3-dim Tabu Search + ML Algorithm to Solve 2E-CVRPTW with Fuzzy Demand

1. Calculate the expected damands using K-NN and Initialization

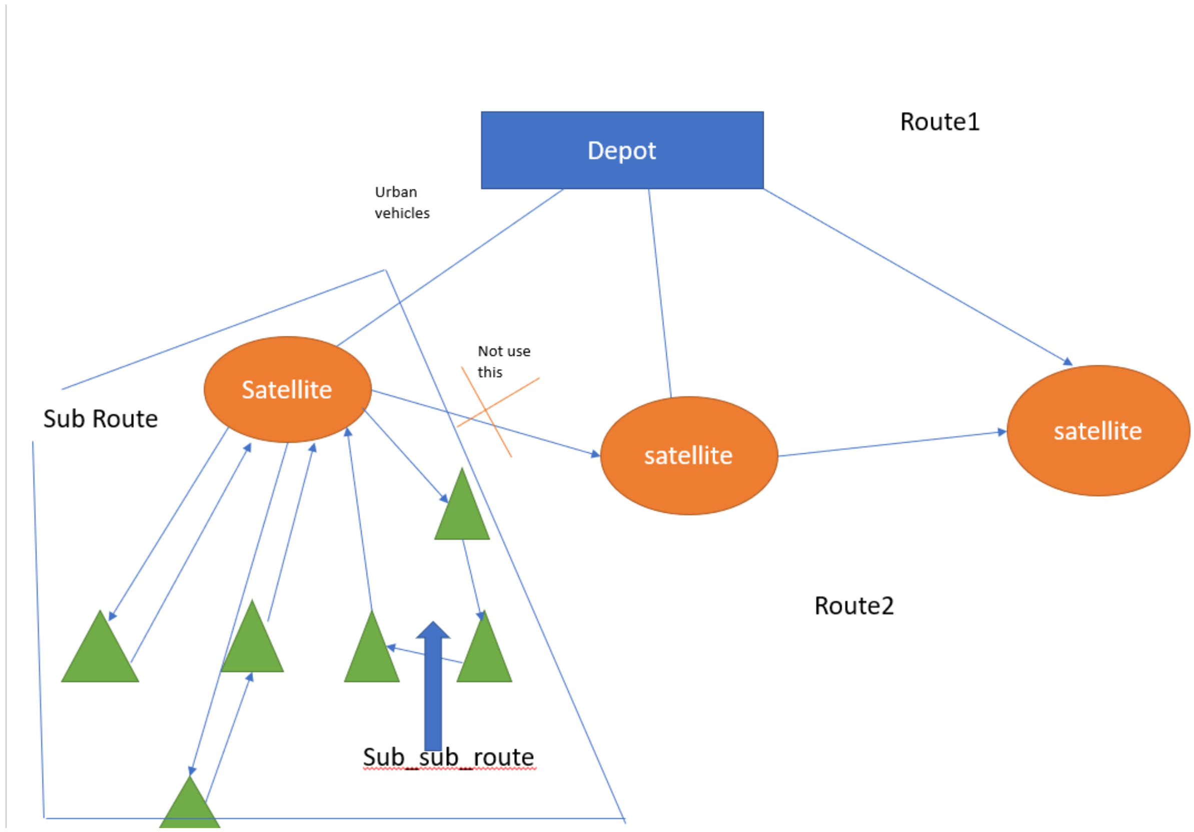
```
updated_demand <- mean_demand
Training data <- Coordinates of customers
Labeling data <- updated_demand
Training Using K-NN
expected_demand <- predicted value

define penalty function with time window
initial_solution <- [route1, route2] (each element of route1 and route2 is also list)
  for each customer c:
        find the suitable satellite s.
        route2[s].append(c)
  calculate required amount R of goods in each satellite.
   For each satellite s:
        N_require_urban <- int(R[s] / Q1) + 1
        route1 <- [-1] * N_require_urban
```

2. Implement the tabu search algorithm in sub route2

It means, first we optimize the each sub route of each satellite in route2.
the sameple of sub route of each satellite is [[0,1,4], [2,3], [6], [5,7]].
In the sample sub route, we used 4 city freighters and each element of sub route is the route of each frighters. (See the below figure)
According to the Bell Equation:

Min (route) = $min_{route2}(min_{sub_{route}}(min_{sub_{sub_{route}}}\ cost))$



- Implement Tabu search algorithm in sub_sub_route
- Implement Tabu search algorithm in sub_route

3. Optimize the Total Route using Tabu Genetic Algorithm

route1 = G(route2)

best_solution = [route1, route2]

best_cost = calculate_optimized_cost (route2) (Implement Tabu Search)

bestcandidate <- [route1, route2]

tabuList ← []

tabuList.push([route1, route2])

while (not stoppingCondition())

```
   sNeighborhood ← getNeighbors(bestCandidate)
   bestCandidate ← sNeighborhood[0]
   for (sCandidate in sNeighborhood)
      if ( (not tabuList.contains(sCandidate)) and
(calculate_optimized_cost(sCandidate) > calculate_optimized_cost(bestCandidate)) )
         bestCandidate ← sCandidate
      end
   end
   if (calculate_optimized_cost(bestCandidate) > best_cost)
      best_solution ← bestCandidate
   end
   tabuList.push(bestCandidate)
   if (tabuList.size > maxTabuSize)
      tabuList.removeFirst()
   end
end
return best_solution


if expected_demand < real_demand:
best_solution <- best_solution with max_demand

return best_solution
```