```
function [success_rate success rejection_rate rejection fail_rate fail] =
generate_random_trials(num_trials, hidden_category, M, N, anchor_tree, delta_tree)

  success = 0;
  rejection = 0;

  for k = 1 : num_trials

    radius = 2*hidden_category + rand();
    t = 0:pi/11:2*pi;
    [A,B,C] = cylinder(radius + t.*cos(t));

    data_array = shape_to_data(A,B,C);

    X = [];
    Y = [];
    Z = [];

    num_items = size(data_array,2);


      for i = 1 : num_items

        temp_vec = data_array{i};

        x = temp_vec(1);
        y = temp_vec(2);
        z = temp_vec(3);

        X = [X x];
        Y = [Y y];
        Z = [Z z];

      endfor

  shape_vector = [X Y Z];
  new_data_item{1} = shape_vector;
  missing_data_vector = M + 1 : N;

  [category_index predicted_vector final_delta min_difference
predicted_vector_array] = predict_best_fit_tree_N(anchor_tree, delta_tree,
new_data_item, 1, 1, missing_data_vector, N);

  width = category_index(1);
  depth = category_index(2);

  %if true, then the data was placed in the tree
  if(final_delta != Inf)
```

```
    temp = anchor_tree{width,depth};
    predicted_category = temp(N+1);

    %if true, then the data was correctly classified
    if(predicted_category == hidden_category)

      success = success + 1;

    endif

  %otherwise, the data was rejected by the algorithm
  else

      rejection = rejection + 1;

  endif

endfor

success_rate = (success/num_trials)*100;
rejection_rate = (rejection/num_trials)*100;

%if data was not correctly classified or rejected, then it was incorrectly
classified
fail_rate = 100 - (success_rate + rejection_rate);
fail = num_trials - (success + rejection);

endfunction
```