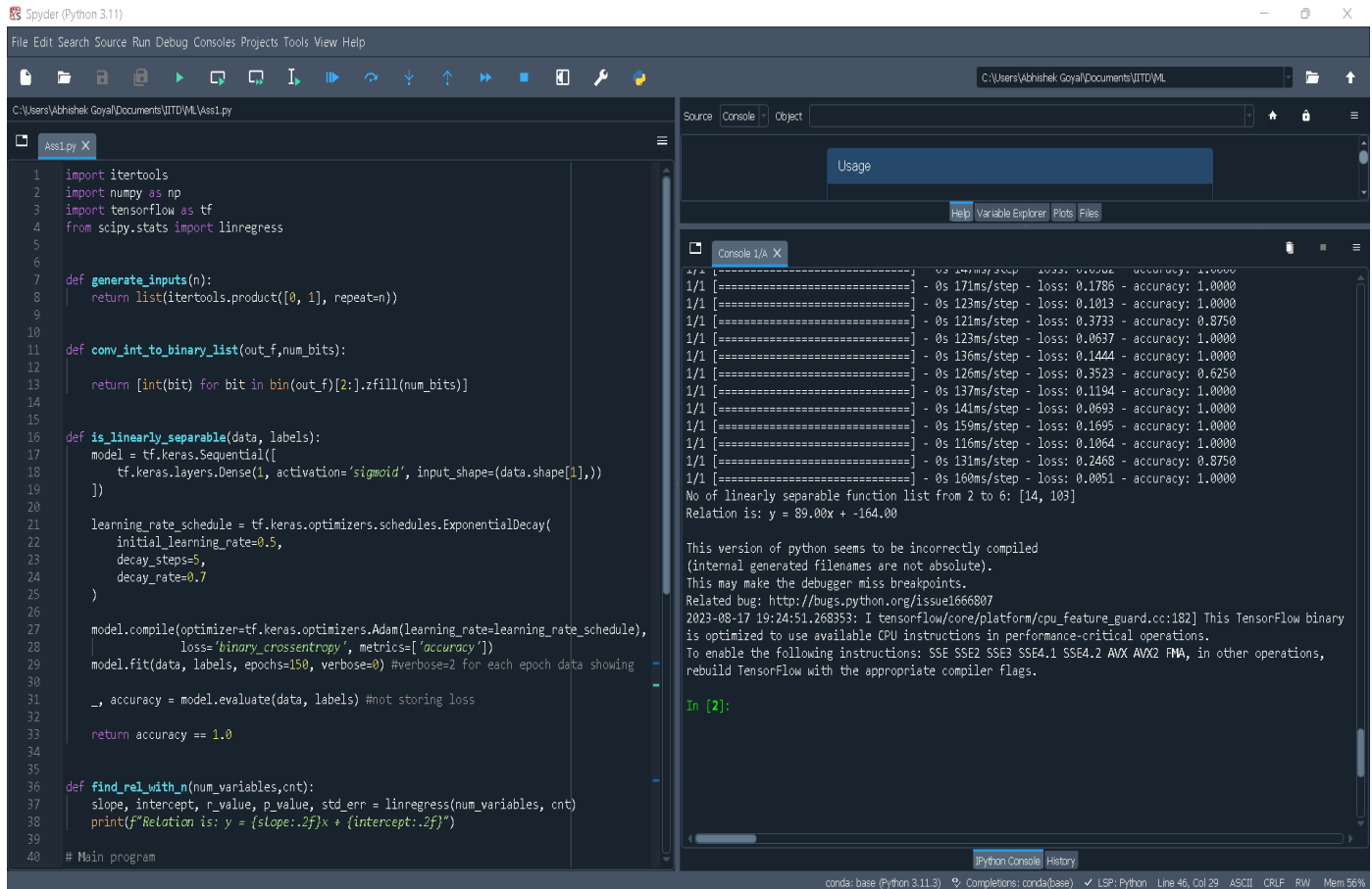


Assignment 01 Solution

Output produced for Input 2,3 only as other inputs were taking long time.



The screenshot shows the Spyder Python IDE interface. The left pane displays a Jupyter notebook with the following code:

```
1 import itertools
2 import numpy as np
3 import tensorflow as tf
4 from scipy.stats import linregress
5
6
7 def generate_inputs(n):
8     return list(itertools.product([0, 1], repeat=n))
9
10
11 def conv_int_to_binary_list(out_f,num_bits):
12
13     return [int(bit) for bit in bin(out_f)[2:].zfill(num_bits)]
14
15
16 def is_linearly_separable(data, labels):
17     model = tf.keras.Sequential([
18         tf.keras.layers.Dense(1, activation='sigmoid', input_shape=(data.shape[1],))
19     ])
20
21     learning_rate_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
22         initial_learning_rate=0.5,
23         decay_steps=5,
24         decay_rate=0.7
25     )
26
27     model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=learning_rate_schedule),
28                 loss='binary_crossentropy', metrics=['accuracy'])
29     model.fit(data, labels, epochs=150, verbose=0) #verbose=2 for each epoch data showing
30
31     _, accuracy = model.evaluate(data, labels) #not storing loss
32
33     return accuracy == 1.0
34
35
36 def find_rel_with_n(num_variables,cnt):
37     slope, intercept, r_value, p_value, std_err = linregress(num_variables, cnt)
38     print(f'Relation is: y = {slope:.2f}x + {intercept:.2f}')
39
40 # Main program
```

The right pane shows the console output, which includes a table of results for 10 different input configurations (1/1 to 1/10). The results show that the model is linearly separable for all inputs, with accuracy 1.0000. The console also displays a warning message about TensorFlow's function retracing.

Input	Time	Loss	Accuracy
1/1	0s 171ms/step	0.1786	1.0000
1/1	0s 123ms/step	0.1013	1.0000
1/1	0s 121ms/step	0.3733	0.5750
1/1	0s 123ms/step	0.0637	1.0000
1/1	0s 136ms/step	0.1444	1.0000
1/1	0s 126ms/step	0.3523	0.6250
1/1	0s 137ms/step	0.1194	1.0000
1/1	0s 141ms/step	0.0693	1.0000
1/1	0s 159ms/step	0.1695	1.0000
1/1	0s 116ms/step	0.1064	1.0000
1/1	0s 131ms/step	0.2468	0.5750
1/1	0s 160ms/step	0.0051	1.0000

No of linearly separable function list from 2 to 6: [14, 103]
Relation is: y = 89.00x + -164.00

This version of python seems to be incorrectly compiled (internal generated filenames are not absolute). This may make the debugger miss breakpoints. Related bug: <http://bugs.python.org/issue1666007>
2023-08-17 19:24:51.268353: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: SSE SSE2 SSE3 SSE4.1 SSE4.2 AVX AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.

In [2]:

Logs:

Checking binary functions with 2 variables:

1/1 [=====] - 0s 185ms/step - loss: 0.0066 - accuracy: 1.0000

1/1 [=====] - 0s 205ms/step - loss: 0.1262 - accuracy: 1.0000

1/1 [=====] - 0s 122ms/step - loss: 0.0474 - accuracy: 1.0000

1/1 [=====] - 0s 194ms/step - loss: 0.0576 - accuracy: 1.0000

WARNING:tensorflow:5 out of the last 5 calls to <function

Model.make_test_function.<locals>.test_function at 0x0000022A95191E40> triggered tf.function

retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating

@tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects

instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has

reduce_retracing=True option that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.

1/1 [=====] - 0s 136ms/step - loss: 0.0937 - accuracy: 1.0000

WARNING:tensorflow:6 out of the last 6 calls to <function Model.make_test_function.<locals>.test_function at 0x0000022A962A9260> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has reduce_retracing=True option that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.

1/1 [=====] - 0s 137ms/step - loss: 0.0755 - accuracy: 1.0000

1/1 [=====] - 0s 126ms/step - loss: 0.6932 - accuracy: 0.2500

1/1 [=====] - 0s 179ms/step - loss: 0.0591 - accuracy: 1.0000

1/1 [=====] - 0s 148ms/step - loss: 0.1234 - accuracy: 1.0000

1/1 [=====] - 0s 120ms/step - loss: 0.6932 - accuracy: 0.5000

1/1 [=====] - 0s 137ms/step - loss: 0.0595 - accuracy: 1.0000

1/1 [=====] - 0s 120ms/step - loss: 0.0723 - accuracy: 1.0000

1/1 [=====] - 0s 147ms/step - loss: 0.0582 - accuracy: 1.0000

1/1 [=====] - 0s 171ms/step - loss: 0.1786 - accuracy: 1.0000

1/1 [=====] - 0s 123ms/step - loss: 0.1013 - accuracy: 1.0000

1/1 [=====] - 0s 121ms/step - loss: 0.3733 - accuracy: 0.8750

1/1 [=====] - 0s 123ms/step - loss: 0.0637 - accuracy: 1.0000

1/1 [=====] - 0s 136ms/step - loss: 0.1444 - accuracy: 1.0000

1/1 [=====] - 0s 126ms/step - loss: 0.3523 - accuracy: 0.6250

1/1 [=====] - 0s 137ms/step - loss: 0.1194 - accuracy: 1.0000

1/1 [=====] - 0s 141ms/step - loss: 0.0693 - accuracy: 1.0000

1/1 [=====] - 0s 159ms/step - loss: 0.1695 - accuracy: 1.0000

1/1 [=====] - 0s 116ms/step - loss: 0.1064 - accuracy: 1.0000

1/1 [=====] - 0s 131ms/step - loss: 0.2468 - accuracy: 0.8750

1/1 [=====] - 0s 160ms/step - loss: 0.0051 - accuracy: 1.0000

No of linearly separable function list from 2 to 6: [14, 103]

Relation is: $y = 89.00x + -164.00$

OUTPUT BY CODE : [14,103]