South Dakota School of Mines and Technology

Data Mining Theory, Spring 2022

CSC 554 - M01

Exam #2

Classifications

The titanic.csv dataset contains 2201 instances or samples and 4 features out of which one is a target category.

The features are as follows:

1) Status - Whether the individual was crew, first, second or third class passengers
2) Age - The age of the individual
3) Sex - M or F
4) Survived - Binary = Y or N

The target category is whether the individual survived or not.

Part 1

For this part I created two models - A Tree Classifier and a Naive Bayes Classifier (most of the other classifiers had really low accuracy numbers). I evaluated both models using 10 fold cross validation and noted the accuracies. I then wrote a small python script to calculate the 95% confidence intervals (the script can be found in Appendix I). The results obtained are as follows:

The accuracy for the **Naive Bayes** classifier model is **77.8%** and the 95% confidence interval calculated from that is **[76.01, 79.48]**

The accuracy for the **Tree** classifier model is **78.8%** and the 95% confidence interval calculated from that is **[77.04 , 80.45]**

The difference in accuracy between the two classifiers is **0.998%.**

Part 2

As per the instructions in the document, I setup a Tree Model and a Naive Bayes Model trained and tested using 10-fold cross validation (implemented using the Data Sampler widget). Figure 1 shows this setup.

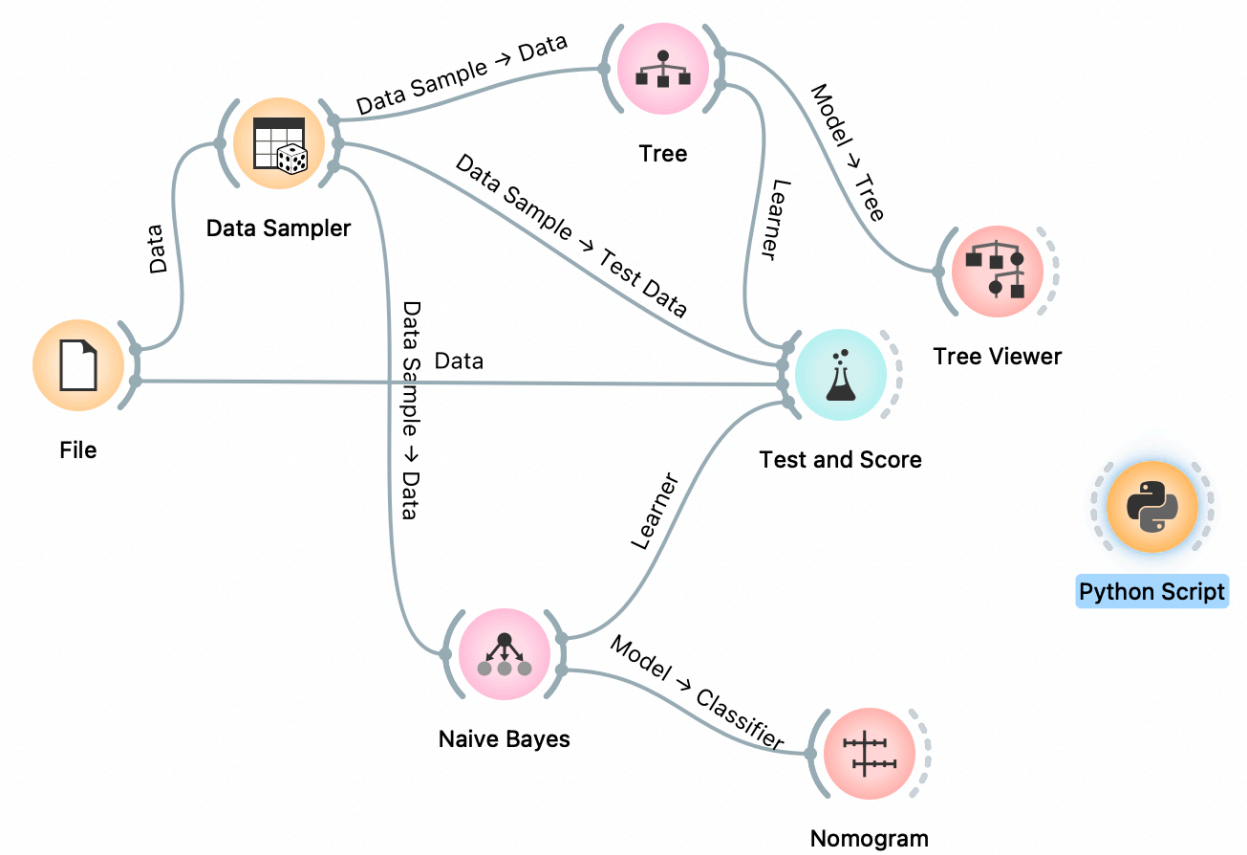

Figure 1. Classifier Comparisons

To calculate the mean of the accuracy differences, I subtracted the Classification accuracies obtained in the Test and Score widget. The mean or $\bar{d}$ is **0.014.**

To calculate the standard deviation, you need to find the error rate (or 1 - accuracy) for each of the partitions. The error rate of each partition was calculated by adjust the parameter in the Data Sample Widget that controls the number of test and training partitions (Figure 2). This was done for values ranging from k-1 (9 in our case) to 1.

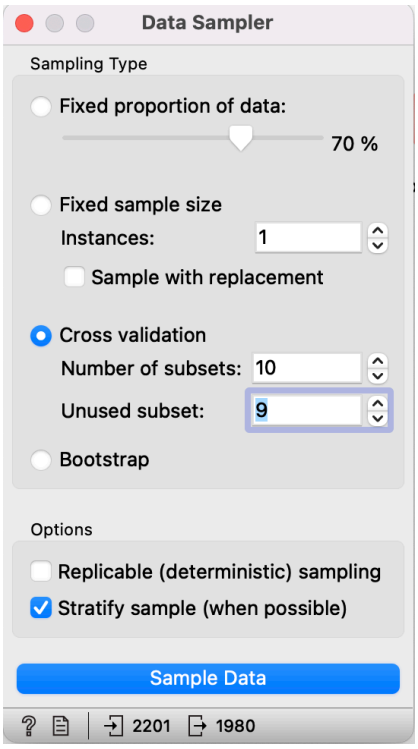

Figure 2.

In other words, what we need is $|e_i - d|^2$. The reported classification (or error rate) accuracy is the mean of the classification accuracies (or error rate) across all k folds. By manipulating the number of training partitions we can get the classification accuracies for each value of k as follows:

$$(e_k - d) = e_{k+1} - e_k$$

The difference between errors (Δe_i or $(e_k - d)$) can be calculated from the above. The results obtained are as follows

$$\Delta e_1 = 0.002 \qquad \Delta e_2 = 0 \qquad \Delta e_3 = 0.001 \qquad \Delta e_4 = 0 \qquad \Delta e_5 = 0.009$$

$$\Delta e_6 = 0.004 \qquad \Delta e_7 = 0.013 \qquad \Delta e_8 = 0 \qquad \Delta e_9 = 0 \qquad \Delta e_{10} = 0$$

Squaring and summing of all of these values followed by their division by k * (k-1) gives us the variance to be **0.0001**.

I then used this value along with the mean difference in the formula:

$$d_t^{cv} = 0.014 \pm 2.26 \cdot 0.0001$$

Since the confidence interval does not span 0, we can say that the observed difference is statistically significant at a 95% confidence interval.

I connected the Tree Classifier to a Tree Viewer and the Naive Bayes to a Nonogram. The results I got are as shown below. From Figure 3 (Tree Classifier), the most important attribute was the **sex**. While from the Naive Bayes Classifier (Fig 4), the highest ranked attribute was the **class.**

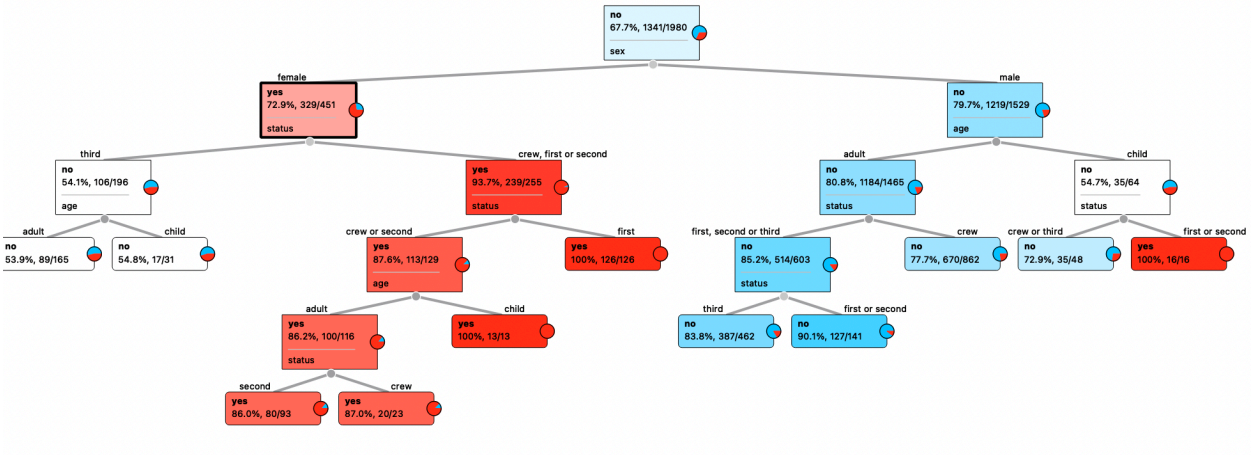

Figure 3. Tree Classifier

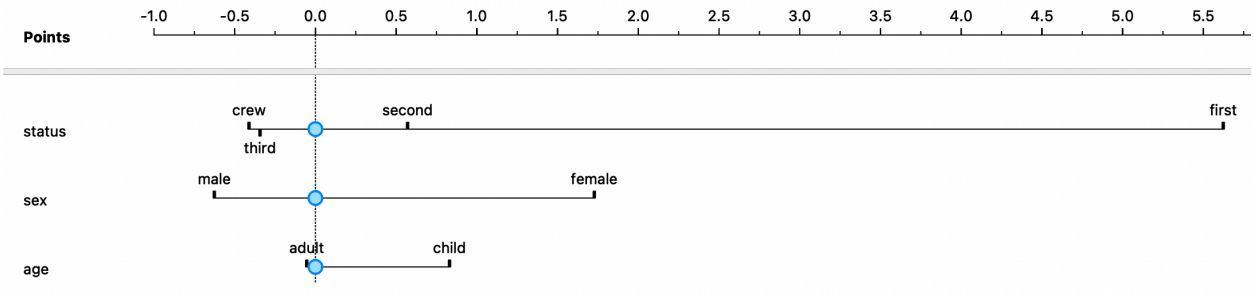

Figure 4. Naive Bayes Classifier

1) Which was the most important attribute to predict survival? What measures did you use?

A. I used the rank widget to find the most important attribute for predicting survival. The results showed that **sex** was the most important attribute (Fig 5.)

	#	Info. gain ▼	Gini
C sex	2	0.142	0.091
C status	4	0.059	0.038
C age	2	0.006	0.004

Fig 5. Rank Widget Result

2) What classifier did you use?

A. I used the Tree Classifier and the Naive Bayes Classifier.

3) What parameters did you use?

A. For the Tree classifier, I used the following parameters in Figure 6. For the Naive Bayes Classifer, there were no parameters I could tweak. I evaluated both using the Test Widget and 10 fold cross validation.

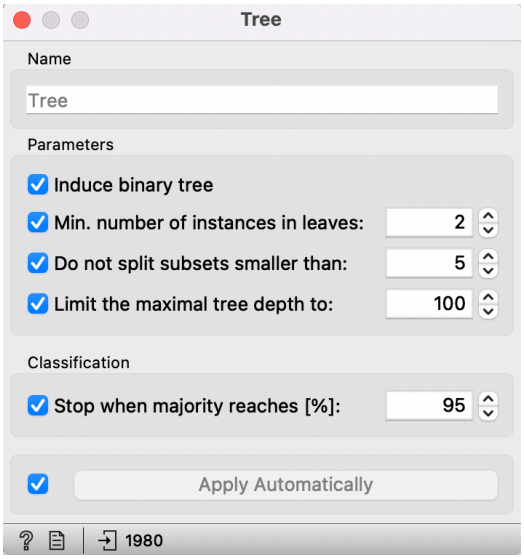

Figure 6. Tree Classifier Parameters

4) Did the most important attribute for survival reflect in your models?

A. The Tree Classifier reflects the most important attribute but the Naive Bayes Classifier considers the Class to be the most important.

5) What was the 95% confidence interval for the accuracy for both models?

A. The accuracy for the **Naive Bayes** classifier model is **77.8%** and the 95% confidence interval calculated from that is **[76.01, 79.48].**

The accuracy for the **Tree** classifier model is **78.8%** and the 95% confidence interval calculated from that is **[77.04 , 80.45].**

6) What was the difference in accuracy between your two classifiers?

A. The difference in accuracy between the two classifiers is **0.998%**.

7) Was the difference statistically significant ?

A. The difference was not statically significant, as shown above.

Clustering

The seeds_dataset.tab dataset contains 210 instances or samples and 6 features out of which one is a target category. The features are numeric and pertain to the physical properties of the seed.

The features are as follows:

1) Perimeter
2) Compactness
3) Length
4) Width
5) Asymetry
6) Groove

The target category is the Kernel, which is categorical and classifies the seed as belonging to type 1, 2 or 3.

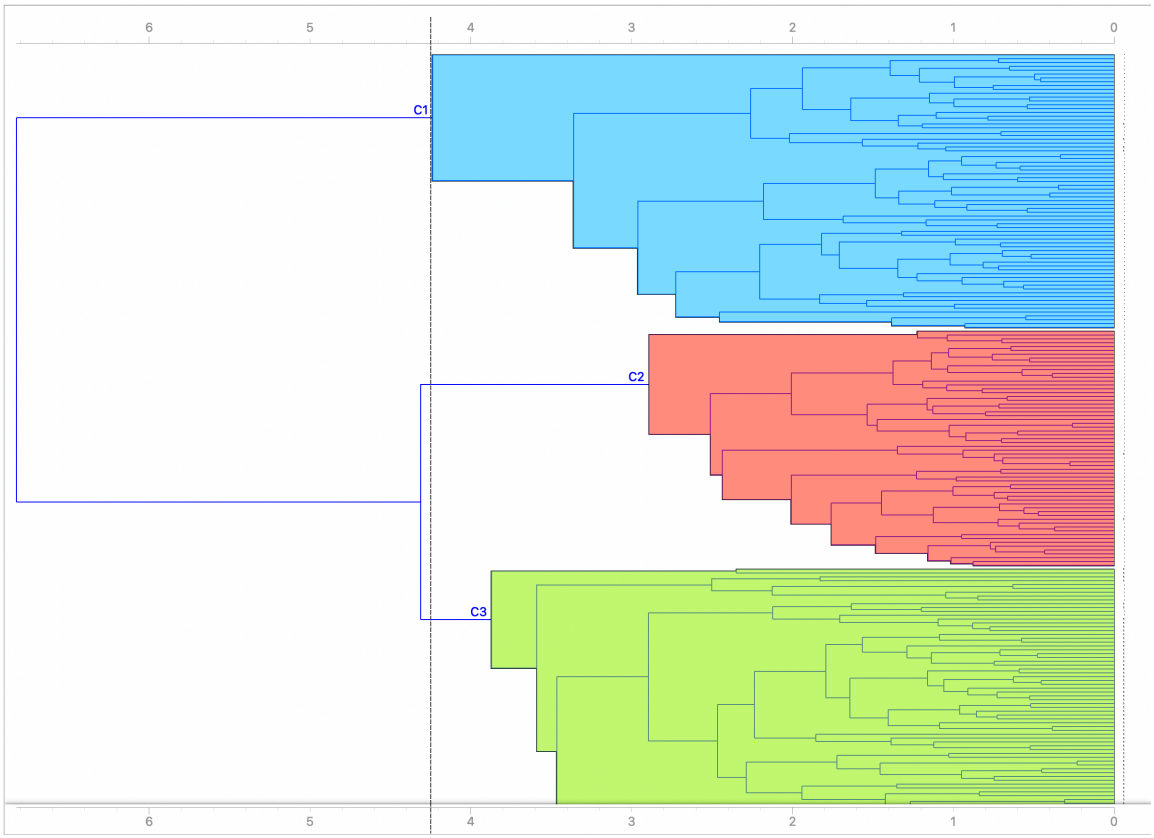

Fig 7. Hierarchical Clustering

I set the distance measure in the distance widget to Euclidean and used Average-Linkage Hierarchical Clustering and adjusted the distance measure till I found 3 clusters (shown above in Figure 7). The distance map obtained is shown below in Figure 8.

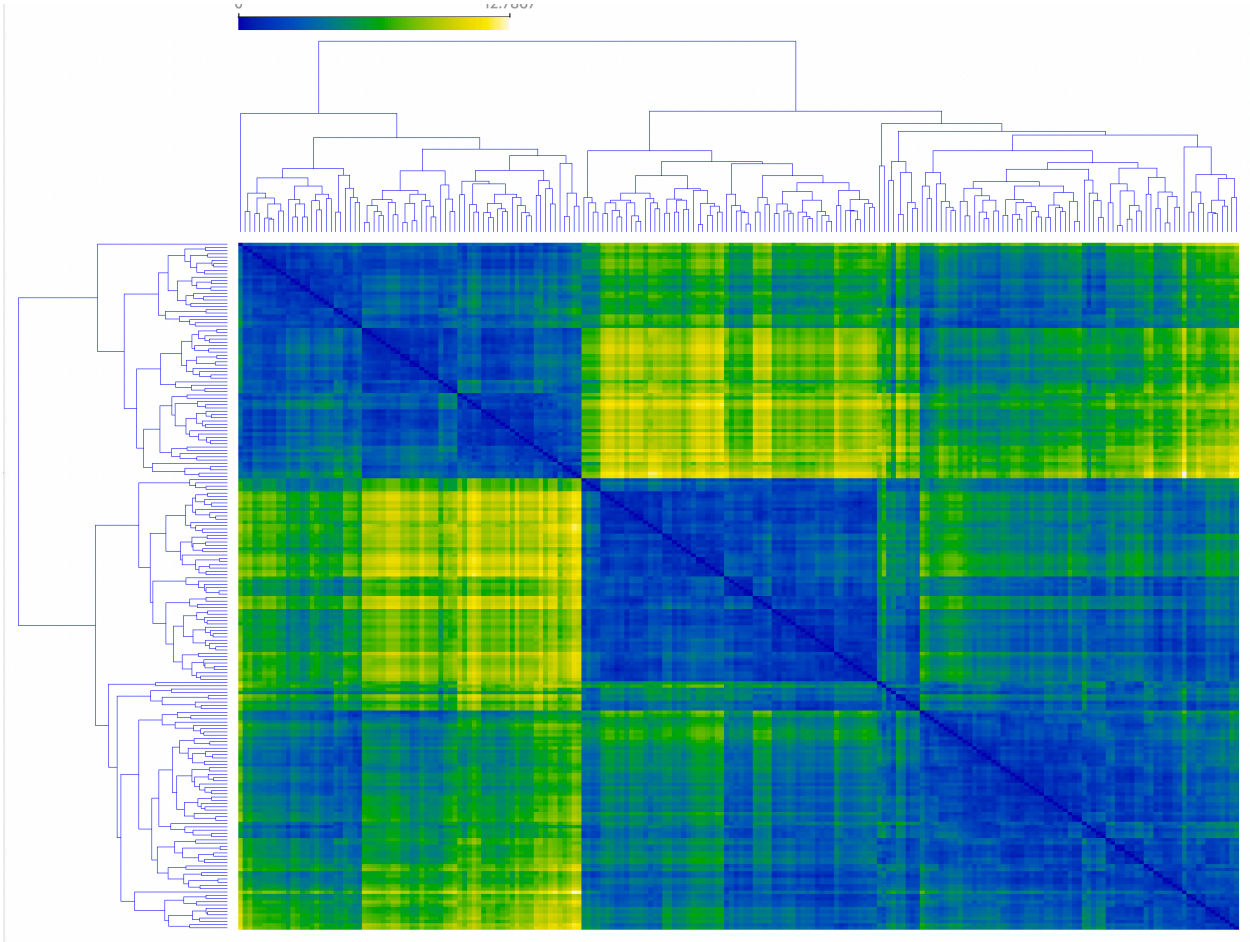

Fig 8. Distance Map

I used the select rows widget fill in the table shown below (Table 1).

	K-means			Hierarchical		
Cluster/ Kernel	1	2	3	1	2	3
C1	64	2	8	4	68	0
C2	5	68	0	2	0	60
C3	1	0	62	64	2	10

Table 1. K-Means and Hierarchical Frequencies

The calculate the entropy, we first need to calculate e_i or the entropy for each row , which is defined as the following:

$$e_i = -\Sigma_{j=1}^{L}\ p_{ij}\, log_2 p_{ij}$$

I wrote a script to automate this (Appendix II) and the results I got are as follows:

K - Means

$e_1 = 0.669$

$e_2 = 0.360$

$e_3 = 0.118$

Hierarchical

$e_1 = 0.309$

$e_2 = 0.206$

$e_3 = 0.731$

The entropy is then calculated using the following formula:

$$e = \Sigma_{i=1}^{K}\ \frac{m_i}{m}\, e_i$$

Using this the K-Means Entropy was calculated to be **0.396** and the Hierarchical entropy was calculated to be **0.432.**

The formula for calculating the recall is

$$recall(i, j) = \frac{m_{ij}}{m_j}$$

Using that the program calculated the recall matrix for K-means to be:

0.914	0.029	0.114
0.071	0.971	0.000
0.014	0.000	0.886

Table 2. Recall Matrix K-means

And the recall matrix for hierarchical clustering to be:

	0	1	2
0	0.057	0.971	0.000
1	0.029	0.000	0.857
2	0.914	0.029	0.143

Table 3. Recall Matrix Hierarchical Clustering

I then outputted the data from the clustering algorithms to .csv files and wrote a script to parse the data and calculate the Jaccard Coefficient and Rand Statistic (Appendix 3).

For the K-means Clustering Algorithm, the values of the f matrix are shown below:

	0	1
0	13642	1058
1	1021	6224

Table 3. f-Matrix K-Means

The sum of it is **21945.**

The Rand Statistic and Jaccard Coefficient for the K-Means Clustering algorithm are **0.9494** and **0.2974** respectively.

For the Hierarchical Clustering Algorithm, the values of the f matrix are as follows:

	0	1
0	13520	1180
1	1128	6117

Table 5. f-Matrix Hierarchical

The sum of it is **21945.**

And the Rand Statistic and Jaccard Coefficient for the Hierarchical Clustering algorithm are **0.9433** and **0.2938** respectively.

I attached a Silhouette Plot and these are the results I got:

For Class 1:

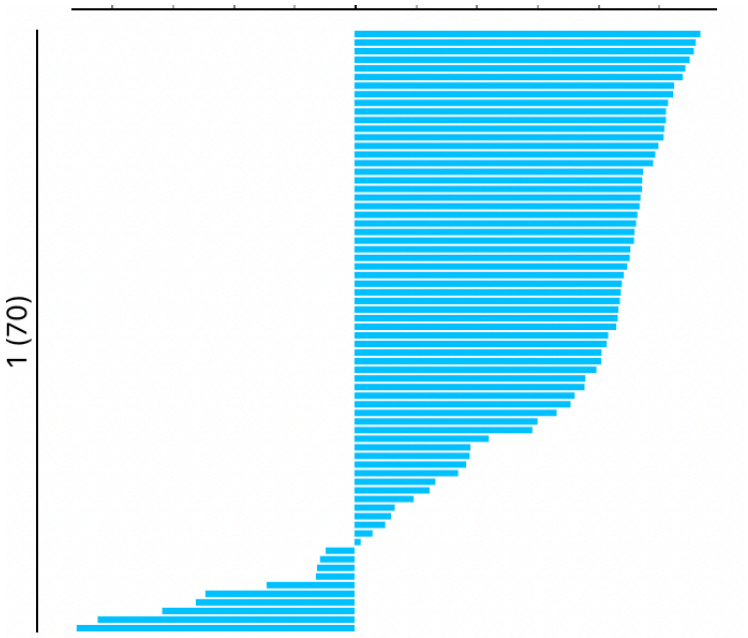

Figure 9. Class 1 Silhouette Plot

For Class 2:

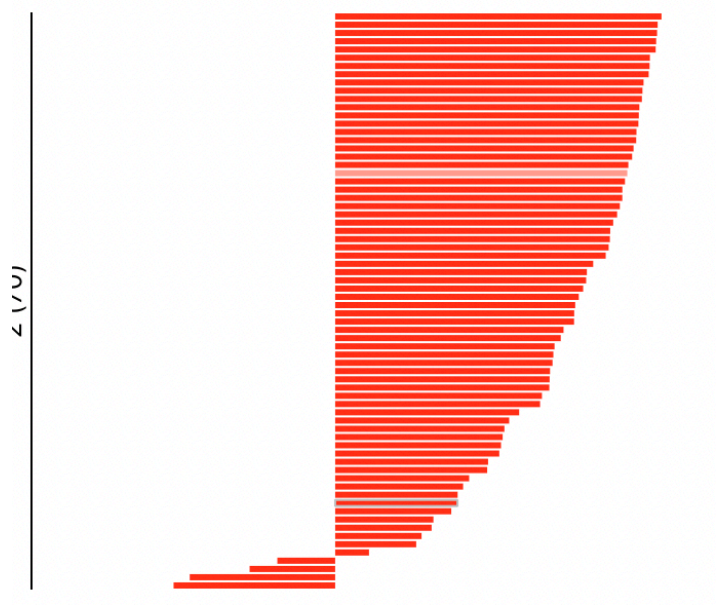

Figure 10. Class 2 Silhouette Plot

For Class 3:

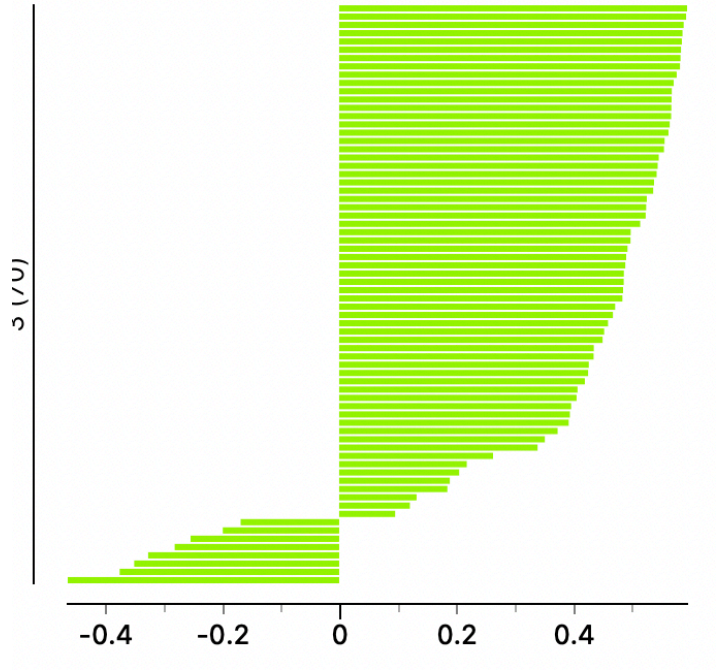

Fig 11. Class 3 Silhouette Plot

1) Attach a Silhouette Plot directly to the dataset. What does this tells us?

A. The Silhouette Plot tells us how close each point in one class is to the other

2) Describe the parameters used for each method (distance metric, linkage, pruning, initialization, re-runs, max iterations, etc)

A. For Hierarchical Clustering, I used Normalized Manhattan Distance as the metric with Average Linkage as this provided the best clustering. I did not use any pruning.
For K-Means Clustering, I used initialization with K-Means++ with 10 reruns and a maximum of 300 iterations.

3) Attach the Distance Matrix to the Distance widget, what could we use this information for?

A. We could use the information from the data matrix to calculate the distance between 2 pairs of seeds. This can then be used to find outliers from cluster means.

4) Give the Silhouette Score and the Selection Height of the 3 clusters?

A. The selection height is **4.25 Manhattan Distances**. The Silhouette Score is **0.400.**

5) What was the entropy and recall for each cluster, and what was the total entropy for each method?

A. The entropy for each cluster is shown below:

K - Means	**Hierarchical**
$e_1 = 0.669$	$e_1 = 0.309$
$e_2 = 0.360$	$e_2 = 0.206$
$e_3 = 0.118$	$e_3 = 0.731$

The K-Means Entropy was calculated to be **0.396** and the Hierarchical entropy was calculated to be **0.432.**

Table 2 and Table 3 show the recall matrix (page 10).

6) What was the count for the f matrix and the sum of it?

A. Table 4 and Table 5 on page 10 and 11 show the f values for each method. The sum of the f values for K-means is **21945** and the sum for Hierarchical Clustering is **21945.** Since the sums are the same, I think my algorithm is fairly correct.

7) What was the Rand Statistic and Jaccard Coefficient?

A. The Rand Statistic and Jaccard Coefficient for the K-Means Clustering algorithm are **0.9494 and 0.2974** respectively. And the Rand Statistic and Jaccard Coefficient for the Hierarchical Clustering algorithm are **.9433 and 0.2938** respectively.

8) How good were the methods?

A. K-Means performed better than Hierarchical Clustering in my case as seen from the Rand Statistic and Jaccard Coefficient.

Association Analysis

The votes.csv dataset contains 435 instances and 17 features out of which one is the target variable. I analyzed the data with both - skipping the party variable and setting it to the target variable - and didn't notice any change. My assumption as to why we want to skip the party variable is that it would end up being the most frequently associated feature.

The target variable is **party** which denote which party an instance/person belongs to.
All the other features have binary outcomes (Yes or No) pertaining whether the individual voted for or against the feature

The other features are:

1) handicapped-infants
2) water-project-cost-sharing
3) adoption-of-the-budget-resolution
4) physician-fee-freeze
5) el-salvador-aid
6) religious-groups-in-school
7) anti-satellite-test-ban
8) aid-to-nicaragua-contras
9) mx-missile
10) immigration
11) synfuels-corporation-cutbacks
12) education-spending
13) superfund-right-to-sue
14) crime
15) duty-free-exports
16) export-administration-act-south-africa

My results from the Frequent Itemsets widgets are as follows:

Itemsets	Support	%
religious-groups-in-schools=y	272	62.53
export-administration-act-south-africa=y	269	61.84
synfuels-corporation-cutback=n	264	60.69
adoption-of-the-budget-resolution=y	253	58.16
crime=y	248	57.01
physician-fee-freeze=n	247	56.78
aid-to-nicaraguan-contras=y	242	55.63
anti-satellite-test-ban=y	239	54.94
handicapped-infants=n	236	54.25
education-spending=n	233	53.56
duty-free-exports=n	233	53.56
immigration=y	216	49.66
el-salvador-aid=y	212	48.74
immigration=n	212	48.74
superfund-right-to-sue=y	209	48.05
el-salvador-aid=n	208	47.82
mx-missile=y	207	47.59
mx-missile=n	206	47.36
superfund-right-to-sue=n	201	46.21
water-project-cost-sharing=y	195	44.83
water-project-cost-sharing=n	192	44.14
handicapped-infants=y	187	42.99
anti-satellite-test-ban=n	182	41.84
aid-to-nicaraguan-contras=n	178	40.92
physician-fee-freeze=y	177	40.69
duty-free-exports=y	174	40
adoption-of-the-budget-resolution=n	171	39.31
education-spending=y	171	39.31
crime=n	170	39.08

Fig 12. Frequent Itemsets Results

From the above data, we can see that people that voted yes for religious groups in schools the highest support or in other words they appear more frequently alongside other features.

From the confidence %, we can imply that in 62.53% of all the supersets formed, this feature will appear.

Expanding the Itemsets we can see what other elements each items is frequently associated with or the supersets its part of. Figure 13 shows the expansion for religious groups in schools.

Itemsets	Support	%
religious-groups-in-schools=y	272	62.53
export-administration-act-south-africa=y	158	36.32
synfuels-corporation-cutback=n	166	38.16
crime=y	214	49.2
duty-free-exports=n	170	39.08
duty-free-exports=n	193	44.37
superfund-right-to-sue=y	186	42.76
crime=y	166	38.16
duty-free-exports=n	154	35.4
mx-missile=n	188	43.22
crime=y	166	38.16
duty-free-exports=n	153	35.17
anti-satellite-test-ban=n	169	38.85
aid-to-nicaraguan-contras=n	166	38.16
mx-missile=n	153	35.17
crime=y	156	35.86
education-spending=y	159	36.55

Figure 13. Expanded View of religious-groups-in-school

In the Association Analysis widget, I set the **minimum support to 40%** and the **minimum confidence to 90%**.

Playing around with the antecedent and consequent filters, I realized that changing their bounds filters the number of items in the antecedent and consequent sets. For example if I set the bounds of the antecedent filter to [3, 4] ; the widget would find rules for me that involve LHS sets that contain a minimum of 3 and a maximum of 4 elements. (By items and elements here I mean features of the dataset)

In our case, considering the dataset, for the antecedent, there are **no rules with more than 3 items;** and there are **no rules with more than 2 items** for the consequent set.

Some of the top rules that showed up for me are shown in the figure below (Figure 14):

Supp	Conf	Covr	Strg	Lift	Levr	Antecedent		Consequent
0.469	0.981	0.478	1.163	1.763	0.203	el-salvador-aid=n	→	aid-to-nicaraguan-contras=y
0.455	0.938	0.485	1.199	1.613	0.173	physician-fee-freeze=n, aid-to-nicaraguan-contras=y	→	adoption-of-the-budget-resolution=y
0.455	0.921	0.494	1.149	1.622	0.175	adoption-of-the-budget-resolution=y, aid-to-nicaraguan-contras=y	→	physician-fee-freeze=n
0.455	0.904	0.503	1.105	1.625	0.175	adoption-of-the-budget-resolution=y, physician-fee-freeze=n	→	aid-to-nicaraguan-contras=y
0.453	0.929	0.487	1.283	1.486	0.148	el-salvador-aid=y	→	religious-groups-in-schools=y
0.448	0.938	0.478	1.188	1.651	0.177	el-salvador-aid=n	→	physician-fee-freeze=n
0.446	0.915	0.487	1.170	1.605	0.168	el-salvador-aid=y	→	crime=y
0.441	0.910	0.485	0.986	1.903	0.209	physician-fee-freeze=n, aid-to-nicaraguan-contras=y	→	el-salvador-aid=n
0.441	0.941	0.469	1.211	1.658	0.175	el-salvador-aid=n, aid-to-nicaraguan-contras=y	→	physician-fee-freeze=n
0.441	0.985	0.448	1.241	1.770	0.192	physician-fee-freeze=n, el-salvador-aid=n	→	aid-to-nicaraguan-contras=y
0.441	0.928	0.476	1.169	1.667	0.177	mx-missile=y	→	aid-to-nicaraguan-contras=y
0.441	0.923	0.478	1.014	1.903	0.209	el-salvador-aid=n	→	physician-fee-freeze=n, aid-to-nicaraguan-contr
0.432	0.904	0.478	1.216	1.554	0.154	el-salvador-aid=n	→	adoption-of-the-budget-resolution=y
0.432	0.904	0.478	1.149	1.645	0.169	el-salvador-aid=n	→	anti-satellite-test-ban=y
0.432	0.913	0.474	1.320	1.460	0.136	mx-missile=n	→	religious-groups-in-schools=y
0.430	0.930	0.462	1.204	1.672	0.173	adoption-of-the-budget-resolution=y, anti-satellite-test-ban=y	→	aid-to-nicaraguan-contras=y
0.428	0.921	0.464	1.252	1.583	0.158	physician-fee-freeze=n, education-spending=n	→	adoption-of-the-budget-resolution=y
0.428	0.925	0.462	1.229	1.630	0.165	adoption-of-the-budget-resolution=y, education-spending=n	→	physician-fee-freeze=n
0.428	0.912	0.469	1.240	1.568	0.155	el-salvador-aid=n, aid-to-nicaraguan-contras=y	→	adoption-of-the-budget-resolution=y
0.428	0.989	0.432	1.287	1.778	0.187	adoption-of-the-budget-resolution=y, el-salvador-aid=n	→	aid-to-nicaraguan-contras=y
0.425	0.939	0.453	1.228	1.688	0.173	physician-fee-freeze=n, anti-satellite-test-ban=y	→	aid-to-nicaraguan-contras=y
0.423	0.902	0.469	1.172	1.642	0.165	el-salvador-aid=n, aid-to-nicaraguan-contras=y	→	anti-satellite-test-ban=y
0.423	0.979	0.432	1.287	1.759	0.183	el-salvador-aid=n, anti-satellite-test-ban=y	→	aid-to-nicaraguan-contras=y
0.418	0.924	0.453	1.284	1.588	0.155	physician-fee-freeze=n, anti-satellite-test-ban=y	→	adoption-of-the-budget-resolution=y
0.418	0.905	0.462	1.229	1.595	0.156	adoption-of-the-budget-resolution=y, anti-satellite-test-ban=y	→	physician-fee-freeze=n
0.416	0.933	0.446	1.273	1.643	0.163	aid-to-nicaraguan-contras=y, education-spending=n	→	physician-fee-freeze=n
0.416	0.928	0.448	1.297	1.596	0.155	physician-fee-freeze=n, el-salvador-aid=n	→	adoption-of-the-budget-resolution=y
0.416	0.963	0.432	1.314	1.696	0.171	adoption-of-the-budget-resolution=y, el-salvador-aid=n	→	physician-fee-freeze=n
0.414	0.928	0.446	1.402	1.484	0.135	el-salvador-aid=y, crime=y	→	religious-groups-in-schools=y
0.414	0.914	0.453	1.259	1.603	0.156	el-salvador-aid=y, religious-groups-in-schools=y	→	crime=y
0.411	0.923	0.446	1.304	1.586	0.152	aid-to-nicaraguan-contras=y, education-spending=n	→	adoption-of-the-budget-resolution=y
0.411	0.904	0.455	1.051	1.891	0.194	adoption-of-the-budget-resolution=y, physician-fee-freeze=n, aid-to-nicaraguan-contras=y	→	el-salvador-aid=n
0.411	0.932	0.441	1.318	1.603	0.155	physician-fee-freeze=n, el-salvador-aid=n, aid-to-nicaraguan-contras=y	→	adoption-of-the-budget-resolution=y
0.411	0.962	0.428	1.328	1.695	0.169	adoption-of-the-budget-resolution=y, el-salvador-aid=n, aid-to-nicaraguan-contras=y	→	physician-fee-freeze=n
0.411	0.989	0.416	1.337	1.778	0.180	adoption-of-the-budget-resolution=y, physician-fee-freeze=n, el-salvador-aid=n	→	aid-to-nicaraguan-contras=y

0.13607081516712907

Figure 14. Association Rules

I honestly couldn't find any weirdly interesting rules, even after changing to the minimum support % to being 1%. Most of the data felt kind of hard to interpret as it was taken from a whole different era. Wanted a dataset like the milk and beer dataset.

Appendix 1

```
import math
import numpy as np

N = 2201
acc_naive_bayes = 0.778
acc_tree = 0.788

acc = np.array([acc_naive_bayes, acc_tree])

# acc = acc_tree

Z = 1.96

num1 = 2 * N * acc + Z ** 2
num2 = Z * ((Z**2 + 4*N*acc - 4*N*(acc**2)) ** 0.5)
denom = 2 * (N + Z**2)

conf_interval = (num1 - num2)/denom,  (num1 + num2)/denom
mean_difference = np.mean([(conf_interval[0])[0]*100, (conf_interval[1])[0]*100]) - \
                  np.mean([(conf_interval[0])[1]*100,(conf_interval[1])[1]*100])


print("95% Confidence Interval for the Naive Bayes Classifier= [", (conf_interval[0])[0]*100, ",",
      (conf_interval[1])[0]*100, "]")
print("95% Confidence Interval for the Tree Classifier= [", (conf_interval[0])[1]*100, ",",
      (conf_interval[1])[1]*100, "]")
print("The mean difference is ", abs(mean_difference))
```

Appendix 2

```python
import numpy as np
import math

############### Data Table ##################
k_means = np.array([[64, 2, 8],
                    [5,68,0],
                    [1, 0, 62]])

hierarchical = np.array([[4, 68, 0],
                    [2,0,60],
                    [64, 2, 10]])
N = np.sum(k_means) # total number of elements

##### Entropy vars #######
entropy_kmeans = []
entropy_hierarchical = []
entropy_hierarchical_sum = 0
entropy_kmeans_sum = 0

##### Recall Vars ########
recall_kmeans = np.zeros(np.shape(k_means))
recall_hierarchical = np.zeros(np.shape(k_means))

############## Entropy Calculations ##########################################################################################
for x in k_means:
    row_entropy = 0
    for y in x:
        p = y / np.sum(x, axis=0, dtype=float)
        if p == 0:
            continue
        row_entropy = row_entropy + p * math.log2(p)
    entropy_kmeans.append(-row_entropy)

for x in hierarchical:
    row_entropy = 0
    for y in x:
        p = y / np.sum(x, axis=0, dtype=float)
        if p == 0:
            continue
        row_entropy = row_entropy + p * math.log2(p)
    entropy_hierarchical.append(-row_entropy)

for i in range(0,3):
    entropy_hierarchical_sum = entropy_hierarchical_sum + np.sum(hierarchical[i])*entropy_hierarchical[i]/N
    entropy_kmeans_sum = entropy_kmeans_sum + np.sum(k_means[i])*entropy_kmeans[i]/N
#####################################################################################################################

for i in range(0, len(k_means)):
    for j in range(0, len(k_means[i])):
        recall_kmeans[i,j] = k_means[i,j]/np.sum(k_means[:,j])
        recall_hierarchical[i,j] = hierarchical[i,j]/np.sum(hierarchical[:,j])

print("End")
```

Appendix 3

```python
import csv
import numpy as np

def main():
    kernel_hierarchical=[]
    cluster_hierarchical=[]
    kernel_kmeans=[]
    cluster_kmeans=[]
    f_hier = np.zeros([2,2])
    f_kmeans = np.zeros([2,2])

    with open('heirarchical.csv', newline='') as csvfile:
        csvReader = csv.DictReader(csvfile)
        for row in csvReader:
            kernel_hierarchical.append(row['Kernel'])
            cluster_hierarchical.append(row['Cluster'])

    with open('k_means.csv', newline='') as csvfile:
        csvReader = csv.DictReader(csvfile)
        for row in csvReader:
            kernel_kmeans.append(row['Kernel'])
            cluster_kmeans.append(row['Cluster'])

    kernel_kmeans = kernel_kmeans[2:len(kernel_kmeans)]
    kernel_hierarchical= kernel_hierarchical[2:len(kernel_hierarchical)]
    cluster_kmeans = cluster_kmeans[2:len(cluster_kmeans)]
    cluster_hierarchical = cluster_hierarchical[2:len(cluster_hierarchical)]

    for i in range(0, len(kernel_kmeans)):
        for j in range(i+1, len(kernel_kmeans)):
            if kernel_kmeans[i] == kernel_kmeans[j] and cluster_kmeans[i] == cluster_kmeans[j]:
                f_kmeans[1,1] = f_kmeans[1,1] + 1
            elif kernel_kmeans[i] != kernel_kmeans[j] and cluster_kmeans[i] != cluster_kmeans[j]:
                f_kmeans[0,0] = f_kmeans[0,0] + 1
            elif kernel_kmeans[i] == kernel_kmeans[j] and cluster_kmeans[i] != cluster_kmeans[j]:
                f_kmeans[1,0] = f_kmeans[1,0] + 1
            elif kernel_kmeans[i] != kernel_kmeans[j] and cluster_kmeans[i] == cluster_kmeans[j]:
                f_kmeans[0,1] = f_kmeans[0,1] + 1
            else:
                print("Unclassified - KMeans")

    for i in range(0, len(kernel_hierarchical)):
        for j in range(i+1, len(kernel_hierarchical)):
            if kernel_hierarchical[i] == kernel_hierarchical[j] and cluster_hierarchical[i] == cluster_hierarchical[j]:
                f_hier[1, 1] = f_hier[1, 1] + 1
            elif kernel_hierarchical[i] != kernel_hierarchical[j] and cluster_hierarchical[i] != cluster_hierarchical[j]
                f_hier[0, 0] = f_hier[0, 0] + 1
            elif kernel_hierarchical[i] == kernel_hierarchical[j] and cluster_hierarchical[i] != cluster_hierarchical[j]
                f_hier[1, 0] = f_hier[1, 0] + 1
            elif kernel_hierarchical[i] != kernel_hierarchical[j] and cluster_hierarchical[i] == cluster_hierarchical[j]
                f_hier[0, 1] = f_hier[0, 1] + 1
            else:
                print("Unclassified - Hierarchical")

    print("K-means Rand Statistic and Jaccard Coeff are: ", (f_kmeans[0,0]+f_kmeans[1,1])/np.sum(f_kmeans), " and ",
          (f_kmeans[1,1])/np.sum(f_kmeans))
    print("Hierarchical Rand Statistic and Jaccard Coeff are: ", (f_hier[0, 0] + f_hier[1, 1]) / np.sum(f_hier),
          " and ", (f_hier[1, 1]) / np.sum(f_hier))


if __name__ == '__main__':
    main()
```