# LookAlike trainer2 deployment document

## 1. Codes structure

| name | description |
|---|---|
| Train_tf2.py | Main training code for training & testing LookAlike model |
| Model_tf2.py | Code for defining DIN model in tf2 |
| Utils_tf2.py | Utility functions mainly to transfer input data format from tf1 to tf2 |
| __init__.py | Required to be defined as a python package |

Train_tf2.py is the program to start with training and testing

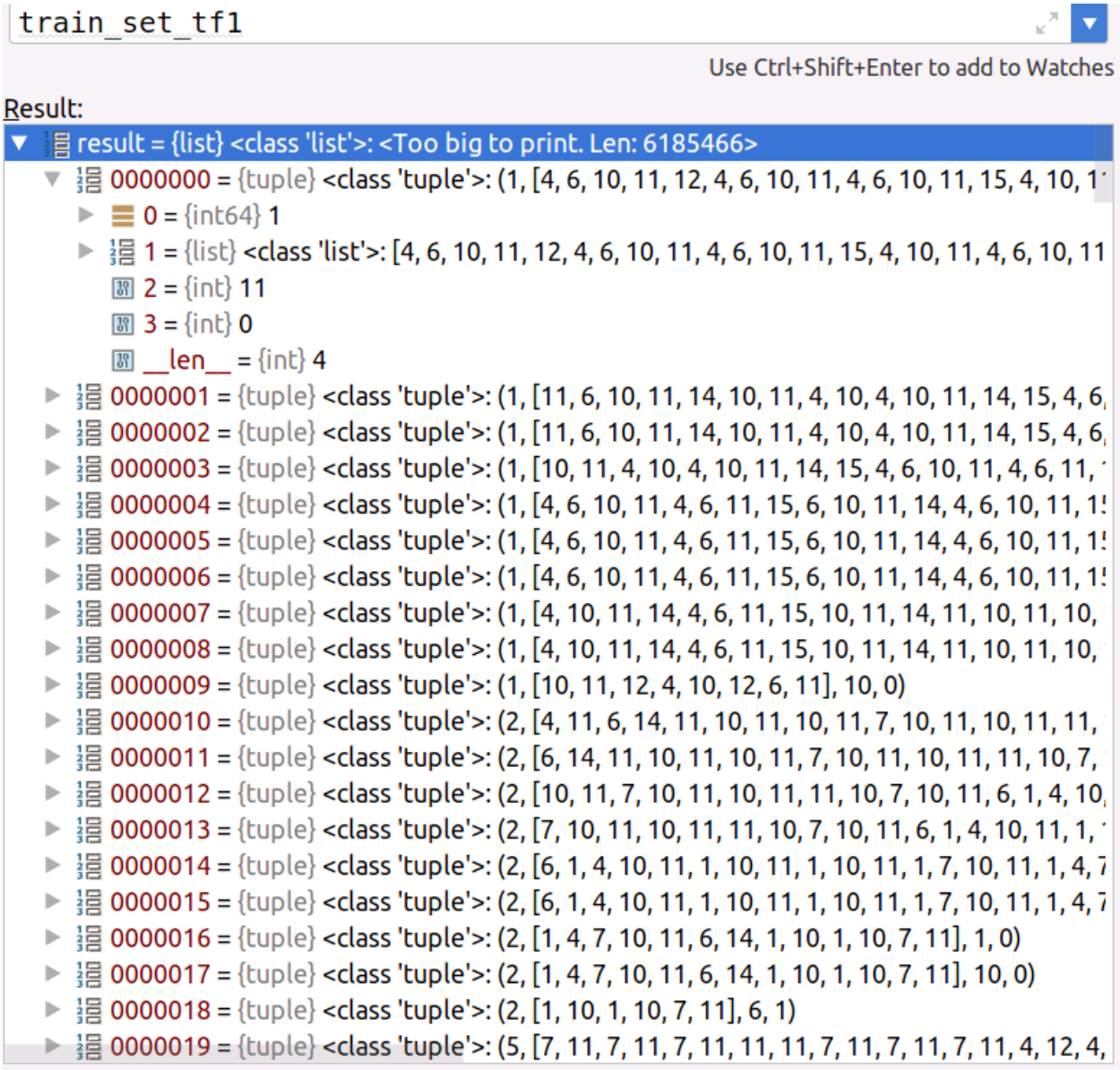
## 2. Inputs

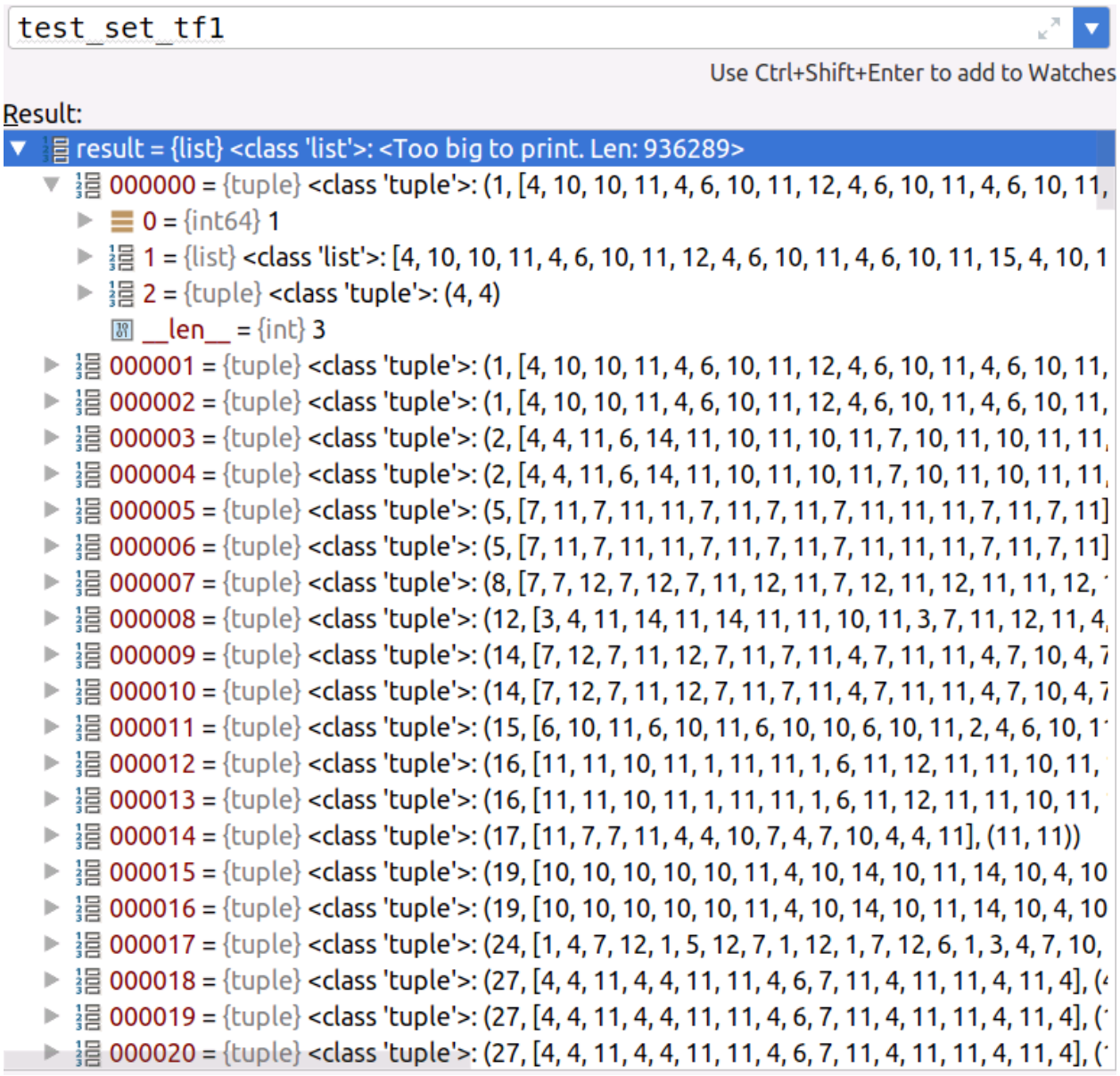
The format of input data file is pickle.

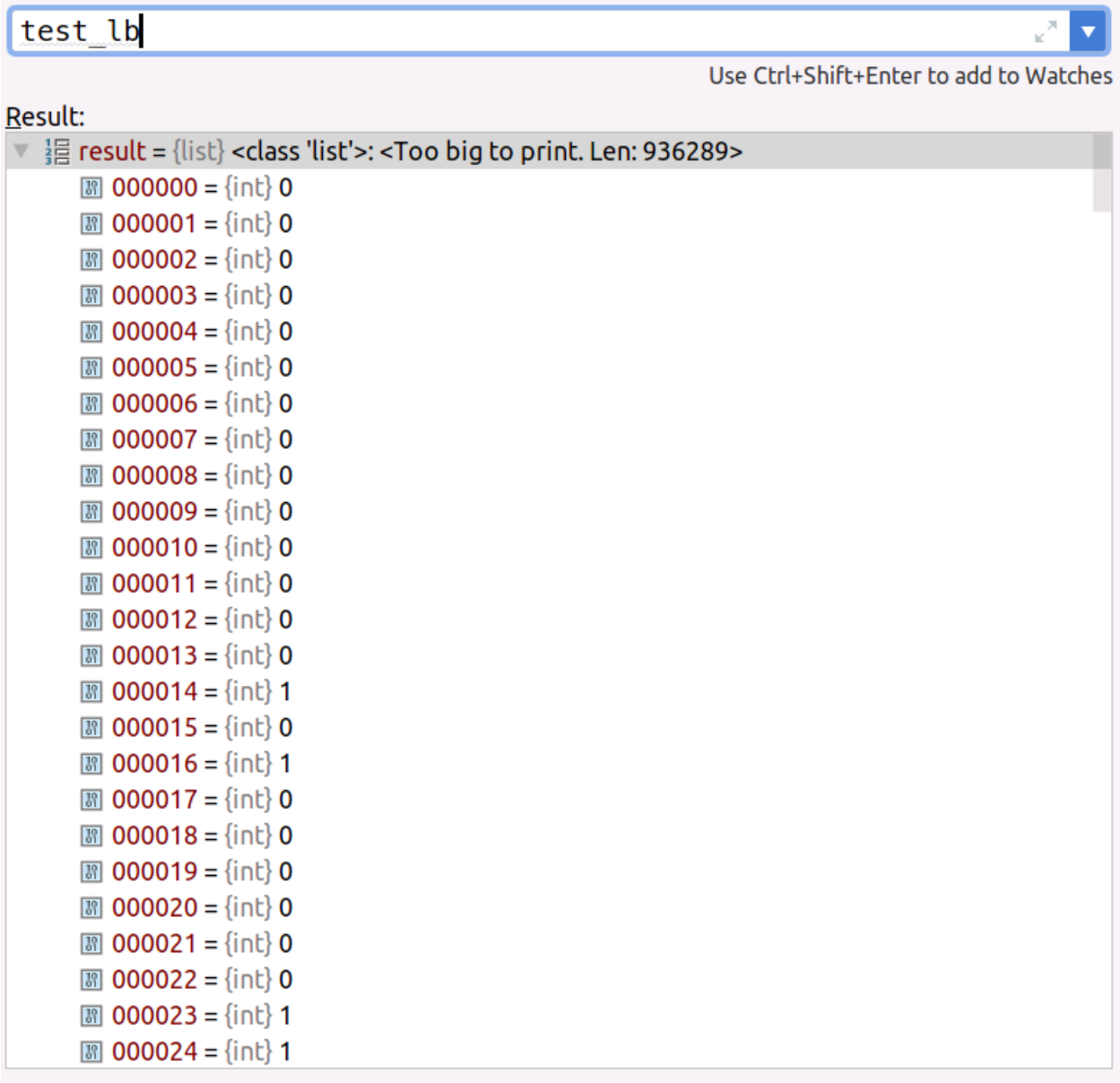
There are 2 pickles with exactly the same format as tf1

- training and testing data file, such as: ad_dataset_lookalike_xxxxxx.pkl
- testing label file, such as: label_lookalike_xxxxxx.pkl

These 2 files have the same format as tf1 trainer as shown below.


train_set_tf1
Use Ctrl+Shift+Enter to add to Watches
Result:
result = {list} <class 'list'>: <Too big to print. Len: 6185466>
0000000 = {tuple} <class 'tuple'>: (1, [4, 6, 10, 11, 12, 4, 6, 10, 11, 4, 6, 10, 11, 15, 4, 10, 1
0 = {int64} 1
1 = {list} <class 'list'>: [4, 6, 10, 11, 12, 4, 6, 10, 11, 4, 6, 10, 11, 15, 4, 10, 11, 4, 6, 10, 11
2 = {int} 11
3 = {int} 0
__len__ = {int} 4
0000001 = {tuple} <class 'tuple'>: (1, [11, 6, 10, 11, 14, 10, 11, 4, 10, 4, 10, 11, 14, 15, 4, 6,
0000002 = {tuple} <class 'tuple'>: (1, [11, 6, 10, 11, 14, 10, 11, 4, 10, 4, 10, 11, 14, 15, 4, 6,
0000003 = {tuple} <class 'tuple'>: (1, [10, 11, 4, 10, 4, 10, 11, 14, 15, 4, 6, 10, 11, 4, 6, 11,
0000004 = {tuple} <class 'tuple'>: (1, [4, 6, 10, 11, 4, 6, 11, 15, 6, 10, 11, 14, 4, 6, 10, 11, 1
0000005 = {tuple} <class 'tuple'>: (1, [4, 6, 10, 11, 4, 6, 11, 15, 6, 10, 11, 14, 4, 6, 10, 11, 1
0000006 = {tuple} <class 'tuple'>: (1, [4, 6, 10, 11, 4, 6, 11, 15, 6, 10, 11, 14, 4, 6, 10, 11, 1
0000007 = {tuple} <class 'tuple'>: (1, [4, 10, 11, 14, 4, 6, 11, 15, 10, 11, 14, 11, 10, 11, 10,
0000008 = {tuple} <class 'tuple'>: (1, [4, 10, 11, 14, 4, 6, 11, 15, 10, 11, 14, 11, 10, 11, 10,
0000009 = {tuple} <class 'tuple'>: (1, [10, 11, 12, 4, 10, 12, 6, 11], 10, 0)
0000010 = {tuple} <class 'tuple'>: (2, [4, 11, 6, 14, 11, 10, 11, 10, 11, 7, 10, 11, 10, 11, 11,
0000011 = {tuple} <class 'tuple'>: (2, [6, 14, 11, 10, 11, 10, 11, 7, 10, 11, 10, 11, 11, 10, 7,
0000012 = {tuple} <class 'tuple'>: (2, [10, 11, 7, 10, 11, 10, 11, 11, 10, 7, 10, 11, 6, 1, 4, 10,
0000013 = {tuple} <class 'tuple'>: (2, [7, 10, 11, 10, 11, 11, 10, 7, 10, 11, 6, 1, 4, 10, 11, 1,
0000014 = {tuple} <class 'tuple'>: (2, [6, 1, 4, 10, 11, 1, 10, 11, 1, 10, 11, 1, 7, 10, 11, 1, 4, 7
0000015 = {tuple} <class 'tuple'>: (2, [6, 1, 4, 10, 11, 1, 10, 11, 1, 10, 11, 1, 7, 10, 11, 1, 4, 7
0000016 = {tuple} <class 'tuple'>: (2, [1, 4, 7, 10, 11, 6, 14, 1, 10, 1, 10, 7, 11], 1, 0)
0000017 = {tuple} <class 'tuple'>: (2, [1, 4, 7, 10, 11, 6, 14, 1, 10, 1, 10, 7, 11], 10, 0)
0000018 = {tuple} <class 'tuple'>: (2, [1, 10, 1, 10, 7, 11], 6, 1)
0000019 = {tuple} <class 'tuple'>: (5, [7, 11, 7, 11, 7, 11, 11, 11, 7, 11, 7, 11, 7, 11, 4, 12, 4,


test_set_tf1
Use Ctrl+Shift+Enter to add to Watches
Result:
result = {list} <class 'list'>: <Too big to print. Len: 936289>
000000 = {tuple} <class 'tuple'>: (1, [4, 10, 10, 11, 4, 6, 10, 11, 12, 4, 6, 10, 11, 4, 6, 10, 11,
0 = {int64} 1
1 = {list} <class 'list'>: [4, 10, 10, 11, 4, 6, 10, 11, 12, 4, 6, 10, 11, 4, 6, 10, 11, 15, 4, 10, 1
2 = {tuple} <class 'tuple'>: (4, 4)
__len__ = {int} 3
000001 = {tuple} <class 'tuple'>: (1, [4, 10, 10, 11, 4, 6, 10, 11, 12, 4, 6, 10, 11, 4, 6, 10, 11,
000002 = {tuple} <class 'tuple'>: (1, [4, 10, 10, 11, 4, 6, 10, 11, 12, 4, 6, 10, 11, 4, 6, 10, 11,
000003 = {tuple} <class 'tuple'>: (2, [4, 4, 11, 6, 14, 11, 10, 11, 10, 11, 7, 10, 11, 10, 11, 11,
000004 = {tuple} <class 'tuple'>: (2, [4, 4, 11, 6, 14, 11, 10, 11, 10, 11, 7, 10, 11, 10, 11, 11,
000005 = {tuple} <class 'tuple'>: (5, [7, 11, 7, 11, 11, 7, 11, 7, 11, 7, 11, 11, 11, 7, 11, 7, 11]
000006 = {tuple} <class 'tuple'>: (5, [7, 11, 7, 11, 11, 7, 11, 7, 11, 7, 11, 11, 11, 7, 11, 7, 11]
000007 = {tuple} <class 'tuple'>: (8, [7, 7, 12, 7, 12, 7, 11, 12, 11, 7, 12, 11, 12, 11, 11, 12,
000008 = {tuple} <class 'tuple'>: (12, [3, 4, 11, 14, 11, 14, 11, 11, 10, 11, 3, 7, 11, 12, 11, 4,
000009 = {tuple} <class 'tuple'>: (14, [7, 12, 7, 11, 12, 7, 11, 7, 11, 4, 7, 11, 11, 4, 7, 10, 4, 7
000010 = {tuple} <class 'tuple'>: (14, [7, 12, 7, 11, 12, 7, 11, 7, 11, 4, 7, 11, 11, 4, 7, 10, 4, 7
000011 = {tuple} <class 'tuple'>: (15, [6, 10, 11, 6, 10, 11, 6, 10, 10, 6, 10, 11, 2, 4, 6, 10, 1
000012 = {tuple} <class 'tuple'>: (16, [11, 11, 10, 11, 1, 11, 11, 1, 6, 11, 12, 11, 11, 10, 11,
000013 = {tuple} <class 'tuple'>: (16, [11, 11, 10, 11, 1, 11, 11, 1, 6, 11, 12, 11, 11, 10, 11,
000014 = {tuple} <class 'tuple'>: (17, [11, 7, 7, 11, 4, 4, 10, 7, 4, 7, 10, 4, 4, 11], (11, 11))
000015 = {tuple} <class 'tuple'>: (19, [10, 10, 10, 10, 10, 11, 4, 10, 14, 10, 11, 14, 10, 4, 10
000016 = {tuple} <class 'tuple'>: (19, [10, 10, 10, 10, 10, 11, 4, 10, 14, 10, 11, 14, 10, 4, 10
000017 = {tuple} <class 'tuple'>: (24, [1, 4, 7, 12, 1, 5, 12, 7, 1, 12, 1, 7, 12, 6, 1, 3, 4, 7, 10,
000018 = {tuple} <class 'tuple'>: (27, [4, 4, 11, 4, 4, 11, 11, 4, 6, 7, 11, 4, 11, 11, 4, 11, 4],
000019 = {tuple} <class 'tuple'>: (27, [4, 4, 11, 4, 4, 11, 11, 4, 6, 7, 11, 4, 11, 11, 4, 11, 4],
000020 = {tuple} <class 'tuple'>: (27, [4, 4, 11, 4, 4, 11, 11, 4, 6, 7, 11, 4, 11, 11, 4, 11, 4],




The function named “tf1_to_tf2_data_conversion” defined in utils_tf2.py converts the above mentions tf1 data format to tf2 data format and then saved in a pickle file named “train_test_lookalike.pkl”.

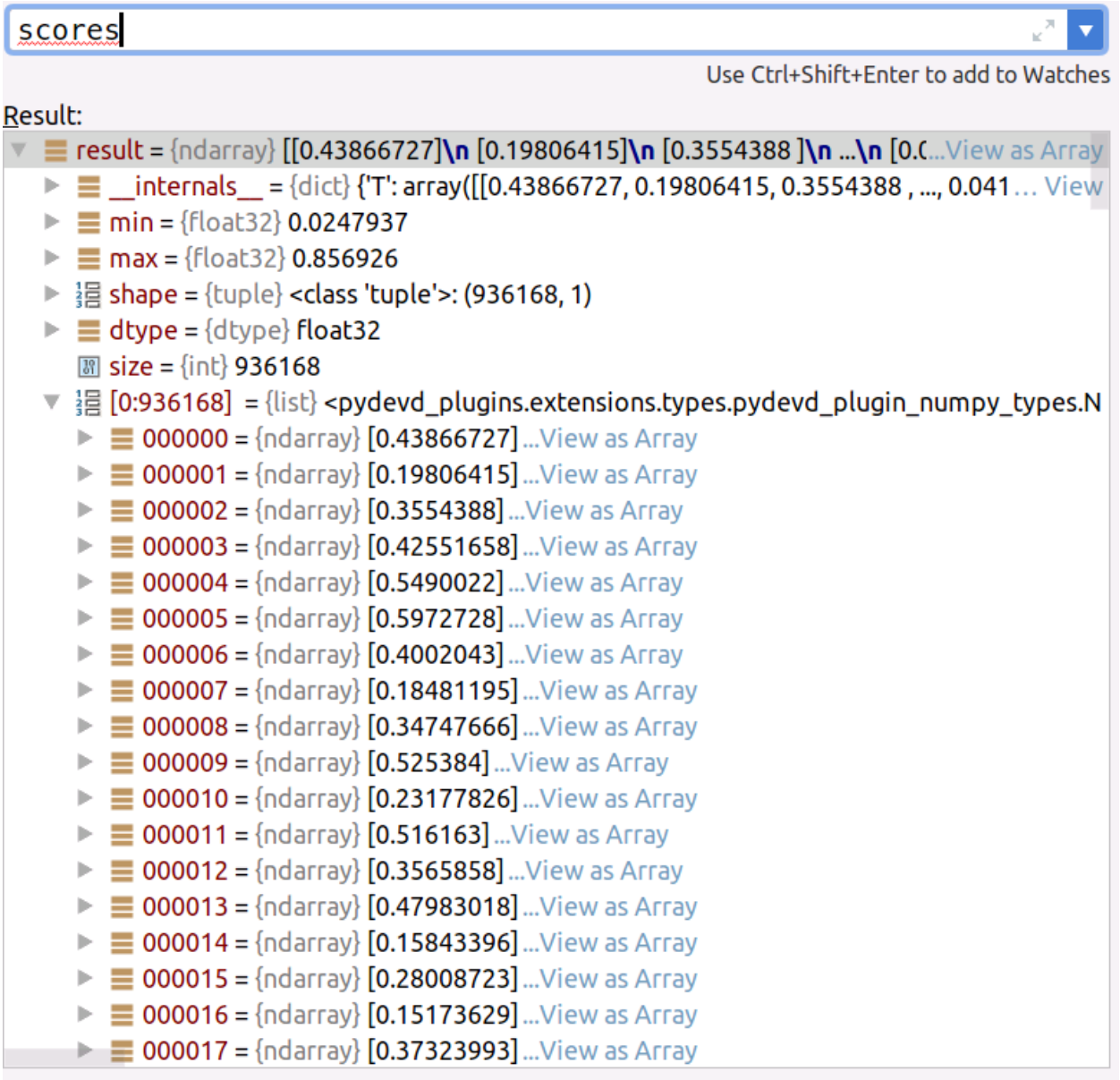
If “train_test_lookalike.pkl” already exists, train_tf2.py will skip the conversion step and read tf2 input data from “train_test_lookalike.pkl” directly.

Input data for trainer2 are stored in 2 tuples of variables, i.e. (train_X, train_y) and (test_x, test_y), they will be feed into function model.fit to train the tf2 DIN model

## 3. output

scores = model.predict(test_X, batch_size=batch_size)

will return scores for all testing samples as shown below




auc_test = model.evaluate(test_X, test_y, batch_size=batch_size)

will return AUC (area under curve – accuracy) for all testing samples, for example:

```
test AUC: 0.690566
```