Assignment 2

Question 2.1

a)

a) Assume an object m' such that $m' \neq m$ and $TD(C,m') < TD(C,m)$

Since $TD(C,m) = \sum_{p \in C} dist(p,m)$ and $dist(p,m) = \sum_{i=1}^{d} \delta(p_i, m_i)$

$TD(C,m') < TD(C,m) = \sum_{p \in C} \sum_{i=1}^{d} \delta(p_i, m'_i) < \sum_{p \in C} \sum_{i=1}^{d} \delta(p_i, m_i)$

Since the function for both sides can only increase

there must exist one attribute i such that $\sum_{p \in C} \delta(p_i, m'_i) < \sum_{p \in C} \delta(p_i, m_i)$

Therefore by contradiction, $TD(C,m)$ minimize the cluster cost

b)

```
kmodes(dataset, numberOfClusters)
        cluster = random.sample(dataset, numberOfClusters)
        dataset = assign_cluster(dataset, cluster)
        While(True):
                cluster = generate_cluster(dataset, numberOfClusters)
                dataset = assign_cluster(dataset, cluster)
                if cluster remain unchanged:
                        break
        return dataset


assign_cluster(dataset, cluster)
        for row in dataset:
                for assign in cluster:
                        assigned_number = find_min_distance(assign, row)
                row.append(assigned_number)
```

```
        return dataset


generate_cluster(dataset, numberOfClusters)
        length = len(data[0])-1
        cluster_table = []

        for i in range(numberOfClusters):
        temp_cluster = []
        for row in data:
                if row[length] == i:
                        temp_cluster.append(row)
        column_table = []
        for i in range(length):
                columns = []
                for temp_r in  temp_cluster:
                        columns.append(temp_r[i])
                column_table.append(columns)
        One_cluster = most_frequent(column_table)
        cluster_table.append(One_cluster)
  return cluster_table
```

Question 2.2

A: The data contains 2480 missing entries in total, and all missing entries are from the same column. Thus, dropping out all rows with missing entries will lose an enormous amount of data so this method is unfeasible. Another method is to let the algorithm handles the impure data. However, what I did is to randomly assign the attribute by their frequency of occurrence. This method is better than letting the algorithm handles the data as a missing entry would be counted as a false 100% of the time, whereas randomly assign would have a better chance of getting it right.