from .parsers import LollyModelFeaturesParser

class TFModelInitializerBuilder:

def \_\_init\_\_(self, model\_features\_parser=LollyModelFeaturesParser()):

self.\_model\_features\_parser = model\_features\_parser

def build(self, lolly\_model\_reader):

'''

:param lolly\_model\_reader: LollyModelReader instance

:return: tf\_model\_initializer dictionary of the following format:

{

"features": {

"bias": 0.0,

"binary": {

# (feature name : feature weight) pairs

"feature\_name\_1": 0.0,

...

"feature\_nameN": 0.0

},

"discretized": {

# (feature name : index aligned lists of bin\_boundaries and weights

"feature\_name\_1": {

"bin\_boundaries": [1, ..., inf],

"weights": [0.0, ..., 0.0]

}

...

"feature\_name\_K": {

"bin\_boundaries": [1, ..., inf],

"weights": [0.0, ..., 0.0]

}

}

}

}

'''

tf\_model\_initializer = {

"features": {}

}

features = self.\_model\_features\_parser.parse(lolly\_model\_reader)

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

self.\_set\_discretized\_features(features["discretized"], tf\_model\_initializer)

self.\_dedup\_binary\_features(features["binary"], features["discretized"])

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

return tf\_model\_initializer

def \_set\_discretized\_features(self, discretized\_features, tf\_model\_initializer):

if len(discretized\_features) == 0:

return

num\_bins = max([len(bins) for bins in discretized\_features.values()])

bin\_boundaries\_and\_weights = {}

for feature\_name in discretized\_features:

bin\_boundaries\_and\_weights[feature\_name] = self.\_extract\_bin\_boundaries\_and\_weights(

discretized\_features[feature\_name], num\_bins)

tf\_model\_initializer["features"]["discretized"] = bin\_boundaries\_and\_weights

def \_dedup\_binary\_features(self, binary\_features, discretized\_features):

[binary\_features.pop(feature\_name) for feature\_name in discretized\_features]

def \_extract\_bin\_boundaries\_and\_weights(self, discretized\_feature\_buckets, num\_bins):

bin\_boundary\_weight\_pairs = []

for bucket in discretized\_feature\_buckets:

bin\_boundary\_weight\_pairs.append([bucket[0], bucket[2]])

# The default DBv2 HashingDiscretizer bin membership interval is (a, b]

#

# The Earlybird Lolly prediction engine discretizer bin membership interval is [a, b)

#

# Thus, convert (a, b] to [a, b) by inverting the bin boundaries.

for bin\_boundary\_weight\_pair in bin\_boundary\_weight\_pairs:

if bin\_boundary\_weight\_pair[0] < float("inf"):

bin\_boundary\_weight\_pair[0] \*= -1

while len(bin\_boundary\_weight\_pairs) < num\_bins:

bin\_boundary\_weight\_pairs.append([float("inf"), float(0)])

bin\_boundary\_weight\_pairs.sort(key=lambda bin\_boundary\_weight\_pair: bin\_boundary\_weight\_pair[0])

bin\_boundaries, weights = list(zip(\*bin\_boundary\_weight\_pairs))

return {

"bin\_boundaries": bin\_boundaries,

"weights": weights

}