

UNIVERSITY COLLEGE LONDON
DEPARTMENT OF SPACE AND CLIMATE PHYSICS

Candidate Code: HYXC3

Programme Title: MSc Scientific Computing

Module Code: SPCE0038

Module Title: Machine Learning with Big Data

End Assessment

In submitting this coursework, I assert that the work presented is entirely my own except where properly marked and cited.

Date of Submission:	dd/mm/yy
---------------------	----------

Question 1

1(a)

TODO

1(b)

TODO

1(c)

TODO

1(d)

TODO

1(e)

TODO

1(f)

TODO

1(g)

TODO

Question 2

2(a)

TODO

2(b)

TODO

2(c)

TODO

2(d)

TODO

2(e)

TODO

2(f)

TODO

2(g)

TODO

2(h)

TODO

Question 3

3(a)

TODO

3(b)

TODO

3(c)

TODO

3(d)

TODO

3(e)

TODO

3(f)

TODO

Question 4

4(a)

TODO

4(b)

TODO

4(c)

TODO

4(d)

TODO

4(e)

TODO

4(f)

TODO

```
# Fetch batch function:
```

```
def fetch_batch(epoch, batch_index, batch_size):  
    return X_batch, y_batch
```

```
# Set up computational graph:
```

```
import tensorflow as tf  
reset_graph ()
```

```
n_epochs = 1000  
learning_rate = 0.01
```

```
X = tf.constant(scaled_housing_data_plus_bias, dtype=tf.float32, name="X")  
y = tf.constant(housing_data_target, dtype=tf.float32, name="y")
```

```
theta = tf.Variable(tf.random_uniform([n + 1, 1], -1.0, 1.0), name="theta")  
y_pred = tf.matmul(X, theta, name="predictions")  
error = y_pred - y  
mse = tf.reduce_mean(tf.square(error), name="mse")  
optimizer = tf.train.GradientDescentOptimizer(learning_rate)  
training_op = optimizer.minimize(mse)
```

```
# Execute:
```

```
init = tf.global_variables_initializer()
```

```
with  
tf.Session() as sess:  
    sess.run(init)  
    for epoch in range(n_epochs):  
        if epoch % 100 == 0:  
            print("Epoch", epoch, "MSE=", mse.eval()) sess.run(training_op)  
    best_theta = theta.eval()
```

```

1  # Fetch batch function:
2
3  def fetch_batch(epoch, batch_index, batch_size):
4      return X_batch, y_batch
5
6
7  # Set up computational graph:
8
9  import tensorflow as tf
10 reset_graph ()
11
12 n_epochs = 1000
13 learning_rate = 0.01
14
15 X = tf.constant(scaled_housing_data_plus_bias, dtype=tf.float32, name="X")
16 y = tf.constant(housing_data_target, dtype=tf.float32, name="y")
17
18 theta = tf.Variable(tf.random_uniform([n + 1, 1], -1.0, 1.0), name="theta")
19 y_pred = tf.matmul(X, theta, name="predictions")
20 error = y_pred - y
21 mse = tf.reduce_mean(tf.square(error), name="mse")
22 optimizer = tf.train.GradientDescentOptimizer(learning_rate)
23 training_op = optimizer.minimize(mse)
24
25
26 # Execute:
27
28 init = tf.global_variables_initializer()
29
30 with
31 tf.Session() as sess:
32     sess.run(init)
33     for epoch in range(n_epochs):
34         if epoch % 100 == 0:
35             print("Epoch", epoch, "MSE=", mse.eval()) sess.run(training_op)
36     best_theta = theta.eval()

```

Listing 1: Question 4f

Question 5

5(a)

TODO

5(b)

TODO

5(c)

TODO

5(d)

TODO

5(e)

TODO

5(f)

TODO