class LollyModelScorer(object):

def \_\_init\_\_(self, data\_example\_parser):

self.\_data\_example\_parser = data\_example\_parser

def score(self, data\_example):

value\_by\_feature\_name = self.\_data\_example\_parser.parse(data\_example)

features = self.\_data\_example\_parser.features

return self.\_score(value\_by\_feature\_name, features)

def \_score(self, value\_by\_feature\_name, features):

score = features["bias"]

score += self.\_score\_binary\_features(features["binary"], value\_by\_feature\_name)

score += self.\_score\_discretized\_features(features["discretized"], value\_by\_feature\_name)

return score

def \_score\_binary\_features(self, binary\_features, value\_by\_feature\_name):

score = 0.0

for binary\_feature\_name, binary\_feature\_weight in binary\_features.items():

if binary\_feature\_name in value\_by\_feature\_name:

score += binary\_feature\_weight

return score

def \_score\_discretized\_features(self, discretized\_features, value\_by\_feature\_name):

score = 0.0

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

if discretized\_feature\_name in value\_by\_feature\_name:

feature\_value = value\_by\_feature\_name[discretized\_feature\_name]

score += self.\_find\_matching\_bucket\_weight(buckets, feature\_value)

return score

def \_find\_matching\_bucket\_weight(self, buckets, feature\_value):

for left\_side, right\_side, weight in buckets:

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

if feature\_value >= left\_side and feature\_value < right\_side:

return weight

raise LookupError("Couldn't find a matching bucket for the given feature value.")