



CREDIT TASK 10

Malidi Wageesha Jinadasa

104508896

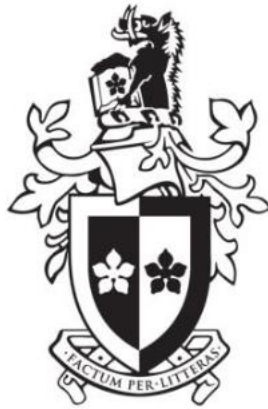
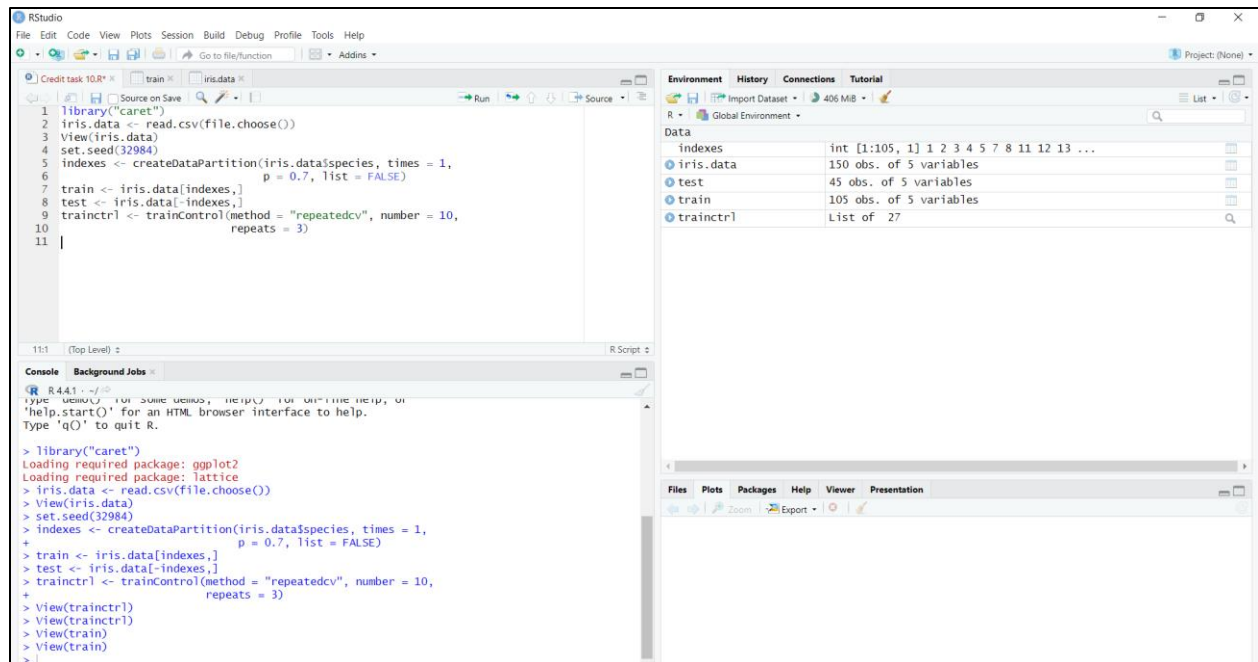


Table of Contents

Code for loading data	3
Viewing Iris data after loading	3
Split Train Data	4
Split Test Data	4
Random Forest Classification	5
Random Forest Predicted Test Results	5
Code for Random Forest Classification.....	5
Accuracy of Randon Forest using confusion Matrix.....	6
Clustering Using K-Means	7
Code For K-Means Clustering	7
Results of K-means Clustering.....	7
Question 1. What variable should you enter for each of the parameters? Why?	8
Question 2. For clustering, should we use the train, test or complete dataset? Why?	8
Code for Viewing components	8
Viewing Centers component	8
Viewing Cluster Assignments	8
Viewing total number of data points and the cluster sum of squares	9
Code for Comparing results	9
The Combined data frame with new columns.....	10
Question 3 When you compare the cluster assignment of k-means with the species prediction of ransom forest, do these agree? Can you tell which species it is? Discuss in a few sentences.....	10
Code for plotting actual Species	10
Code for plotting Random Forest predictions	11
Code for plotting k-means assignments	12
Question 4: can you see why k-means () assigned the clusters as it did? State your observations.....	12

Code for loading data



Viewing Iris data after loading

	sepal_length	sepal_width	petal_length	petal_width	species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa
11	5.4	3.7	1.5	0.2	setosa
12	4.8	3.4	1.6	0.2	setosa
13	4.8	3.0	1.4	0.1	setosa
14	4.3	3.0	1.1	0.1	setosa

Split Train Data

```
> print(train)
```

	sepal_length	sepal_width	petal_length	petal_width	species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa
11	5.4	3.7	1.5	0.2	setosa
12	4.8	3.4	1.6	0.2	setosa
13	4.8	3.0	1.4	0.1	setosa
14	4.3	3.0	1.1	0.1	setosa
15	5.8	4.0	1.2	0.2	setosa
17	5.4	3.9	1.3	0.4	setosa
18	5.1	3.5	1.4	0.3	setosa
19	5.7	3.8	1.7	0.3	setosa
20	5.1	3.8	1.5	0.3	setosa

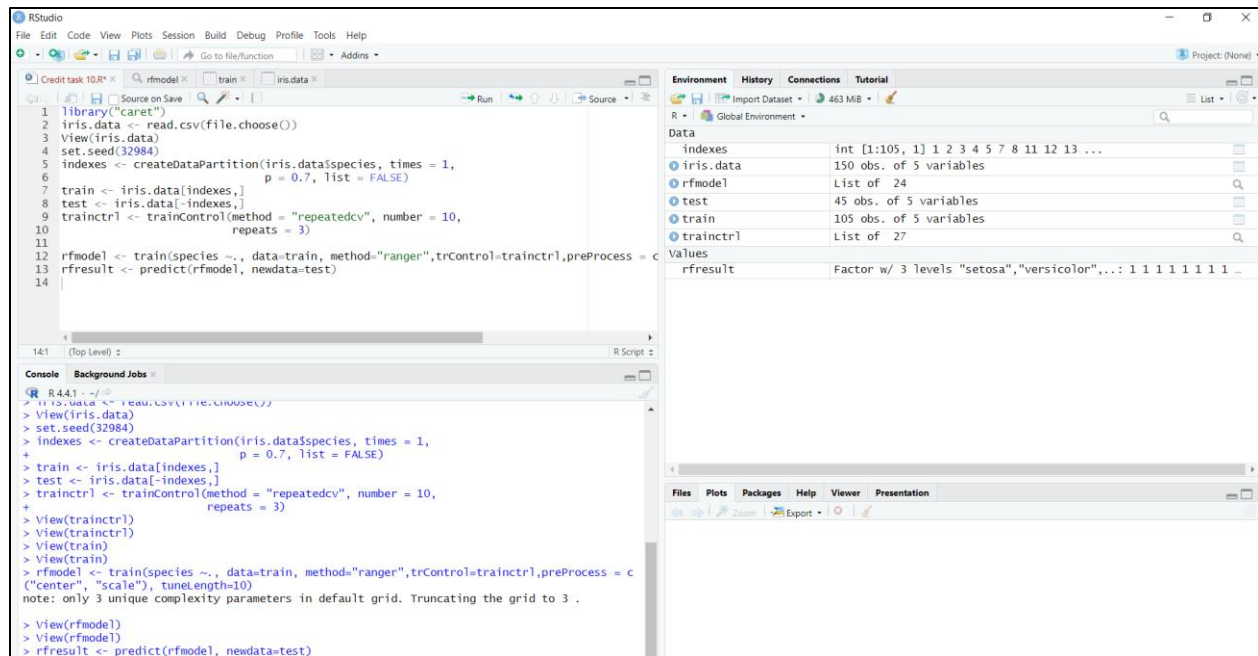
Split Test Data

```
[1] "-----test data-----"
```

```
> print(test)
```

	sepal_length	sepal_width	petal_length	petal_width	species
6	5.4	3.9	1.7	0.4	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa
16	5.7	4.4	1.5	0.4	setosa
21	5.4	3.4	1.7	0.2	setosa
26	5.0	3.0	1.6	0.2	setosa
30	4.7	3.2	1.6	0.2	setosa
38	4.9	3.1	1.5	0.1	setosa
39	4.4	3.0	1.3	0.2	setosa
42	4.5	2.3	1.3	0.3	setosa
44	5.0	3.5	1.6	0.6	setosa
45	5.1	3.8	1.9	0.4	setosa
48	4.6	3.2	1.4	0.2	setosa
49	5.3	3.7	1.5	0.2	setosa
50	5.0	3.3	1.4	0.2	setosa
52	6.4	3.2	4.5	1.5	versicolor

Random Forest Classification



The screenshot shows the RStudio interface. The script editor contains the following code:

```
1 library("caret")
2 iris.data <- read.csv(file.choose())
3 View(iris.data)
4 set.seed(32984)
5 indexes <- createDataPartition(iris.data$species, times = 1,
6                               p = 0.7, list = FALSE)
7 train <- iris.data[indexes,]
8 test <- iris.data[-indexes,]
9 trainctrl <- trainControl(method = "repeatedcv", number = 10,
10                          repeats = 3)
11 rfmodel <- train(species ~., data=train, method="ranger", trControl=trainctrl, preProcess = c
12 rfresult <- predict(rfmodel, newdata=test)
14
```

The Environment pane on the right shows the following objects:

- `indexes`: int [1:105, 1] 1 2 3 4 5 7 8 11 12 13 ...
- `iris.data`: 150 obs. of 5 variables
- `rfmodel`: List of 24
- `test`: 45 obs. of 5 variables
- `train`: 105 obs. of 5 variables
- `trainctrl`: List of 27
- `rfresult`: Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 1 1 ...

The Console shows the output of the code, including the message: "note: only 3 unique complexity parameters in default grid. Truncating the grid to 3."

Random Forest Predicted Test Results

```
> print(rfresult)
 [1] setosa setosa setosa setosa setosa setosa setosa setosa
 [9] setosa setosa setosa setosa setosa setosa setosa versicolor
[17] versicolor versicolor versicolor versicolor virginica versicolor versicolor versicolor
[25] versicolor versicolor versicolor versicolor versicolor versicolor virginica virginica
[33] versicolor virginica virginica virginica virginica virginica virginica virginica
[41] virginica virginica virginica virginica virginica
Levels: setosa versicolor virginica
> |
```

Code for Random Forest Classification

```
# Add the Random Forest predictions to the test set
test$RandomForestPrediction <- rfresult

# Confusion matrix to evaluate Random Forest accuracy
print(confusionMatrix(table(test$RandomForestPrediction, test$species)))
```

Accuracy of Random Forest using confusion Matrix

```
> test$RandomForestPrediction <- rfresult  
> print(confusionMatrix(table(test$RandomForestPrediction, test$species)))  
Confusion Matrix and Statistics
```

	setosa	versicolor	virginica
setosa	15	0	0
versicolor	0	14	1
virginica	0	1	14

Overall Statistics

Accuracy : 0.9556
95% CI : (0.8485, 0.9946)
No Information Rate : 0.3333
P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.9333

Mcnemar's Test P-Value : NA

Statistics by Class:

	Class: setosa	Class: versicolor	Class: virginica
Sensitivity	1.0000	0.9333	0.9333
Specificity	1.0000	0.9667	0.9667
Pos Pred Value	1.0000	0.9333	0.9333
Neg Pred Value	1.0000	0.9667	0.9667
Prevalence	0.3333	0.3333	0.3333
Detection Rate	0.3333	0.3111	0.3111
Detection Prevalence	0.3333	0.3333	0.3333
Balanced Accuracy	1.0000	0.9500	0.9500

[illegible]

As parameters we include the features of the cluster. This provides the necessary details to group similar data points. We remove species column as k-means is unsupervised learning. And include the number of centers based on the number of different species. Since there are 3 distinct clusters *Sentosa*, *Versicolor*, *Virginica* we include 3 centers.

We should use the complete dataset because clustering is an unsupervised learning technique. So, it does not depend on any labels. It should analyze the whole dataset to detect patterns. Providing the full dataset will give more accurate grouping of data points.

```
# You can view specific components of the kmeansresult
print(kmeansresult$centers) # The cluster centers
print(kmeansresult$cluster) # The cluster assignments for each data point

# Add the cluster assignments to the dataset for visualization
iris.data$kmeans_cluster <- as.factor(kmeansresult$cluster) # Create a new column for the

# View each component size and sum of squares
print(kmeansresult$size) # Number of data points in each cluster
print(kmeansresult$tot.withinss) # Total within-cluster sum of squares
```

```
> print(kmeansresult$centers) # The cluster centers
  sepal_length sepal_width petal_length petal_width
1    5.901613    2.748387    4.393548    1.433871
2    6.850000    3.073684    5.742105    2.071053
3    5.006000    3.418000    1.464000    0.244000
> |
```

[illegible]

Viewing total number of data points and the cluster sum of squares

```
> iris.data$kmeans_cluster <- as.factor(kmeansresult$cluster) #  
> print(kmeansresult$size)  
[1] 62 38 50  
> print(kmeansresult$tot.withinss) # Total within-  
[1] 78.94084  
> |
```

Code for Comparing results

```
iris.data$kmeans_cluster <- as.factor(kmeansresult$cluster)  
  
# Subset the test dataset for comparison (same rows for RF and kmeans)  
  
# Assigning the kmeans clusters to test set  
test$kmeans_cluster <- iris.data$kmeans_cluster[-indexes]  
  
# Combine Random Forest predictions and kmeans clusters into a single dataframe  
df <- data.frame(  
  ActualSpecies = test$species,  
  RandomForestPrediction = rfresult,  
  KmeansCluster = test$kmeans_cluster  
)  
  
# Showing the combined dataframe  
print(df)
```

The Combined data frame with new columns

```
> print(df)
```

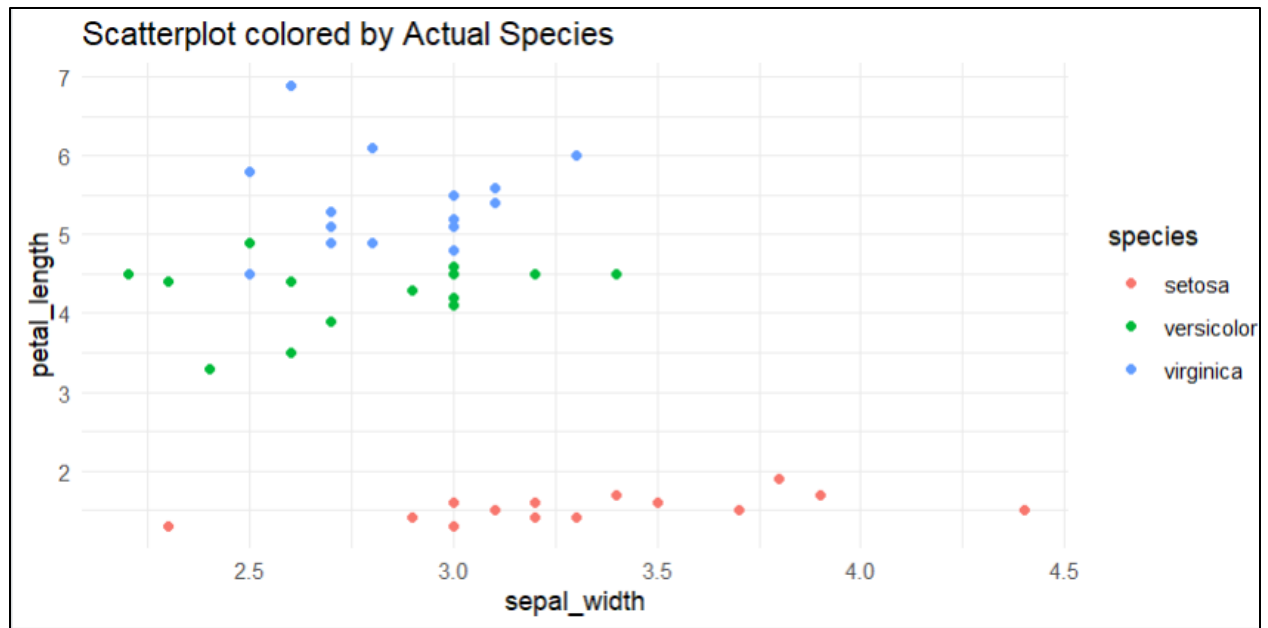
	ActualSpecies	RandomForestPrediction	KmeansCluster
1	setosa	setosa	3
2	setosa	setosa	3
3	setosa	setosa	3
4	setosa	setosa	3
5	setosa	setosa	3
6	setosa	setosa	3
7	setosa	setosa	3
8	setosa	setosa	3
9	setosa	setosa	3
10	setosa	setosa	3
11	setosa	setosa	3
12	setosa	setosa	3
13	setosa	setosa	3
14	setosa	setosa	3
15	setosa	setosa	3
16	versicolor	versicolor	1
17	versicolor	versicolor	1
18	versicolor	versicolor	1
19	versicolor	versicolor	1
20	versicolor	versicolor	1
21	versicolor	virginica	1
22	versicolor	versicolor	1
23	versicolor	versicolor	1

Question 3 When you compare the cluster assignment of k-means with the species prediction of random forest, do these agree? Can you tell which species it is? Discuss in a few sentences.

The comparison between k-means clustering and random forest predictions shows that they do not fully agree. This is expected because k-means is an unsupervised method that assigns clusters based on feature similarity without considering the actual species labels. However, there might be some overlap in the clusters and species predictions. By reviewing the clustering results and comparing them with the species predictions, we can identify which species the clusters likely represent, though they may not align perfectly due to the different approaches of the two methods.

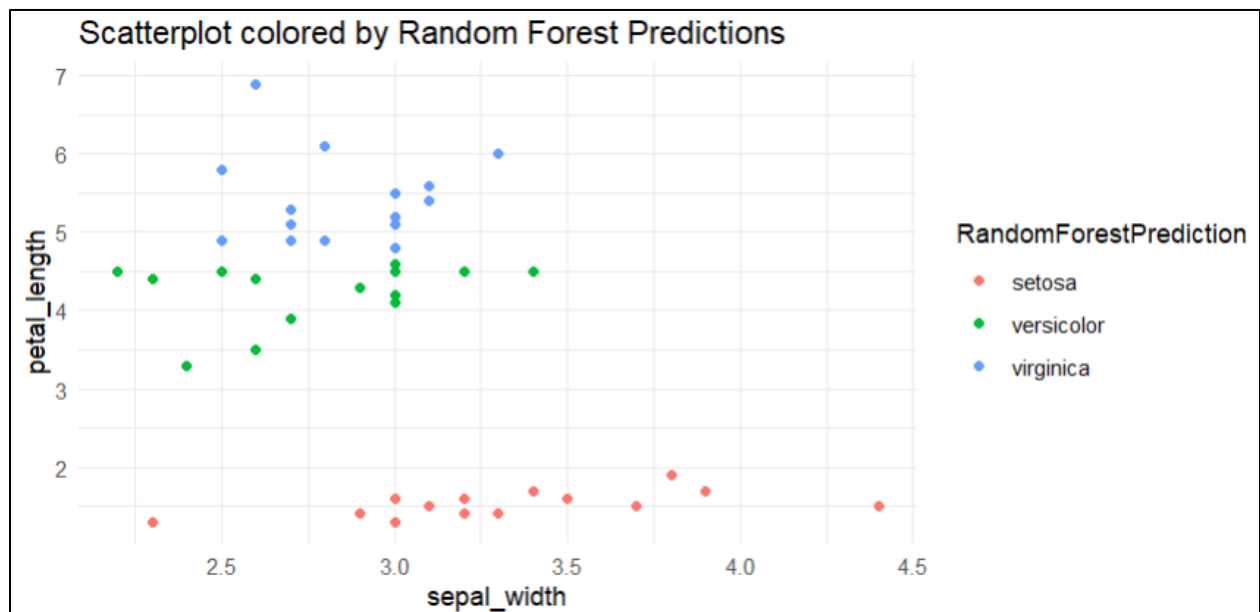
Code for plotting actual Species

```
# Scatterplot with actual species
ggplot(test, aes(x = sepal_width, y = petal_length, color = species)) +
  geom_point() + theme_minimal() +
  labs(title = "Scatterplot colored by Actual Species")
```



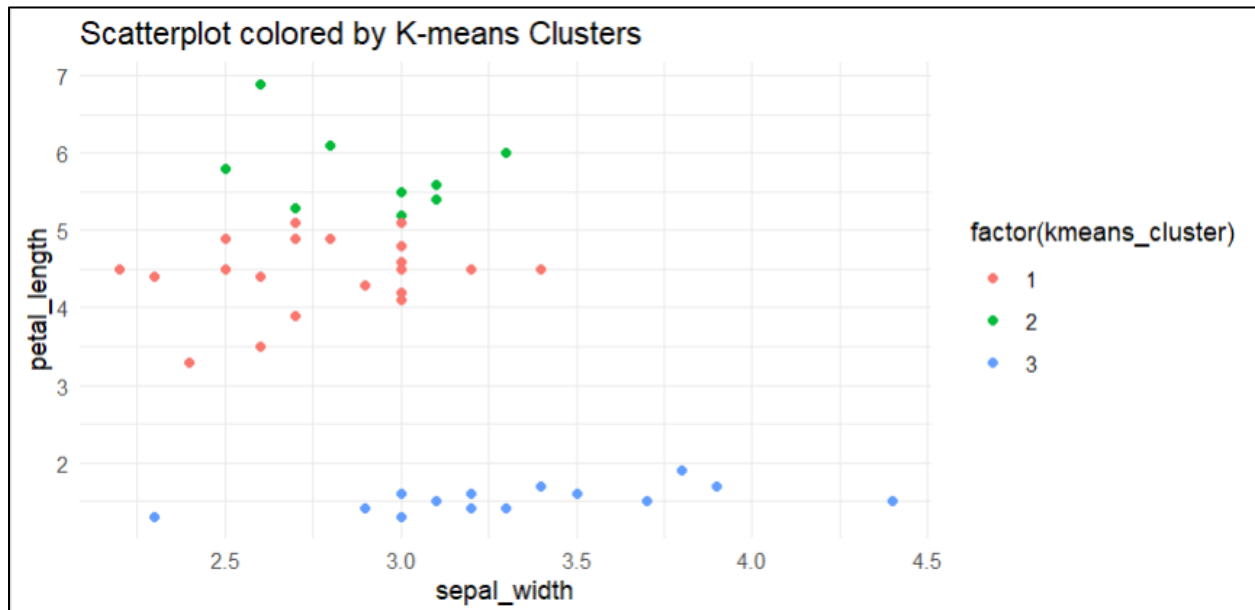
Code for plotting Random Forest predictions

```
# Scatterplot with Random Forest predictions
ggplot(test, aes(x = sepal_width, y = petal_length, color = RandomForestPrediction)) +
  geom_point() + theme_minimal() +
  labs(title = "Scatterplot colored by Random Forest Predictions")
```



Code for plotting k-means assignments

```
# Scatterplot with k-means cluster assignments
ggplot(test, aes(x = sepal_width, y = petal_length, color = factor(kmeans_cluster))) +
  geom_point() + theme_minimal() +
  labs(title = "Scatterplot colored by K-means Clusters")
```



Question 4: can you see why k-means () assigned the clusters as it did? State your observations

K-means assigned clusters based on the similarity of the data points in terms of the features (sepal length, sepal width, petal length, petal width). The clusters are formed to minimize the distance between the data points and the centroid of each cluster. From the observations, k-means clusters data points that are closer in terms of feature values, which is why certain points are grouped together. The centroids represent the average feature values for the clusters, and points are assigned to the nearest centroid based on their feature similarities.

Based on Observations and comparatively to the Actual Scatter plot Random Forest seems to have predicted better than K means clustering.