Let us discuss the performance of a benchmark algorithm.

The Random Forest algorithm is known for its attractive property of detecting variable interactions and excellent performance as a learning algorithm.

For the decision, we're selecting the Random Forest algorithm as a benchmark-- initially, we randomly partitioned the full data set into two separate parts, where the split was 50-50, and the partitioning was done evenly within each cost bin.

The first part, the training set, was used to develop the method.

The second part, the test set, was used to evaluate the model's performance.

The table in this slide reports the accuracy of the Random Forest algorithm on each of the three buckets.

Let us now introduce the idea of clustering.

Patients in each bucket may have different characteristics.

For this reason, we create clusters for each cost bucket and make predictions for each cluster using the Random Forest algorithm.

Clustering is mostly used in the absence of a target variable to search for relationships among input variables or to organize data into meaningful groups.

In this study, although the target variable is well-defined as a heart attack or not a heart attack, there are many different trajectories that are associated with the target.

There's not one set pattern of health or diagnostic combination that leads a person to heart attack.

Instead, we'll show that there are many different dynamic health patterns and time series diagnostic relations preceding a heart attack.

The clustering methods were used were spectral clustering and k-means clustering.

We focus, in the lecture, on the k-means clustering.

The broad description of the algorithm is as follows.

We first specify the number of clusters k.

Then we randomly assign each data point to a cluster.

We then compute the cluster centroids.

We re-assign each point to the closest cluster centroid.

We then re-compute the cluster centroids, and we repeat steps 4 and 5 until no improvement is made.

Let us illustrate the k-means algorithm in action.

We specify the desired number of clusters k.

In this case, we use k=2.

We then randomly assign each data point to a cluster.

In this case, we have the three points in red, and the two points in black.

We then compute the cluster centroids, indicated by the red x and the grey x.

We re-assign each point to the closest cluster centroid, and now you observe that this point changes from a red to a grey.

We re-compute the cluster centroids, and we repeat the previous steps, 4 and 5 until no improvement is made.

We observe that, in this case, the k-means clustering is done, and this is our final clustering.

Let us discuss some practical considerations.

The number of clusters k can be selected from previous knowledge or by simply experimenting.

We can strategically select initial partition of points into clusters if we have some knowledge of the data.

We can also run the algorithm several times with different random starting points.

In the recitations, we'll learn how to run the k-means algorithm in R.

So how do we measure performance?

After we construct the clusters in the training set, we assign new observations to clusters by proximity to the centroid of each cluster.

We measure performance by recording the average performance rate in each cluster.

Let us now discuss the performance of the clustering methods.

We perform clustering on each bucket using k=10 clusters.

In the table we record the average prediction rate of each cost bucket.

We observe a very visible improvement when we use clustering-- from 49% to 64%, from 56% to 73%, from 58% to 78%.