test_split imported from sklearn.model_selection (unused-import)
main.py:10:0: C0411: standard import "import argparse" should be placed before "import tensorflow as tf" (wron
g-import-order)
main.py:11:0: C0411: standard import "import time" should be placed before "import tensorflow as tf" (wrong-im
port-order)
main.py:16:0: C0411: third party import "from sklearn.model_selection import train_test_split" should be place
d before "import model" (wrong-import-order)

------------------------------------------------------------------
Your code has been rated at 7.63/10 (previous run: 7.42/10, +0.21)
```

main.py (After Corrective Actions)



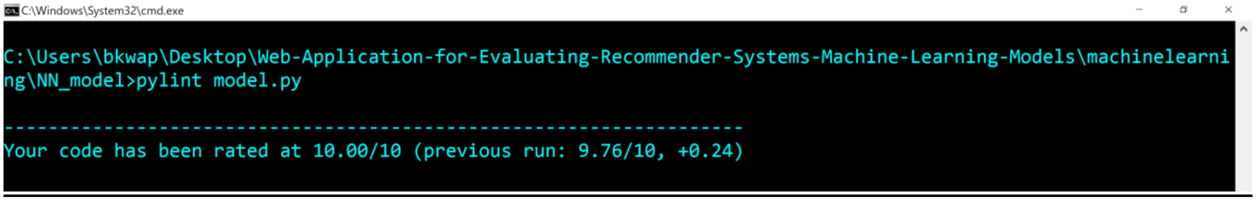
```
C:\Users\bkwap\Desktop\Web-Application-for-Evaluating-Recommender-Systems-Machine-Learning-Models\machinelearni
ng\NN_model>pylint main.py

-------------------------------------------------------------------
Your code has been rated at 10.00/10 (previous run: 9.89/10, +0.11)
```

model.py



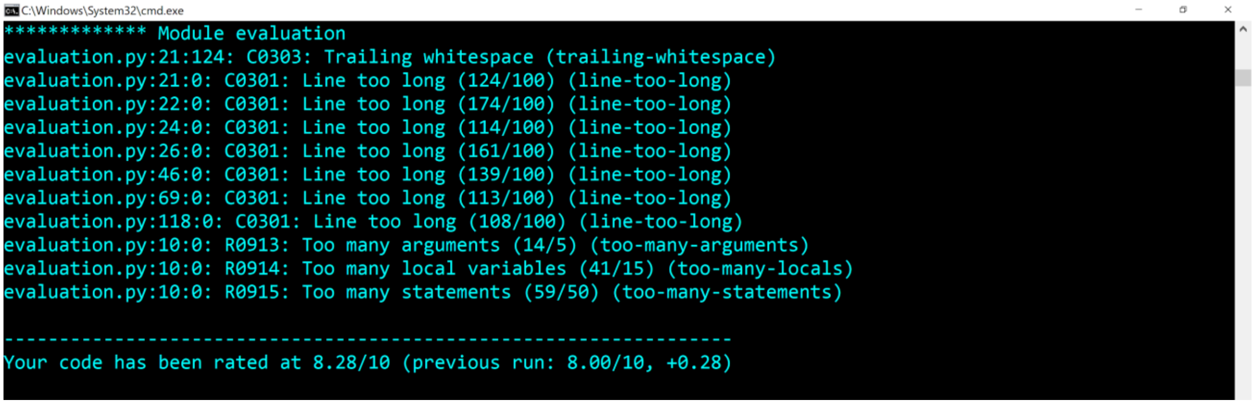
```
************* Module model
model.py:8:0: E0611: No name 'python' in module 'tensorflow' (no-name-in-module)
model.py:12:0: R0902: Too many instance attributes (38/7) (too-many-instance-attributes)
model.py:83:20: E1101: Instance of 'CheckpointState' has no 'model_checkpoint_path' member (no-member)
model.py:16:4: R0912: Too many branches (18/12) (too-many-branches)
model.py:16:4: R0915: Too many statements (60/50) (too-many-statements)
model.py:148:4: R0914: Too many local variables (19/15) (too-many-locals)
model.py:217:4: R0914: Too many local variables (28/15) (too-many-locals)
model.py:272:19: C1801: Do not use `len(SEQUENCE)` without comparison to determine if a sequence is empty (len-
as-condition)
model.py:317:15: E0203: Access to member 'predict' before its definition line 319 (access-member-before-definit
ion)
model.py:222:8: W0201: Attribute 'error_during_train' defined outside __init__ (attribute-defined-outside-init)

model.py:256:24: W0201: Attribute 'error_during_train' defined outside __init__ (attribute-defined-outside-init
)
model.py:278:16: W0201: Attribute 'error_during_train' defined outside __init__ (attribute-defined-outside-init
)
model.py:225:8: W0201: Attribute 'itemidmap' defined outside __init__ (attribute-defined-outside-init)
model.py:318:12: W0201: Attribute 'current_session' defined outside __init__ (attribute-defined-outside-init)
model.py:324:12: W0201: Attribute 'current_session' defined outside __init__ (attribute-defined-outside-init)
model.py:319:12: W0201: Attribute 'predict' defined outside __init__ (attribute-defined-outside-init)

------------------------------------------------------------------
Your code has been rated at 8.61/10 (previous run: 8.12/10, +0.50)
```

model.py (After Corrective Actions**)**



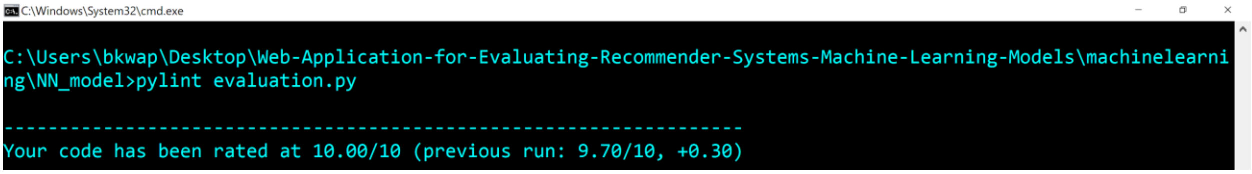
```

C:\Users\bkwap\Desktop\Web-Application-for-Evaluating-Recommender-Systems-Machine-Learning-Models\machinelearni
ng\NN_model>pylint model.py

-------------------------------------------------------------------
Your code has been rated at 10.00/10 (previous run: 9.76/10, +0.24)
```

evaluation.py



```
************* Module evaluation
evaluation.py:21:124: C0303: Trailing whitespace (trailing-whitespace)
evaluation.py:21:0: C0301: Line too long (124/100) (line-too-long)
evaluation.py:22:0: C0301: Line too long (174/100) (line-too-long)
evaluation.py:24:0: C0301: Line too long (114/100) (line-too-long)
evaluation.py:26:0: C0301: Line too long (161/100) (line-too-long)
evaluation.py:46:0: C0301: Line too long (139/100) (line-too-long)
evaluation.py:69:0: C0301: Line too long (113/100) (line-too-long)
evaluation.py:118:0: C0301: Line too long (108/100) (line-too-long)
evaluation.py:10:0: R0913: Too many arguments (14/5) (too-many-arguments)
evaluation.py:10:0: R0914: Too many local variables (41/15) (too-many-locals)
evaluation.py:10:0: R0915: Too many statements (59/50) (too-many-statements)

------------------------------------------------------------------
Your code has been rated at 8.28/10 (previous run: 8.00/10, +0.28)
```

evaluation.py (After Corrective Actions**)**



```
C:\Users\bkwap\Desktop\Web-Application-for-Evaluating-Recommender-Systems-Machine-Learning-Models\machinelearni
ng\NN_model>pylint evaluation.py

-------------------------------------------------------------------
Your code has been rated at 10.00/10 (previous run: 9.70/10, +0.30)
```