**HOMEWORK 1**

**Question 1:**

a) To find the smallest set of points *S* such that every point in the interval [0,1] is within 0.01 of at least one point in *S*:

* As every point has to be within 0.01 of at least one point in *S*, the total cover distance of a point should be:

0.01 + 0.01 = 0.02

* We also have the total length of the interval [0,1] is: 1

Therefore, the number of point satisfies the requirement should be:

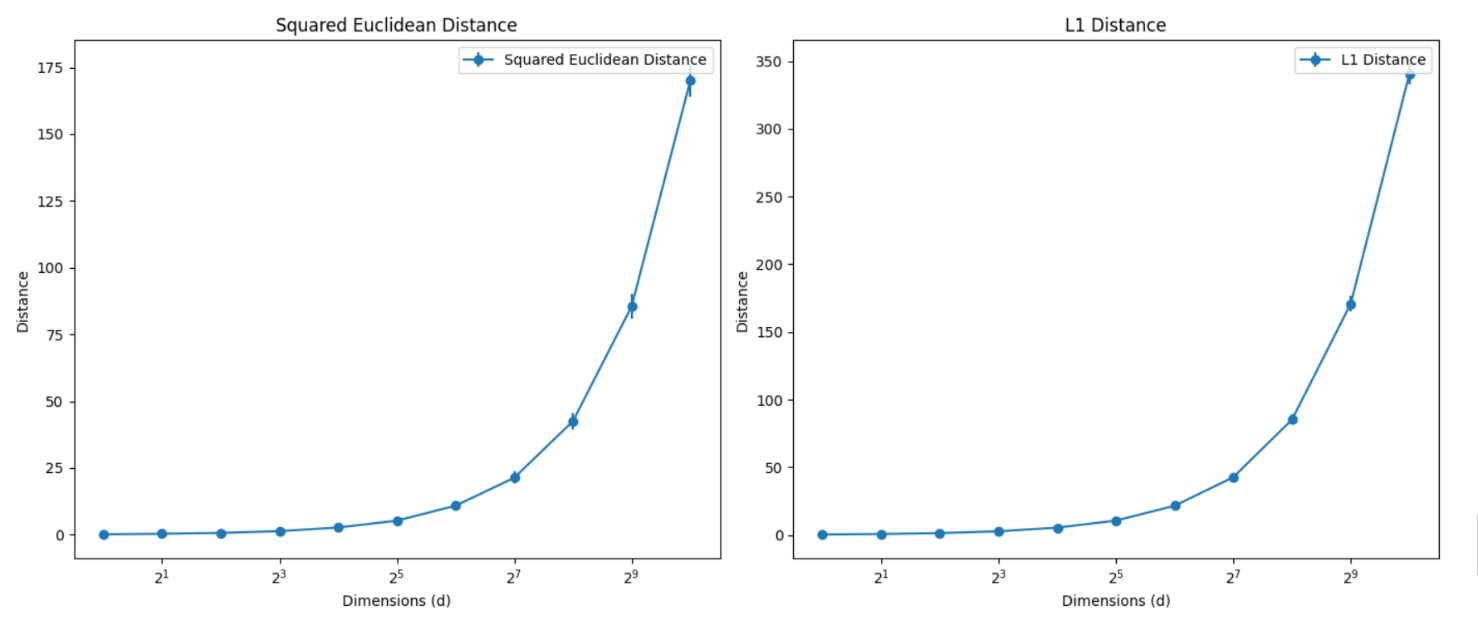
Number of points = ​= 50

b) When working with a classification problem that involves multiple features (e.g., 10 features), maintaining the guarantee that any new test point is within a specified distance of an existing point becomes significantly more challenging due to several factors:

* + Curse of Dimensionality: In a high-dimensional space, even a small amount of noise can lead to significant variations in distance metrics, making it harder to ensure that new points are close to existing ones.
  + Increased Complexity in Coverage: In a one-dimensional space, covering the interval with points is straightforward. However, in higher dimensions, the notion of "closeness" becomes more complex. For example, in a 10-dimensional space, a point might be close to some features but far from others, complicating the task of ensuring that every point is within a certain distance from a reference point.
  + Higher Computational Requirements: With more features, the computational resources required to analyze and classify data increase since the algorithms that work well in lower dimensions may become insufficient in the higher dimensions.
  + Visualization Challenges: Visualizing data in more than three dimensions is inherently difficult, which can hinder the understanding of the data distribution and the relationships among points.
  + Dimensionality Reduction Techniques: To manage high-dimensional data, techniques such as dimensionality reduction (e.g., PCA, t-SNE) may be employed. However, these techniques can sometimes distort the data structure, leading to potential inaccuracies in maintaining the desired proximity guarantees.

Overall, the combination of increased complexity, computational demands, and the challenges of high-dimensional spaces makes it much more difficult to ensure that every new test point is within a specified distance of existing points in a dataset with multiple features.

c) The code: [Question 1C](https://colab.research.google.com/drive/1zvy8SP-IIwedvwnbXavgMp3XUjbEI2Tc?usp=sharing)

The output figure:

d)

Using the linearity of expectation, we can find the expected value of *R*:

E[R] = *E*[*Z*1​+*Z*2​+⋯+*Zd*​] = *E*[*Z*1​]+*E*[*Z*2​]+⋯+*E*[*Zd*​]

Since the expected value *E*[*Zi*​] is the same for all *i*:

Using the independence of *Zi***​** and the property of variance, we can find the variance of *R*:

Since the variance Var[*Zi*​] is the same for all *i*:

e)

Let *R* be the squared Euclidean distance between two randomly sampled points from a unit cube in *d* dimensions. We want to define the event *E* as:

This event states that the squared distance *R* deviates from its expected value  *E*[*R*] by at least *k*.

According to Markov's inequality, we can bound the probability of event *E*:

From part d, we have E[R] = d/6 and Var[R] = 7d/180.

Therefore, we have:

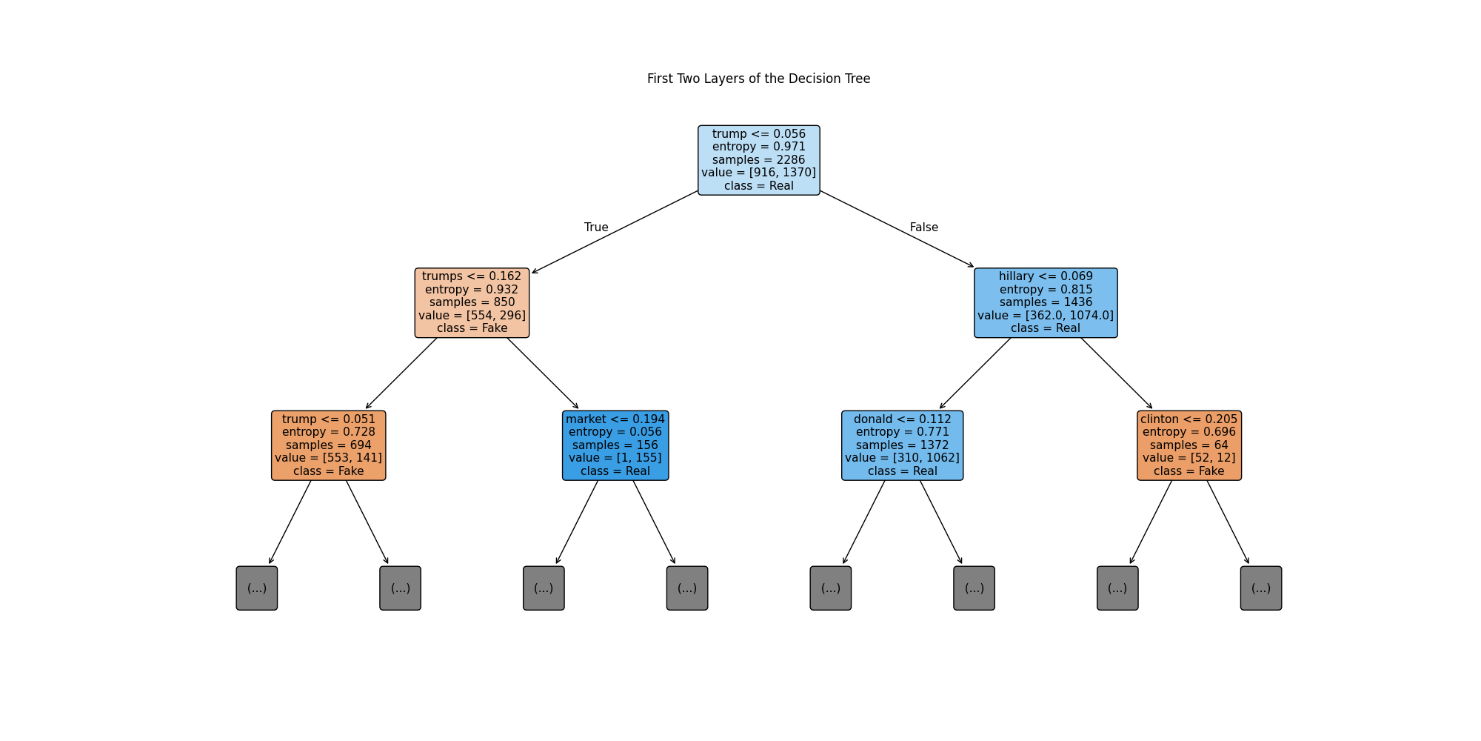
We also have *k* = *cd*,

Therefore:

As *d* approaches infinity, the bound for  *P*(*E*) becomes:

This result indicates that the probability that the squared distance *R* deviates from its expected value *E*[*R*] by at least *k* (where *k* is proportional to *d*) approaches zero as the dimension increases.

**Question 2**

c) The first two layers of the tree:  


**Question 3**

a)

- The gradient descent update rule for *wj*​ is given by:

where *n* is the learning rate.

- The gradient of the squared loss term is:

This term represents the contribution of the prediction error to the gradient.

- The L1 penalty is given by:

The gradient of is defined:

- The L2 penalty is given by:

The gradient is:

Now we can combine these gradients to get the update rules for :

* If ​> 0:
* If ​= 0:
* If ​< 0:

Bias update:

The bias *b* is updated using the standard gradient descent rule:

b)

Since we apply only the L2 penalty, the cost function becomes:

Taking the derivative of with respect to :

Setting this derivative equal to zero gives us the linear system:

Where

Here, ​ is the Kronecker delta, which is 1 if  and 0 otherwise.

c)

Let XXX be the design matrix with one row per training example, and let t\mathbf{t}t be the vector of target values. The cost function can be written as:

Where is a diagonal matrix with entries

To derive a closed-form solution for the parameter w*w*, we can express the system of equations in matrix form:

Where

* + - Matrix A:

* + - Vector c:

The closed-form solution for *w* is obtained by solving the normal equations:

Thus, the solution is: