Pseudo-code:

```
def grow(dataset, attributes, node):

   if majority_label_percent >= 0.9 or len(attributes) == 0):
      node.label = majority_label
      return node
   else:
      if isRoot():
         calculate entropy of root
         node.entropy = (-1)*((x*logx) + (y*logy))

      split_variables = getSplitVariables(dataset, attributes, node.entropy) // return
split attribute, number of categories (E_cat) or numerical bins (E_bin)

      node.attribute = split_variables['attribute']
      remove selected attribute

      if split_variables['type'] == 'categorical':
         partitions = split_variables['E_cat']
         for each category:
            create child node and set its attributes
            partitioned_data = filtered dataset with category value
            grow(partitioned_data, attributes, child_node)
      else:
         partitions = split_variables['E_bin']
         for each bin:
            create child node and set its attributes
            partitioned_data = dataset that fall in this bin
            grow(partitioned_data, attributes, child_node)

def getSplitVariables(dataset, attributes, parent_entropy):
   # find split attribute using information gain

   split_data = {}
   total_records = dataset.shape[0]
   num_cols = numerical columns
   cat_cols = categorical columns

   # calculate entropy of categorical attributes
   for each categorical attribute:
      categories = dataset[attr].unique()
      attr_data = {}
      E_attr = 0
      cat_data = {}

      # calculate entropy of each category
```

```
    for each category:
      income_gt_50k_cat = number of income >50K
      income_lte_50k_cat = number of income <=50K
      total_records_in_cat = income_gt_50k_cat + income_lte_50k_cat

      E_cat = calculate nentropy of category
      E_attr += (total_records_in_cat/total_records)*E_cat
      cat_data[cat] = E_cat

    attr_data['attribute'] = attr
    attr_data['info_gain'] = parent_entropy - E_attr
    attr_data['type'] = 'categorical'
    attr_data['E_cat'] = cat_data
    attr_data['E_bin'] = None
    split_data[attr] = attr_data

  # calculate entropy of numerical attributes
  for each numerical attribute:
    bins = pd.qcut(dataset[attr], q=4, duplicates='drop', retbins=True)[1]
    attr_data = {}
    E_attr = 0
    num_data = {}

    for each bin:
      income_gt_50k_bin = number of income >50K
      income_lte_50k_bin = number of income <=50K
      total_records_in_bin = income_gt_50k_bin + income_lte_50k_bin

      E_bin = calculate entropy of bin
      E_attr += (total_records_in_bin/total_records)*E_bin
      num_data[bin_range] = E_bin

    attr_data['attribute'] = attr
    attr_data['info_gain'] = parent_entropy - E_attr
    attr_data['type'] = 'numerical'
    attr_data['E_cat'] = None
    attr_data['E_bin'] = num_data
    split_data[attr] = attr_data

  max_info_attr = attribute with maximum information gain


  return max_info_attr
```

Average accuracy of 5-fold-cross-validation: 76%

Handling missing values:
Missing values where imputed using pandas library function df.replace() which imputes missing values based on surrounding data records.