from .hashing\_utils import make\_feature\_id, numpy\_hashing\_uniform

import numpy as np

import tensorflow.compat.v1 as tf

import twml

class TFModelWeightsInitializerBuilder(object):

def \_\_init\_\_(self, num\_bits):

self.num\_bits = num\_bits

def build(self, tf\_model\_initializer):

'''

:return: (bias\_initializer, weight\_initializer)

'''

initial\_weights = np.zeros((2 \*\* self.num\_bits, 1))

features = tf\_model\_initializer["features"]

self.\_set\_binary\_feature\_weights(initial\_weights, features["binary"])

self.\_set\_discretized\_feature\_weights(initial\_weights, features["discretized"])

return tf.constant\_initializer(features["bias"]), twml.contrib.initializers.PartitionConstant(initial\_weights)

def \_set\_binary\_feature\_weights(self, initial\_weights, binary\_features):

for feature\_name, weight in binary\_features.items():

feature\_id = make\_feature\_id(feature\_name, self.num\_bits)

initial\_weights[feature\_id][0] = weight

def \_set\_discretized\_feature\_weights(self, initial\_weights, discretized\_features):

for feature\_name, discretized\_feature in discretized\_features.items():

feature\_id = make\_feature\_id(feature\_name, self.num\_bits)

for bin\_idx, weight in enumerate(discretized\_feature["weights"]):

final\_bucket\_id = numpy\_hashing\_uniform(feature\_id, bin\_idx, self.num\_bits)

initial\_weights[final\_bucket\_id][0] = weight