"""

Feature configuration for DeepBird jobs returns dictionary of sparse and dense Features

"""

from twitter.deepbird.io.legacy.contrib import feature\_config

import twml

class FeatureConfig(feature\_config.FeatureConfig):

def get\_feature\_spec(self):

"""

Generates a serialization-friendly dict representing this FeatureConfig.

"""

doc = super(FeatureConfig, self).get\_feature\_spec()

# Override the class in the spec.

doc["class"] = "twml.contrib.FeatureConfig"

return doc

class FeatureConfigBuilder(feature\_config.FeatureConfigBuilder):

# Overwrite self.build() to return twml.FeatureConfig instead

def build(self):

"""

Returns an instance of FeatureConfig with the features passed to the FeatureConfigBuilder.

"""

(

keep\_tensors,

keep\_sparse\_tensors,

feature\_map,

features\_add,

feature\_name\_to\_feature\_parser,

feature\_in\_bq\_name,

) = self.\_build()

discretize\_dict = {}

for config in self.\_sparse\_extraction\_configs:

if config.discretize\_num\_bins and config.discretize\_output\_size\_bits:

if config.discretize\_type == "percentile":

calibrator = twml.contrib.calibrators.PercentileDiscretizerCalibrator

elif config.discretize\_type == "hashed\_percentile":

calibrator = twml.contrib.calibrators.HashedPercentileDiscretizerCalibrator

elif config.discretize\_type == "hashing":

calibrator = twml.contrib.calibrators.HashingDiscretizerCalibrator

else:

raise ValueError("Unsupported discretizer type: " + config.discretize\_type)

discretize\_dict[config.output\_name] = calibrator(

config.discretize\_num\_bins,

config.discretize\_output\_size\_bits,

allow\_empty\_calibration=config.allow\_empty\_calibration,

)

elif config.discretize\_num\_bins or config.discretize\_output\_size\_bits:

raise ValueError(

"Discretize\_num\_bins AND discretize\_output\_size\_bits need to be in the FeatureConfig"

)

return FeatureConfig(

features={},

labels=self.\_labels,

weight=self.\_weight,

filters=self.\_filter\_features,

tensor\_types=keep\_tensors,

sparse\_tensor\_types=keep\_sparse\_tensors,

feature\_types=feature\_map,

sparse\_extraction\_configs=self.\_sparse\_extraction\_configs,

feature\_extraction\_configs=self.\_feature\_extraction\_configs,

feature\_group\_extraction\_configs=self.\_feature\_group\_extraction\_configs,

image\_configs=self.\_image\_configs,

discretize\_config=discretize\_dict,

feature\_ids=features\_add,

decode\_mode=self.\_decode\_mode,

legacy\_sparse=self.\_legacy\_sparse,

feature\_name\_to\_feature\_parser=feature\_name\_to\_feature\_parser,

feature\_in\_bq\_name=feature\_in\_bq\_name,

)

TensorExtractionConfig = feature\_config.TensorExtractionConfig

FeatureGroupExtractionConfig = feature\_config.FeatureGroupExtractionConfig

ImageExtractionConfig = feature\_config.ImageExtractionConfig

\_set\_tensor\_namedtuple = feature\_config.\_set\_tensor\_namedtuple