AP ITEC 3040 Group Project

Final Report

Professor Xiaofeng Zhou

Members:   
Corey He - 217253527

Moh’d Almasri – 216855280

Dylan Nguyen - 215721640

Abderahman Hussein- 217293515

Prayanshu Singla- 216615429

Dataset: Early detection of diabetes

Checkpoint Progress Report

What do you want to solve: We want to find a subset of attributes (visible symptoms related to diabetes) from the original dataset that we can use to create a model that can diagnose a patient of diabetes.

Why is it important: The diabetes diagnosis model is important because it will allow physicians to diagnose diabetes through observations and the patient’s health record before the patient enters the later-stages of diabetes.

What results do you expect: We expect that we will be able to find a set of attributes from the dataset’s attributes that can be used to create a model that accurately diagnoses diabetes.

Describe and Summarise the data you choose: The dataset we chose is a dataset from a study done to diagnose diabetes within patients. In the dataset, there are 520 tuples, with a binary “Positive/Negative” class, and 16 attributes. Attributes are in order down below.

* Age
* Gender
* Polyuria: Medical term for the condition in which an individual produces more than 3 liters of urine per day
* Polydipsia: Medical term for the condition in which an individual feels extreme thirst
* Sudden Weight Loss
* Weakness
* Polyphagia: Medical term for the condition in which individual consumes abnormally large amounts of food
* Genital Thrush: Medical term for the yeast infection on the genitals
* Visual Blurring: Medical term for the condition in which the eye lens swells, causing the individual’s eyesight to blur
* Itching
* Irritability
* Delayed Healing
* Partial Paresis: Paresis is the condition in which muscle movement is weakened. Partial Paresis means the partial weakening of muscles.
* Muscle Stiffness: Muscle stiffness from atrophication
* Alopecia: Medical term for baldness caused by the immune system attacking the hair follicles.
* Obesity.

**Classification problem**: We want the model to be able to go through a tuple’s subset of attributes and their values and determine whether or not the tuple’s class (diabetes diagnosis) is positive or negative.

Schedule:

1. Clean Data (unnecessary) the data is already clean.
2. Analysing existing data, and creating graphs to understand the existing data, we did pie charts to see the numbers of each attribute as well as other analyses to understand the relationship between all the attributes.
3. Reduce data dimensionality to reduce the number of attributes we did a couple of tests on age and gender to see if they will help us with data deduction. The next step was to have a look at the positive values in order to make a decision on what we can reduce
4. Applying the chi-square test, we did a chi-square test to confirm our finding in the previous steps, and then we did it with the class to reach our smallest possible attribute subset.
5. The next step is to do a decision tree to classify the attributes for future unknown attributes.
   1. We will use other algorithms to determine if the symptoms that someone has are possible diabetes or not.

Progress Report:

We went through the tuples and the attributes and found no missing/incorrect values. Due to the small size of the dataset (520 tuples), we decided that we do not need to reduce the size of our dataset for our EDA. As we explored the dataset and its attributes, we believe that there are more attributes than needed for our model, and as such, we decided the dimensionality of the dataset needed to be reduced.

We have determined a few attributes that may be removed using various methods.

* Dimensionality Reduction via removal of the attributes by lowest occurrences occurring within tuples that have the diabetes class with a “positive” value.
* Dimensionality Reduction via the Chi-Square test. Attribute pairs with chi-square values above the significance level (1%) can have one of their attributes inside the pair removed.

## Age Correlation (by Dylan)

In this graph we wanted to see at what age people detect early symptoms of diabetes. By looking at this graph we can tell that after the age of 35 more than sixty patients experience symptoms of early diabetes shown by the bars colored blue. You can also see that there is a much larger sample size for positive patients as opposed to negative patients. Since the graph shows a broad range of ages that are positive in detecting early diabetes. We also took a look at the negative result as well and to see if the negative result can help us understand whether or not age will play a part in our investigation. In conclusion, the information gathered by this graph does not give us any information needed to formulate a conclusion

## Finding the most common symptoms (by Moh’d)

[See code here](#_heading=h.49x2ik5)

In this part, we wanted to clean up the data, as we all know most of the symptoms that sickness has will either be a major or a minor attribute. To do that we wanted to use the data in the early detection of diabetes data set to make our discussion. In the dataset, we wanted to make our decision with the positive class since these are the people that have diabetes. We created a new table from the dataset with only the positive class so we can eliminate the least occurring values in the data set. After creating the new table we made a bar graph with the number of each yes for the attributes in a bar graph.

In order to get the percentage of each attribute, we divided the number of each attribute by the total number of positive cases and we had these results.

| Attribute Yes-Diabetes / Total positive percentage  (3)Polyuria 243/320 = 0.7593 75.93%  (4)Polydipsia 225/ 320 = 0.7968 79.68%  (5)Sudden weight loss 188/320= 0.5875 58.75%  (6)Weakness 218/320 = 0.6812 68.12%  (7)Polyphagia 189/320 = 0.5906 59.06%  (8)Genital thrush 83/320 = 0.2593 25.93%  (9)Visual blurring 175/320 = 0.5468 54.68% | Attribute Yes-Diabetes / Total positive percentage  (10)Itching 154/320 = 0.4812 48.12%  (11)Irritability 110/320 = 0.3437 34.37%  (12)Delayed healing 153/320 = 0.4781 47.81%  (13)Partial paresis 192/320 = 0.6000 60.00%  (14)Muscle stiffness 135/320 = 0.4218 42.18%  (15)Alopecia 78/320 = 0.2437 24.37%  (16)Obesity 61/320 = 0.1906 19.06% |
| --- | --- |

From lowest to highest percentage (below 50%):

**Obesity, Alopecia, Genital Thrush, Irritability, Muscle Stiffness, Delayed Healing, and itching**

Since these are the lowest value in the positive cases we want to get rid of some of them; however, we cannot make the decision only with this test. To make sure these numbers are true for the positive value we will be doing a chi-square test next to confirm our decision.

## Corey’s Chi-Square Tests

[See code here](#_heading=h.nmf14n)

### Analysis

Chi-Square Test of Independence between two attributes. Contingency tables have been left out from the report due to the space they would take up.  
Results: (Attribute pairs and Chi-Square value)

| Polyuria & Polydipsia 186.3331  Polyuria & sudden weight loss 103.9969  Polyuria & weakness 72.6862  Polyuria & partial paresis 101.4347  Polydipsia & sudden weight loss 85.7  Polydipsia & weakness 57.4731  Polydipsia & Polyphagia 52.2012  Polydipsia & Visual Blurring 57.0577  Polydipsia & Partial Paresis 101.7039  Polydipsia & Alopecia 50.2832 | Weakness & Visual Blurring 47.1259  Weakness & Itching 49.7917  Weakness & Delayed Healing 58.5337  Polyphagia & Visual Blurring 44.8078  Polyphagia & Partial Paresis 72.5681  Polyphagia & Muscle Stiffness 53.2583  Visual Blurring & Itching 44.092  Visual Blurring & Partial Paresis 68.9569  Visual Blurring & Muscle Stiffness 88.4249  Itching & Delayed Healing 106.8578 |
| --- | --- |

These attribute pairs have chi-square test values above the 1% significance level (6.6352) threshold which means that they are not independent of one another. As they are not independent of one another, it is possible to reduce the dataset of 16 attributes to a smaller subset of the dataset by removing some of the attributes in these pairs.

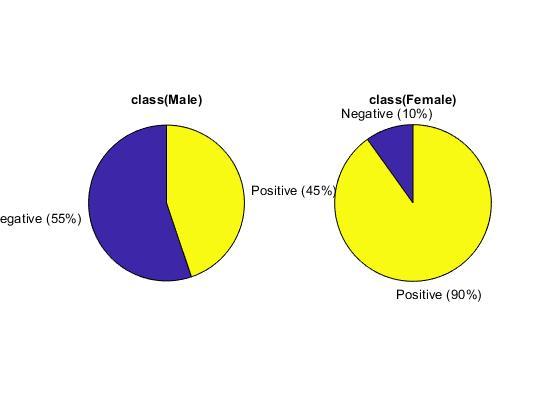
Chi-Square Test of Independence between an attribute and the class [(See code here)](#_heading=h.3o7alnk)

| Polyuria 230.5954  Polydipsia 218.8448  Sudden Weight Loss 99.1077 | Polyphagia 61.0006  Itching 46.6339  Partial Paresis 97.1737 |
| --- | --- |

These attribute pairs have chi-square test values above the 1% significance level (6.6352) threshold which means that they are not independent of one another. As these attributes are not independent of the class, it is very likely that we can use these attributes as the subset of attributes to develop a model to predict the class.

## Gender

* "Sex differences in health and medicine have been recognized for years — in 2004, the American Heart Association’s Go Red For Women campaign, for instance, highlighted that men and women experience heart attacks differently — but researchers have only recently begun to elucidate the underlying hormonal and genetic factors at play. Their findings have implications for diagnosis, public health messages, and treatment. "(Sujata Gupta, 2016).​
* The point above is one of the reasons we chose to separate the dataset into male and female. Given that the majority of the positive tuples for diabetes in our dataset are female (54%), and that only 37% of all the tuples in the dataset are female, it seems worthwhile to split on this attribute. Also considering that the symptoms for female positives and male positives could differ as has been the case in studies that we've referenced at the end of the presentation slides. For this preliminary part of the project, simple EDA procedures have been used on both male and female data entries respectively and placed side-by-side for comparison. The coding for this has been documented and will be submitted with the final report as well as different data mining techniques that are going to be applied.



## Progress Report Conclusion:

We now have an understanding of the attributes that will be useful in creating our model and the attributes that are not. The next step would be to create and train our model using the attributes that we have identified as useful.

Difficulties and issues/ Possible Solutions to said difficulties:

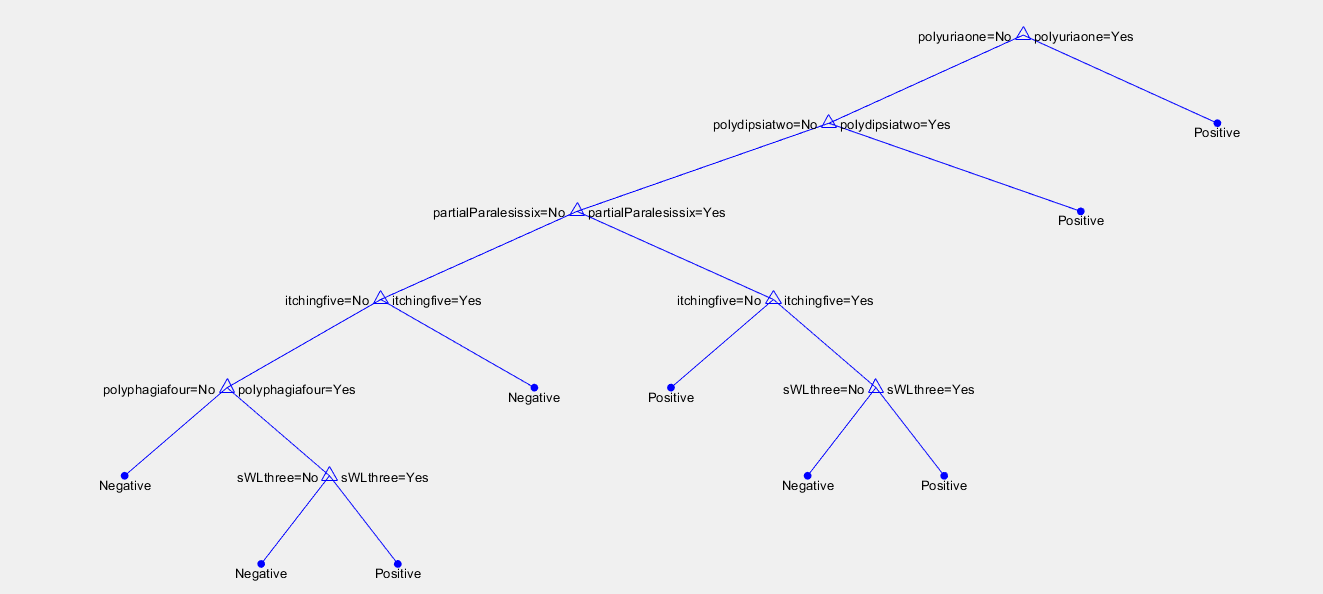
A difficulty we had stemmed from not assigning concrete roles. We all contributed but it became apparent to us after having to reschedule new meetings to check for new findings that it would have saved us much time if we had assigned tasks so we could hold each other accountable

After Progress Checkpoint

Now that we have reduced and created a set of potential attributes for our classification model, it is time to create our model. We decided to create our model based on Binary Decision Trees and the Naive Bayesian Classifier as our attributes are binary categorical attributes.

## Classification using the Binary Decision Tree: Information Gain

The decision Tree was constructed using the information gain method, and since the attributes had only two values (namely “Yes” and “No”) the tree turned out to be a binary decision tree. The results from this tree indicate that Polyuria is the single most attribute that gives us the most insight into whether a person has diabetes or not (i.e, it is the purest attribute). It is important to note that the tree was constructed considering only the 6 attributes that we found in our checkpoint to be most correlated with the class attribute and passed the chi-square test– Plouria, Polydipsia, Polyphagia, sudden weight loss, Itching, and Partial Paresis. The results and accuracy of our tree are indicated below along with the evaluation methods we used. We also created a second decision tree using the Gini index.



Results could be found below.

### Evaluation Method: Holdout [(code here)](#_heading=h.32hioqz)

Training Set: 2/3rds of the 520 tuples.

Test Set: 1/3rd of the 520 tuples.

| Accuracy =  0.8506 | ErrorRate =  0.1494 | Sensitivity =  0.8246 | Specificity =  0.9000 |
| --- | --- | --- | --- |
| Precision =  0.9400 | Recall =  0.8246 | Fmeasure =  0.8785 |  |

When the code is run, these numbers will not be exact as the tuples are randomly selected to be divided into the training and test sets.

Accuracy of 85.06% 85.06% of the 520 tuples were correctly identified.  
Error Rate of 14.94% 14.94% of the 520 tuples were incorrectly identified.  
Sensitivity of 82.46% 82.46% of the positive tuples were correctly identified.  
Specificity of 90% 90% of the negative tuples were correctly identified.  
Precision of 94% 94% of the 520 tuples were correctly identified as positive.  
Recall of 82.46% 82.46% of the positive 320 tuples were correctly identified as positive   
F Measure of 87.85%

### Evaluation Method: K-Fold Cross Validation (k=10) [(code here)](#_heading=h.1hmsyys)

Training Set: 9/10ths of the 520 tuples.

Test Set: 1/10th of the 520 tuples.

| meanAcc =  0.9058 | meanER =  0.0942 | meanSens=  0.9158 | meanSpecif =  0.9442 |
| --- | --- | --- | --- |
| meanPrec =  0.9060 | meanReca =  0.9158 | meanFmeas =  0.9056 |  |

When the code is run, these numbers will not be exact as the tuples are randomly selected to be divided into the training and test sets.

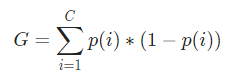
Mean Accuracy of 90.58% 90.58% of the 520 tuples were correctly identified.  
Mean Error Rate of 9.42% 9.42% of the 520 tuples were incorrectly identified.  
Mean Sensitivity of 91.58% 91.58% of the positive tuples were correctly identified.  
Mean Specificity of 94.42% 94.42% of the negative tuples were correctly identified.  
Mean Precision of 90.6% 90.6% of the 520 tuples were correctly identified as positive.  
Mean Recall of 91.58% 91.58% of the positive 320 tuples were correctly identified as positive   
Mean F Measure of 90.56%

The results of the Holdout Method show that the model is somewhat lacking as it only has an accuracy rate of 85.06%. However, using the K-Fold Cross Validation method and getting the mean accuracy of the model shows that the model is actually around 90.58% accurate. This shows that the Holdout Method’s evaluation is somewhat misleading and that our model actually performs better than the Holdout Method says it does.

This can be seen from the low recall rate of 82.46% on the Holdout method compared to the mean recall rate of 91.58% on the K-Fold Cross Validation method.

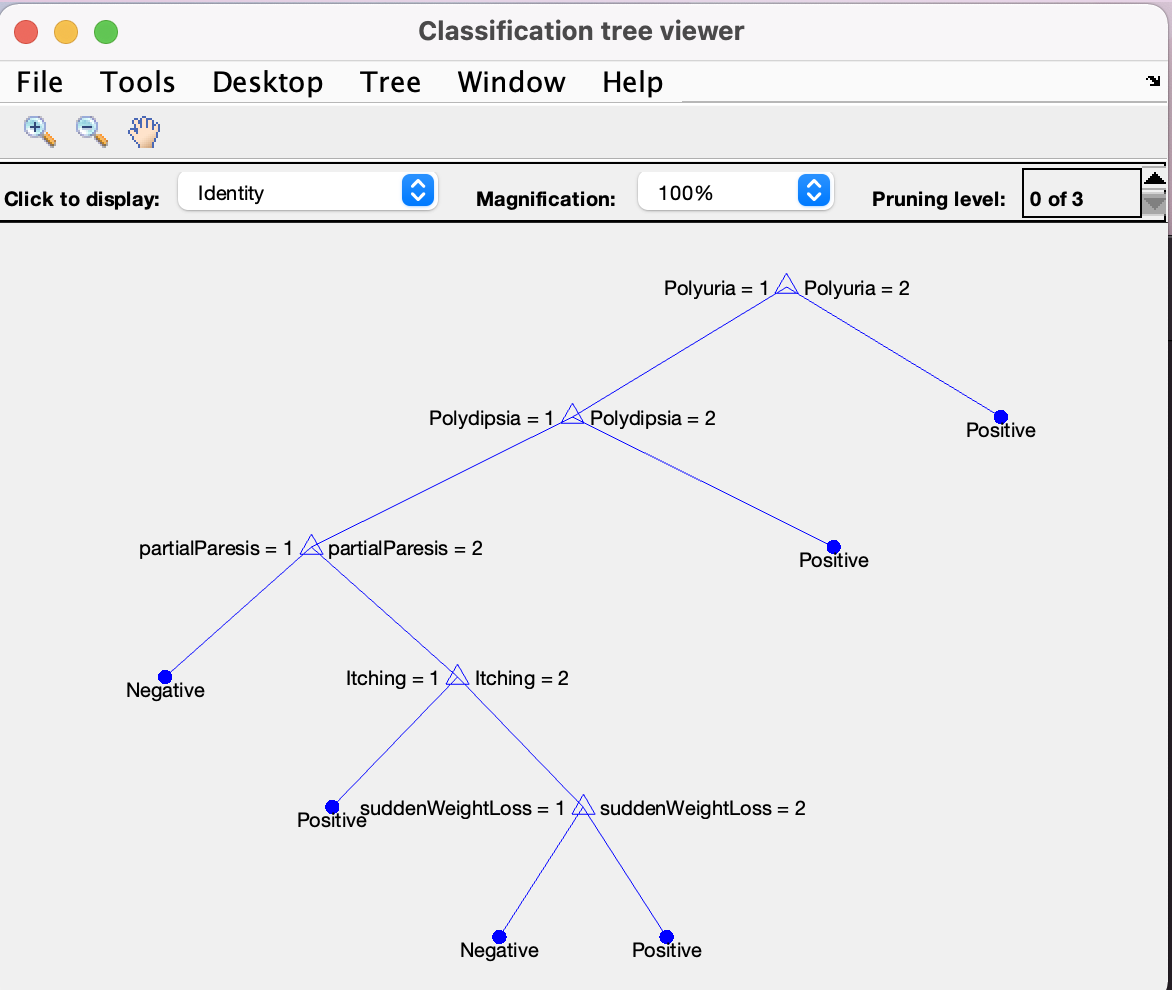
## Classification using the Binary Decision Tree: Gini Index

Decision tree: The decision tree algorithm falls under the category of supervised learning. The decision tree uses the representation to solve the problem in which each leaf node corresponds to a class label and attributes are represented on the internal node of the tree. The formula for the Gini index:-

Here, C is the total number of classes, and p(i) is the probability of picking the data point with the class i.

We used a decision tree algorithm for our dataset to represent boolean function on selected attributes. Following attributes were identified during the attribute selection process: Polydipsia, Polyuria, suddenweightloss, Itching, partialParesis. For supervised learning, we used ⅔ of the dataset to train our algorithm. And the rest of the data will be used for the testing set.

Following is the result of the Binary Decision Tree: Gini index



### Evaluation Method: Holdout [(code here)](#_heading=h.vx1227)

Training Set: 2/3rds of the 520 tuples.

Test Set: 1/3rd of the 520 tuples.

| Accuracy =  0.8333 | ErrorRate =  0.1667 | Sensitivity =  0.8468 | Specificity =  0.8095 |
| --- | --- | --- | --- |
| Precision =  0.8868 | Recall =  0.8468 | Fmeasure =  0.8664 |  |

(When the code is run, these numbers will not be exact as the tuples are randomly selected to be divided into the training and test sets.)

Accuracy of 83.33% 83.33% of the 520 tuples were correctly identified.  
Error Rate of 16.67% 16.67% of the 520 tuples were incorrectly identified.  
Sensitivity of 84.68% 84.68% of the positive tuples were correctly identified.  
Specificity of 80.95% 80.95% of the negative tuples were correctly identified.  
Precision of 88.68% 88.68% of the 520 tuples were correctly identified as positive.  
Recall of 84.68% 84.68% of the positive 320 tuples were correctly identified as positive   
F Measure of 86.64%

### Evaluation Method: K-Fold Cross Validation (k=10) [(code here)](#_heading=h.3fwokq0)

Training Set: 9/10ths of the 520 tuples.

Test Set: 1/10th of the 520 tuples.

| meanAcc =  0.8846 | meanER =  0.1154 | meanSens =  0.8753 | meanSpecif =  0.9039 |
| --- | --- | --- | --- |
| meanPrec =  0.9327 | meanReca =  0.8753 | meanFmeas =  0.9011 |  |

When the code is run, these numbers will not be exact as the tuples are randomly selected to be divided into the training and test sets.

Mean Accuracy of 88.46% 88.46% of the 520 tuples were correctly identified.  
Mean Error Rate of 11.54% 11.54% of the 520 tuples were incorrectly identified.  
Mean Sensitivity of 87.53% 87.53% of the positive tuples were correctly identified.  
Mean Specificity of 90.39% 90.39% of the negative tuples were correctly identified.  
Mean Precision of 93.27% 93.27% of the 520 tuples were correctly identified as positive.  
Mean Recall of 87.53% 87.53% of the positive 320 tuples were correctly identified as positive   
Mean F Measure of 90.11%

The results of the Holdout Method show that the model is somewhat lacking as it only has an accuracy rate of 83.33%. However, using the K-Fold Cross Validation method and getting the mean accuracy of the model shows that the model is actually around 88.46% accurate. This shows that the Holdout Method’s evaluation is somewhat misleading and that our model actually performs better than the Holdout Method says it does.

This can be seen in the 80.95% Specificity of the Holdout Method compared to the mean Specificity of 90.39% on the K-Fold Cross Validation method’s results. A difference of almost 10% is a big difference.

## 

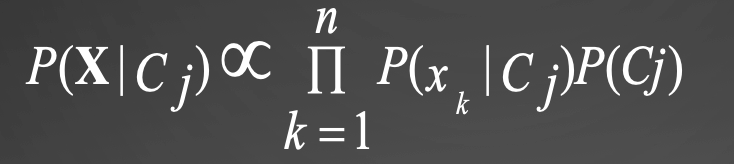
## Classification using the Naive Bayes Classifier [(code here)](#_heading=h.1v1yuxt)

Naïve Bayes Algorithm: Naïve Bayes Algorithm classifies the data set using the Bayes rule of probability. The algorithm is built on the concept that the attributes are unrelated to one another and it treats them separately. The naive Bayes uses the prior probability and the posterior probability of two events. The Bayes Theorem is as follows

*P*(*H* |**X**)= *P*(**X**|*H*)*P*(*H*)/*P*(**X**)

* P(H) (*prior probability*): the initial probability
* P(**X**): the probability that sample data is observed
* P(**X**|H) (likelihood): the probability of observing the sample **X**, given that the hypothesis holds

The classifier is a simple assumption where the values are no dependency relation between the attributes. The naive Bayes classifier is as follows.



In the data set Early detection of diabetes, we used two datasets to do the Naive Bayes. The training set contained two-thirds of the values. The testing set will use the remaining data. The data was randomized by a matlab function to select the data randomly without replacement. We used the gender and all the symptoms to do naive Bayes. as we discussed previously age has an impact on the result of the early diabetes dataset so we included all the values from the second row to the last symptoms.

ClassificationNaiveBayes

PredictorNames: {1×17 cell}

ResponseName: 'Y'

CategoricalPredictors: [2 3 4 5 6 7 8 9 10 11 12 13 14 15 16]

ClassNames: {'Negative' 'Positive'}

ScoreTransform: 'none'

NumObservations: 350

DistributionNames: {1×17 cell}

DistributionParameters: {2×17 cell}

CategoricalLevels: {1×17 cell}

Properties, Methods

Learing top 5 values from finding the most common values

Top 5 height occurring symptoms

P(Polyuria| positive) = 0.496 ––– P(Polyuria| negative) = 0.505

P(Polydispia| positive) = 0.448 ––– P(Polydispia| negative) = 0.552

P(Polyphagia| positive) = 0.456 ––– P(Polyphagia| negative) = 0.544

P(Sudden weight loss| positive) = 0.417 ––– P(Sudden weight loss| negative) = 0.583

P(Partial Paresis | positive) = 0.431 ––– P(Partial Paresis| negative) = 0.509

P(Weakness | positive) = 0.587 ––– P(Weakness| negative) = 0.413

X = (Polyuria, Polydispia, Polyphagia, Sudden weight loss, Partial Paresis)

P(x|Postive) = 0.496\*0.448\* 0.456\* 0.431\*0.587 = 0.026

P(x|Negative) = 0.505\*0.552\* 0.544\* 0.509\* 0.413 = 0.032

P(x|Postive) \* P(Postive) = 0.026 \* 0.62 = 0.016

P(x|Negative) \* P(Negative) = 0.012 \* 0.38 = 0.012

X belong to class positive.

Least 5 height occurring symptoms

P(Obesity| positive) = 0.169 ––– P(Obesity| negative) = 0.831

P(Aloepecia| positive) = 0.344 ––– P(Aloepecia| negative) = 0.656

P(Genital Thrush| positive) = 0.223––– P(Genital Thrush| negative) = 0.777

P(Irritability| positive) = 0.242 ––– P(Irritability| negative) = 0.758

P(muscle stiffness | positive) = 0.375 ––– P(muscleStiffness| negative) = 0.625

X = (Obesity, Aloepecia|, Genital Thrush, muscle Stiffness)

P(x|Postive) = 0.169\*0.344\* 0.223\* 0.242\*0.375 = 0.0011

P(x|Negative) = 0.831\*0.656\* 0.777\* 0.758\* 0.625 = 0.200

P(x|Postive) \* P(Positive) = 0.026 \* 0.62 = 0.0006

P(x|Negative) \* P(Negative) = 0.2 \* 0.38 = 0.076

X belong to class negative.

**Conclusion**

We did a Naive Bayes classifier to the top 5 symptoms we found in the previous test to find the common symptoms for diabetes among the dataset and we found that the test is in fact correct. these symptoms gave us a positive result when we checked using Naive Bayes. On the other hand the least common symptoms the result have shown that the value for the test is negative. This confirms our result of the most common symptoms we did in the previous section. However, in our test, we used the whole dataset for a more accurate result. The Naive Bayes uses all symptoms as well as the gender since we concluded that males and females affect the result of the test in the progress report. In the next part, we will look at an evaluation method to evaluate how accurate our Naive Bayes classifier is for our dataset.

### Evaluation Method: Holdout [(code here)](#_heading=h.2u6wntf)

Training Set: 2/3rds, randomly chosen from the 520 tuples.

Test Set: 1/3rd, from the remaining unchosen 520 tuples.

Results

| Accuracy =  0.8563 | ErrorRate =  0.1437 | Sensitivity =  0.7810 | Specificity =  0.9710 |
| --- | --- | --- | --- |
| Precision =  0.9762 | Recall =  0.7810 | Fmeasure =  0.8677 |  |

When the code is run, these numbers will not be exact as the tuples are randomly selected to be divided into the training and test sets.

Accuracy of 85.63% 85.63% of the 520 tuples were correctly identified.  
Error Rate of 14.37% 14.37% of the 520 tuples were incorrectly identified.  
Sensitivity of 78.10% 78.10% of the positive tuples were correctly identified.  
Specificity of 97.1% 97.1% of the negative tuples were correctly identified.  
Precision of 97.62% 97.62% of the 520 tuples were correctly identified as positive.  
Recall of 78.1% 78.1% of the positive 320 tuples were correctly identified as positive   
F Measure of 86.77%

### Evaluation Method: K-Fold Cross Validation (k=10) [(code here)](#_heading=h.19c6y18)

Training Set: 9/10ths of the 520 tuples.

Test Set: 1/10th of the 520 tuples.

Results

| meanAcc =  0.8750 | meanER =  0.1250 | meanSens =  0.8588 | meanSpecif =  0.8970 |
| --- | --- | --- | --- |
| meanPrec =  0.9327 | meanReca =  0.8588 | meanFmeas =  0.8924 |  |

When the code is run, these numbers will not be exact as the tuples are randomly selected to be divided into the training and test sets.

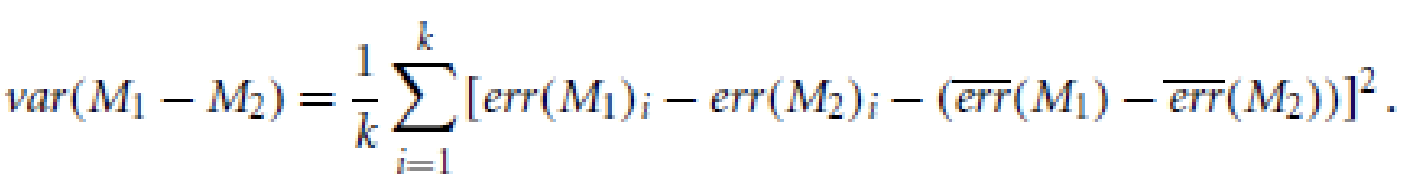
Mean Accuracy of 87.50% 87.50% of the 520 tuples were correctly identified.  
Mean Error Rate of 12.50% 12.50% of the 520 tuples were incorrectly identified.  
Mean Sensitivity of 85.88% 85.88% of the positive tuples were correctly identified.  
Mean Specificity of 89.70% 89.70% of the negative tuples were correctly identified.  
Mean Precision of 93.27% 93.27% of the 520 tuples were correctly identified as positive.  
Mean Recall of 85.88% 85.88% of the positive 320 tuples were correctly identified as positive   
Mean F Measure of 89.24%

The results of the Holdout Method show that the model is somewhat lacking as it only has an accuracy rate of 85.63%. While using the K-Fold Cross Validation method, we got a mean accuracy of 87.50%. With 85.63% and 87.50% as the accuracy rate from two different evaluation methods, this means that the model’s accuracy is confirmed to be below 90%. Which can be a problem when used in the field to diagnose patients with diabetes.

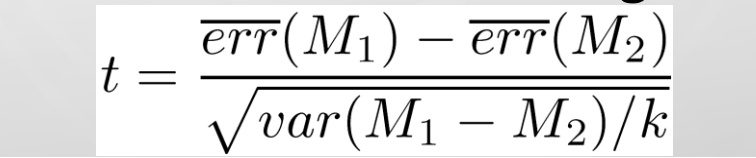
## Comparison of Models

### T-Test Results for Information Gain vs Gini Index Binary Trees:

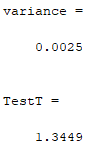
By running the 10 fold cross-validation on both the Information Gain decision tree as well as the Gini Index decision tree we have received the following result. In the Info Gain Decision tree, we have gained the mean error rate of 0.0942. In the Gini Index Decision Tree Model, we got a mean error rate of 0.1154. By inserting those numbers into the variance formula, we can find the variance.



After entering the two mean error rates into the formula along with the error rates of each run (10 runs), we get a variance of 0.0025. With the variance, we plugged the number into T-Test formula, we determined that the two models have a T-Test value of 1.3449



Result:



Degrees of freedom: We have run the K-Cross Fold Validation method 10 times for both models. Therefore our degrees of freedom are 10-1. Df = 9.

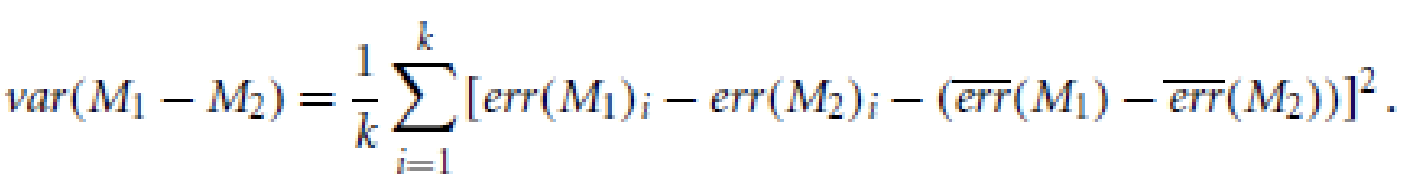
At df=9, and arbitrarily chosen sig=0.10, T-test value 1.3449 is not greater than the t-test value of 1.383 at sig =0.10.

Our models are not significantly different from one another, and therefore we cannot reject our NULL hypothesis that the two models are the same.

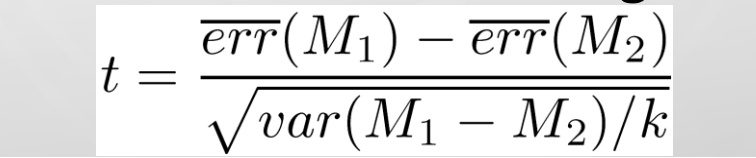
In conclusion, any differences between the two are by chance.

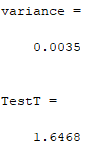
### T-Test Results between Naive Bayesian Classifier and Information Gain Binary Tree

By running the 10 fold cross-validation on both the Information Gain decision tree as well as the Naive Bayesian Classifier we have received the following result. In the Info Gain Decision tree, we have gained the mean error rate of 0.0942. In the Naive Bayesian Classifier, we got a mean error rate of 0.1250. By inserting those numbers into the variance formula, we can find the variance.



After entering the two mean error rates into the formula along with the error rates of each run (10 runs), we get a variance of 0.0035. With the variance, we plugged the number into T-Test formula, we determined that the two models have a T-Test value of 1.6468





Degrees of freedom: We have run the K-Cross Fold Validation method 10 times for both models. Therefore our degrees of freedom is 10-1. Df = 9.

At df=9, and arbitrarily chosen sig=0.10, our T-test value 1.6468 is greater than the t-test statistic value of 1.383 at sig =0.10.

Our models are different from one another as a result from what we can see using the T-Test value, and therefore we can reject our NULL hypothesis that the two models are the same.

However, they are not statistically different between the two as the T-Test value is not greater than z (0.05) = 1.833. The T-value is not greater than z nor less than negative z.

As a result, any difference is by chance.

As we do not have models that are statistically significantly different from one another, we will simply choose the model with the lower error rate, which is the Information Gain Decision Tree.

## Final Conclusion

In conclusion, we have created multiple classification models that are capable of diagnosing a patient with diabetes using a select few given visible attributes that are easily diagnosable. However, our best-performing model is our model using the Information Gain Binary Decision Tree algorithm and it has an accuracy rate of 90.58%. As such, we will look to improve the accuracy of our model as we begin to roll out the model to hospitals for further testing.

| Task | Finished By |
| --- | --- |
| Data Preprocess | Not necessary |
| EDA | All of us. A lot of work has been omitted due to size constraints of the report. |
| Positive-Class with Age Observation | Dylan |
| Reduction in dataset via lowest occurring attributes in tuples with positive class | Moh’d |
| Chi-Square Test | Corey |
| Gender heading | Abderahman Hussein |
| Separation of attributes in regards to age | Prayanshu Singla |
| Naive Bayes Classification | Moh’d |
| Decision Tree Classification: Information Gain | Abderahman |
| Decision Tree Classification: Gini Index | Prayanshu Singla |
| Model Evaluation Methods: Holdout and K-Fold Cross Validation | Corey |
| Model Comparison | Dylan |
| Conclusion | All |

%ALL CODE

%Most code is a Copy-Paste with a few changes.

### %Initial Look at Attributes

%Filepath: C:\Users\(Youraccount)\Documents\MATLAB

%Place the dataset file here.

dataset=readtable('diabetes\_data\_upload.csv','Delimiter',',');

T=dataset;

v1=cell2mat(table2cell(T(:,"Age")))';

histogram(v1,50,'Orientation','Vertical')

xlabel("Age")

ylabel("Instances")

title("Histogram of Age of Observed Subjects")

mean(v1)

median(v1)

std(v1)

prctile(v1,25)

prctile(v1,75)

prctile(v1,50)

menInst=T(find(T.(2)=="Male"),:);

womenInst=T(find(T.(2)=="Female"),:);

men=menInst(find(menInst.(17)=="Positive"),:);

women=womenInst(find(womenInst.(17)=="Positive"),:);

vxx=[height(men);height(menInst)-height(men)];

vxy=[height(women);height(womenInst)-height(women)];

p=pie(vxx)

title("Class(Male)")

pText=findobj(p,'Type','Text');

percentValues=get(pText,'String');

numValues={num2str(height(men));num2str(height(menInst)-height(men))}

txt={'Men with Diabetes: ';'Men without Diabetes: '};

txt2={'(';')'};

strcat(txt2);

combinedtxt=strcat(txt,percentValues,' (',numValues,')');

pText(1).String=combinedtxt(1);

pText(2).String=combinedtxt(2);

v2men=height(menInst);

v2women=height(womenInst);

v2=[v2men;v2women];

p=pie(v2);

pText=findobj(p,'Type','Text');

percentValues=get(pText,'String');

numValues={num2str(v2men);num2str(v2women)}

txt={'Male: ';' Female: '};

txt2={'(';')'};

strcat(txt2);

combinedtxt=strcat(txt,percentValues,' (',numValues,')');

pText(1).String=combinedtxt(1);

pText(2).String=combinedtxt(2);

%If you have a condition called polyuria, it’s because your body makes more pee than normal.

%Adults usually make about 3 liters of urine per day.

%But with polyuria, you could make up to 15 liters per day.

%It's a classic sign of diabetes.

v3=(table2cell(T(:,"Polyuria")))';

v3yes=numel(find(v3=="Yes"));

v3no=numel(find(v3=="No"));

v3=[v3yes;v3no];

p=pie(v3);

pText=findobj(p,'Type','Text');

percentValues=get(pText,'String');

numValues={num2str(v3yes);num2str(v3no)}

txt={' Have Polyuria';' Do not have Polyuria'};

combinedtxt=strcat(percentValues, txt,' (',numValues,')');

pText(1).String=combinedtxt(1);

pText(2).String=combinedtxt(2);

%Number of Men who have Polyuria

menv3yes=height(menInst(find(menInst.(3)=="Yes"),:))

%Number of Men who have Polyuria

menv3no=height(menInst(find(menInst.(3)=="No"),:))

%Number of Women who have Polyuria

fv3yes=height(womenInst(find(womenInst.(3)=="Yes"),:))

%Number of Women who do have Polyuria

fv3no=height(womenInst(find(womenInst.(3)=="No"),:))

v3v=[menv3yes;menv3no;fv3yes;fv3no];

p=pie(v3v);

pText=findobj(p,'Type','Text');

percentValues=get(pText,'String');

numValues={num2str(menv3yes);num2str(menv3no);num2str(fv3yes);num2str(fv3no)}

txt={'Men Have Polyuria';'Men Do not have Polyuria';'Women Have Polyuria';'Women Do not have Polyuria'};

combinedtxt=strcat(percentValues, txt,' (',numValues,')');

pText(1).String=combinedtxt(1);

pText(2).String=combinedtxt(2);

pText(3).String=combinedtxt(3);

pText(4).String=combinedtxt(4);

%Polydipsia is a medical name for the feeling of extreme thirstiness.

%Polydipsia is often linked to urinary conditions that cause you to urinate a lot.

%This can make your body feel a constant need to replace the fluids lost in urination.

%It can also be caused by physical processes that cause you to lose a lot of fluid.

v4=table2cell(T(:,"Polydipsia"));

v4yes=numel(find(v4=="Yes"));

v4no=numel(find(v4=="No"));

v4=[v4yes;v4no];

p=pie(v4);

pText=findobj(p,'Type','Text');

percentValues=get(pText,'String');

numValues={num2str(v4yes);num2str(v4no)}

txt={' Have Polydispia';' Do not have Polydispia'};

combinedtxt=strcat(percentValues, txt,' (',numValues,')');

pText(1).String=combinedtxt(1);

pText(2).String=combinedtxt(2);

v5=table2cell(T(:,"suddenWeightLoss"));

v5yes=numel(find(v5=="Yes"));

v5no=numel(find(v5=="No"));

v5=[v5yes;v5no];

p=pie(v5);

pText=findobj(p,'Type','Text');

percentValues=get(pText,'String');

numValues={num2str(v5yes);num2str(v5no)}

txt={' Have Sudden Weight Loss';' Do not have Sudden Weight Loss'};

combinedtxt=strcat(percentValues, txt,' (',numValues,')');

pText(1).String=combinedtxt(1);

pText(2).String=combinedtxt(2);

%Weakness in body. Self-explanatory.

v6=table2cell(T(:,"weakness"));

v6yes=numel(find(v6=="Yes"));

v6no=numel(find(v6=="No"));

v6=[v6yes;v6no];

p=pie(v6);

pText=findobj(p,'Type','Text');

percentValues=get(pText,'String');

numValues={num2str(v6yes);num2str(v6no)}

txt={' Have Weakness';' Do not have Weakness'};

combinedtxt=strcat(percentValues, txt,' (',numValues,')');

pText(1).String=combinedtxt(1);

pText(2).String=combinedtxt(2);

%Polyphagia is the excessive consumption of food. Also known as hyperphagia

%it is the medical term for excessive or extreme hunger. It is a possible

%sign of diabetes. However there are other conditions that may cause

%polyphagia.

v7=table2cell(T(:,"Polyphagia"));

v7yes=numel(find(v7=="Yes"));

v7no=numel(find(v7=="No"));

v7=[v7yes;v7no];

p=pie(v7);

pText=findobj(p,'Type','Text');

percentValues=get(pText,'String');

numValues={num2str(v7yes);num2str(v7no)}

txt={' Have Polyhagia';' Do not have Polyphagia'};

combinedtxt=strcat(percentValues, txt,' (',numValues,')');

pText(1).String=combinedtxt(1);

pText(2).String=combinedtxt(2);

%Genital Thrush. It is an yeast infection on the genitals. More common in

%women than men. This is considered a possible sign for diabetes due to the

%effects diabetes has on the immune system.

v8=table2cell(T(:,"GenitalThrush"));

v8yes=numel(find(v8=="Yes"));

v8no=numel(find(v8=="No"));

v8=[v8yes;v8no];

p=pie(v8);

pText=findobj(p,'Type','Text');

percentValues=get(pText,'String');

numValues={num2str(v8yes);num2str(v8no)}

txt={' Have GenitalThrush';' Do not have GenitalThrush'};

combinedtxt=strcat(percentValues, txt,' (',numValues,')');

pText(1).String=combinedtxt(1);

pText(2).String=combinedtxt(2);

%Visual Blurring: A condition in which high blood sugar causes the lens of the eye to swell.

%This is a possible symptom of diabetes as type 2 diabetes causes high blood

% sugar due to the inability to use insulin properly.

v9=table2cell(T(:,"visualBlurring"));

v9yes=numel(find(v9=="Yes"));

v9no=numel(find(v9=="No"));

v9=[v9yes;v9no];

p=pie(v9);

pText=findobj(p,'Type','Text');

percentValues=get(pText,'String');

numValues={num2str(v9yes);num2str(v9no)}

txt={' Have Visual Blurring';' Do not have Visual Blurring'};

combinedtxt=strcat(percentValues, txt,' (',numValues,')');

pText(1).String=combinedtxt(1);

pText(2).String=combinedtxt(2);

%Itching.

v10=table2cell(T(:,10));

v10yes=numel(find(v10=="Yes"));

v10no=numel(find(v10=="No"));

v10=[v10yes;v10no];

p=pie(v10);

pText=findobj(p,'Type','Text');

percentValues=get(pText,'String');

numValues={num2str(v10yes);num2str(v10no)}

txt={' Have Itching';' Do not have Itching'};

combinedtxt=strcat(percentValues, txt,' (',numValues,')');

pText(1).String=combinedtxt(1);

pText(2).String=combinedtxt(2);

%Irritability

v11=table2cell(T(:,11));

v11yes=numel(find(v11=="Yes"));

v11no=numel(find(v11=="No"));

v11=[v11yes;v11no];

p=pie(v11);

pText=findobj(p,'Type','Text');

percentValues=get(pText,'String');

numValues={num2str(v11yes);num2str(v11no)}

txt={' Have Irritability';' Do not have Irritability'};

combinedtxt=strcat(percentValues, txt,' (',numValues,')');

pText(1).String=combinedtxt(1);

pText(2).String=combinedtxt(2);

%Delayed Healing

v12=table2cell(T(:,12));

v12yes=numel(find(v12=="Yes"));

v12no=numel(find(v12=="No"));

v12=[v12yes;v12no];

p=pie(v12);

pText=findobj(p,'Type','Text');

percentValues=get(pText,'String');

numValues={num2str(v12yes);num2str(v12no)}

txt={' Have Delayed Healing';' Do not have Delayed Healing'};

combinedtxt=strcat(percentValues, txt,' (',numValues,')');

pText(1).String=combinedtxt(1);

pText(2).String=combinedtxt(2);

%Partial Paresis

v13=table2cell(T(:,13));

v13yes=numel(find(v13=="Yes"));

v13no=numel(find(v13=="No"));

v13=[v13yes;v13no];

p=pie(v13);

pText=findobj(p,'Type','Text');

percentValues=get(pText,'String');

numValues={num2str(v13yes);num2str(v13no)}

txt={' Have Partial Paresis';' Do not have Partial Paresis'};

combinedtxt=strcat(percentValues, txt,' (',numValues,')');

pText(1).String=combinedtxt(1);

pText(2).String=combinedtxt(2);

%Muscle Stiffness

v14=table2cell(T(:,14));

v14yes=numel(find(v14=="Yes"));

v14no=numel(find(v14=="No"));

v14=[v14yes;v14no];

p=pie(v14);

pText=findobj(p,'Type','Text');

percentValues=get(pText,'String');

numValues={num2str(v14yes);num2str(v14no)}

txt={' Have Muscle Stiffness';' Do not have Muscle Stiffness'};

combinedtxt=strcat(percentValues, txt,' (',numValues,')');

pText(1).String=combinedtxt(1);

pText(2).String=combinedtxt(2);

%Aloepecia

v15=table2cell(T(:,15));

v15yes=numel(find(v15=="Yes"));

v15no=numel(find(v15=="No"));

v15=[v15yes;v15no];

p=pie(v15);

pText=findobj(p,'Type','Text');

percentValues=get(pText,'String');

numValues={num2str(v15yes);num2str(v15no)}

txt={' Have Aloepecia';' Do not have Aloepecia'};

combinedtxt=strcat(percentValues, txt,' (',numValues,')');

pText(1).String=combinedtxt(1);

pText(2).String=combinedtxt(2);

%Obesity

v16=table2cell(T(:,16));

v16yes=numel(find(v16=="Yes"));

v16no=numel(find(v16=="No"));

v16=[v16yes;v16no];

p=pie(v16);

pText=findobj(p,'Type','Text');

percentValues=get(pText,'String');

numValues={num2str(v16yes);num2str(v16no)}

txt={' Have Obesity';' Do not have Obesity'};

combinedtxt=strcat(percentValues, txt,' (',numValues,')');

pText(1).String=combinedtxt(1);

pText(2).String=combinedtxt(2);

%Class

v17=table2cell(T(:,17));

v17yes=numel(find(v17=="Positive"));

v17no=numel(find(v17=="Negative"));

v17=[v17yes;v17no];

p=pie(v17);

pText=findobj(p,'Type','Text');

percentValues=get(pText,'String');

numValues={num2str(v17yes);num2str(v17no)}

txt={' Have Diabetes';' Do not have Diabetes'};

combinedtxt=strcat(percentValues, txt,' (',numValues,')');

pText(1).String=combinedtxt(1);

pText(2).String=combinedtxt(2);

### 

### 

### 

### %Age Correlation

%Filepath: C:\Users\(Youraccount)\Documents\MATLAB

v3no=readtable('diabetes\_data\_upload.csv','Delimiter',',');

T= v3no;

T2 = T(T.class == "Positive", :); %select only those rows that are Positive and all columns

T3 = T(T.class == "Negative", :); %select only those rows that are Negative and all columns

v1=cell2mat(table2cell(T2(:,"Age")));

v2=cell2mat(table2cell(T3(:,"Age")));

hold on

histogram(v1);

histogram(v2);

hold off

xlabel("Age")

xticklabels

ylabel("Instances")

title("Histogram of Age of Observed Subjects (Together)")

### %Finding the most common symptoms

% class

v17=table2cell(T(:,"class"));

v17yes=numel(find(v17=="1"));

v17no=numel(find(v17=="0"));

v17=[v17yes;v17no];

p=pie(v17);

pText=findobj(p,'Type','Text');

percentValues=get(pText,'String');

numValues={num2str(v17yes);num2str(v17no)}

txt={' Positives';' Negatives'};

combinedtxt=strcat(percentValues, txt,' (',numValues,')');

pText(1).String=combinedtxt(1);

pText(2).String=combinedtxt(2);

title("Class");

% Counts = nnz(((T.class == "1")))

T2 = T(T.class == "1", :); %select only those rows that are REST and all columns

a1=table2cell(T2(:,"Polyuria"));

a1y=numel(find(a1=="Yes"));

a2=table2cell(T2(:,"Polydipsia"));

a2y=numel(find(a2=="Yes"));

a3=table2cell(T2(:,"suddenWeightLoss"));

a3y=numel(find(a3=="1"));

a4=table2cell(T2(:,"weakness"));

a4y=numel(find(a4=="Yes"));

a5=table2cell(T2(:,"Polyphagia"));

a5y=numel(find(a5=="Yes"));

a6=table2cell(T2(:,"GenitalThrush"));

a6y=numel(find(a6=="Yes"));

a7=table2cell(T2(:,"visualBlurring"));

a7y=numel(find(a7=="Yes"));

a8=table2cell(T2(:,"Itching"));

a8y=numel(find(a8=="Yes"));

a9=table2cell(T2(:,"Irritability"));

a9y=numel(find(a9=="Yes"));

a10=table2cell(T2(:,"delayedHealing"));

a10y=numel(find(a10=="Yes"));

a11=table2cell(T2(:,"partialParesis"));

a11y=numel(find(a11=="Yes"));

a12=table2cell(T2(:,"muscleStiffness"));

a12y=numel(find(a12=="Yes"));

a13=table2cell(T2(:,"Alopecia"));

a13y=numel(find(a13=="Yes"));

a14=table2cell(T2(:,"Obesity"));

a14y=numel(find(a14=="Yes"));

Ynames = categorical({'Polyuria','Polydipsia','suddenWeightLoss','Weakness','Polyphagia','GenitalThrush','visualBlurring','Itching','Irritability','delayedHealing', 'partialParesis', 'muscleStiffness', 'Alopecia', 'Obesity'})

Ynames = reordercats (Ynames, {'Polyuria','Polydipsia','suddenWeightLoss','Weakness','Polyphagia','GenitalThrush','visualBlurring','Itching','Irritability','delayedHealing', 'partialParesis', 'muscleStiffness', 'Alopecia', 'Obesity'})

x =[a1y a2y a3y a4y a5y a6y a7y a8y a9y a10y a11y a12y a13y a14y]

bar (Ynames,x )

### 

### %Chi-Square between all attributes except age, gender and the class

%Chi-square test for all attributes except Age, Gender, and the class

%The Chi-Square test of independence for all attributes except Age, Gender,

%and the Class (Diabetes test-result)

T=readtable('diabetes\_data\_upload.csv','Delimiter',',');

x=4

b=13

atrreduct=[]

for x=3:15

for b=x:16

if (x==b)

else

atr1yes=T(find(T.(x)=="Yes"),:);

atr1no=T(find(T.(x)=="No"),:);

atr2yes=T(find(T.(b)=="Yes"),:);

atr2no=T(find(T.(b)=="No"),:);

atr1yatr2y=height(atr1yes(find(atr1yes.(b)=="Yes"),:));

atr1yatr2n=height(atr1yes(find(atr1yes.(b)=="No"),:));

atr1natr2y=height(atr1no(find(atr1no.(b)=="Yes"),:));

atr1natr2n=height(atr1no(find(atr1no.(b)=="No"),:));

atr1CountY=height(atr1yes);

atr1CountN=height(atr1no);

atr2CountY=height(atr2yes);

atr2CountN=height(atr2no);

str="Attribute "+ x + " and Attribute " +b

continTable=[(atr1yatr2y),(atr1natr2y), atr2CountY;

(atr1yatr2n) (atr1natr2n) atr2CountN;

atr1CountY,atr1CountN, 520]

probAtr1=atr1CountY/520;

probAtr2=atr2CountY/520;

nprobAtr1=atr1CountN/520;

nprobAtr2=atr2CountN/520;

expectedValues=[atr1CountY\*probAtr2, atr1CountN\*probAtr2, probAtr2;

atr1CountY\*nprobAtr2, atr1CountN\*nprobAtr2, nprobAtr2;

probAtr1,nprobAtr1, 520]

xsquare=(continTable(1,1)-expectedValues(1,1))^2/expectedValues(1,1) + (continTable(1,2)-expectedValues(1,2))^2/expectedValues(1,2) + (continTable(2,1)-expectedValues(2,1))^2/expectedValues(2,1) + (continTable(2,2)-expectedValues(2,2))^2/expectedValues(2,2)

%df = 1

%Significance level, I choose 1%, so X=6.635

if (6.635^2 < xsquare)

str="We can reject Ho, the hypothesis: the two attributes have no relation with each other." + ...

"Attribute "+ x + " has a relation with Attribute " +b +...

"Thus, I can reduce the dimensionality by removing one of the two attributes"

atrreduct=[atrreduct;("Attribute "+ x + " and Attribute " +b), xsquare];

else

str="We cannot reject Ho, the hypothesis: the two attributes have no relation with each other."

end

end

end

end

atrreduct

### %Chi-Square with an attribute and class

T=readtable('diabetes\_data\_upload.csv','Delimiter',',');

x=3

b=17

atrreduct=[]

for x=3:16

atr1yes=T(find(T.(x)=="Yes"),:);

atr1no=T(find(T.(x)=="No"),:);

atr2yes=T(find(T.(b)=="Positive"),:);

atr2no=T(find(T.(b)=="Negative"),:);

atr1yatr2y=height(atr1yes(find(atr1yes.(b)=="Positive"),:));

atr1yatr2n=height(atr1yes(find(atr1yes.(b)=="Negative"),:));

atr1natr2y=height(atr1no(find(atr1no.(b)=="Positive"),:));

atr1natr2n=height(atr1no(find(atr1no.(b)=="Negative"),:));

atr1CountY=height(atr1yes);

atr1CountN=height(atr1no);

atr2CountY=height(atr2yes);

atr2CountN=height(atr2no);

str="Attribute "+ x + " and Attribute " +b

continTable=[(atr1yatr2y),(atr1natr2y), atr2CountY;

(atr1yatr2n) (atr1natr2n) atr2CountN;

atr1CountY,atr1CountN, 520]

probAtr1=atr1CountY/520;

probAtr2=atr2CountY/520;

nprobAtr1=atr1CountN/520;

nprobAtr2=atr2CountN/520;

expectedValues=[atr1CountY\*probAtr2, atr1CountN\*probAtr2, probAtr2;

atr1CountY\*nprobAtr2, atr1CountN\*nprobAtr2, nprobAtr2;

probAtr1,nprobAtr1, 520]

xsquare=(continTable(1,1)-expectedValues(1,1))^2/expectedValues(1,1) + (continTable(1,2)-expectedValues(1,2))^2/expectedValues(1,2) + (continTable(2,1)-expectedValues(2,1))^2/expectedValues(2,1) + (continTable(2,2)-expectedValues(2,2))^2/expectedValues(2,2)

%df = 1

%Significance level, I choose 1%, so X=6.635

if (6.635^2 < xsquare)

str="We can reject Ho, the hypothesis: the two attributes have no relation with each other." + ...

"Attribute "+ x + " has a relation with Attribute " +b +...

"Thus, I can reduce the dimensionality by removing one of the two attributes"

atrreduct=[atrreduct;("Attribute "+ x + " and Attribute " +b), xsquare];

else

str="We cannot reject Ho, the hypothesis: the two attributes have no relation with each other."

end

end

atrreduct

## %Information Gain Binary Decision Tree

### %Initial Code

%% InfoGain decision Tree

dataset=readtable('diabetes\_data\_upload.csv','Delimiter',',');

f = dataset;

%creating a random permutation of our dataset

k = randperm(size(f,1));

%using two thirds for our training set

trainSet = f(k(1:347),:);

%Using one third for our test set

testSet = f(k(348:520),:);

%Calculating entropy of training set

info = trainSet(strcmp(trainSet.class, 'Positive'),:);

info2 = trainSet(strcmp(trainSet.class, 'Negative'),:);

%Calculating the info gain on splitting on each attribute, using matlab to get the

%numbers needed

polyuriaYes = trainSet(strcmp(trainSet.Polyuria, 'Yes'),:);

polyuria1 = polyuriaYes(strcmp(polyuriaYes.class, 'Positive'),:);

polyuria2 = polyuriaYes(strcmp(polyuriaYes.class, 'Negative'),:);

polyuriaNo = trainSet(strcmp(trainSet.Polyuria, 'No'),:);

polyuria3 = polyuriaNo(strcmp(polyuriaNo.class, 'Positive'),:);

polyuria4 = polyuriaNo(strcmp(polyuriaNo.class, 'Negative'),:);

polydipsiaYes = trainSet(strcmp(trainSet.Polydipsia, 'Yes'),:);

polydipsia1 = polydipsiaYes(strcmp(polydipsiaYes.class, 'Positive'),:);

polydipsia2 = polydipsiaYes(strcmp(polydipsiaYes.class, 'Negative'),:);

polydipsiaNo = trainSet(strcmp(trainSet.Polydipsia, 'No'),:);

polydipsia3 = polydipsiaNo(strcmp(polydipsiaNo.class, 'Positive'),:);

polydipsia4 = polydipsiaNo(strcmp(polydipsiaNo.class, 'Negative'),:);

suddenWeightLossYes = trainSet(strcmp(trainSet.suddenWeightLoss, 'Yes'),:);

suddenWeightLoss1 = suddenWeightLossYes(strcmp(suddenWeightLossYes.class, 'Positive'),:);

suddenWeightLoss2 = suddenWeightLossYes(strcmp(suddenWeightLossYes.class, 'Negative'),:);

suddenWeightLossNo = trainSet(strcmp(trainSet.suddenWeightLoss, 'No'),:);

suddenWeightLoss3 = suddenWeightLossNo(strcmp(suddenWeightLossNo.class, 'Positive'),:);

suddenWeightLoss4 = suddenWeightLossNo(strcmp(suddenWeightLossNo.class, 'Negative'),:);

PolyphagiaYes = trainSet(strcmp(trainSet.Polyphagia, 'Yes'),:);

Polyphagia1 = PolyphagiaYes(strcmp(PolyphagiaYes.class, 'Positive'),:);

Polyphagia2 = PolyphagiaYes(strcmp(PolyphagiaYes.class, 'Negative'),:);

PolyphagiaNo = trainSet(strcmp(trainSet.Polyphagia, 'No'),:);

Polyphagia3 = PolyphagiaNo(strcmp(PolyphagiaNo.class, 'Positive'),:);

Polyphagia4 = PolyphagiaNo(strcmp(PolyphagiaNo.class, 'Negative'),:);

ItchingYes = trainSet(strcmp(trainSet.Itching, 'Yes'),:);

Itching1 = ItchingYes(strcmp(ItchingYes.class, 'Positive'),:);

Itching2 = ItchingYes(strcmp(ItchingYes.class, 'Negative'),:);

ItchingNo = trainSet(strcmp(trainSet.Itching, 'No'),:);

Itching3 = ItchingNo(strcmp(ItchingNo.class, 'Positive'),:);

Itching4 = ItchingNo(strcmp(ItchingNo.class, 'Negative'),:);

partialParesisYes = trainSet(strcmp(trainSet.partialParesis, 'Yes'),:);

partialParesis1 = partialParesisYes(strcmp(partialParesisYes.class, 'Positive'),:);

partialParesis2 = partialParesisYes(strcmp(partialParesisYes.class, 'Negative'),:);

partialParesisNo = trainSet(strcmp(trainSet.partialParesis, 'No'),:);

partialParesis3 = partialParesisNo(strcmp(partialParesisNo.class, 'Positive'),:);

partialParesis4 = partialParesisNo(strcmp(partialParesisNo.class, 'Negative'),:);

%After determining Polyuria as the root, the rest of the code works on

%continuing the tree. most of the code is for me to get the numbers needed

%to do the calculation on paper

polyuriaYes = trainSet(strcmp(trainSet.Polyuria, 'Yes'),:);

polyuria1 = polyuriaYes(strcmp(polyuriaYes.class, 'Positive'),:);

polyuria2 = polyuriaYes(strcmp(polyuriaYes.class, 'Negative'),:);

attrb1Yes = polyuriaYes(strcmp(polyuriaYes.Polydipsia, 'Yes'),:);

attrb1A = attrb1Yes(strcmp(attrb1Yes.class, 'Positive'),:);

attrb1B = attrb1Yes(strcmp(attrb1Yes.class, 'Negative'),:);

attrb1No = polyuriaYes(strcmp(polyuriaYes.Polydipsia, 'No'),:);

attrb1C = attrb1No(strcmp(attrb1No.class, 'Positive'),:);

attrb1D = attrb1No(strcmp(attrb1No.class, 'Negative'),:);

attrb2Yes = polyuriaYes(strcmp(polyuriaYes.suddenWeightLoss, 'Yes'),:);

attrb2A = attrb2Yes(strcmp(attrb2Yes.class, 'Positive'),:);

attrb2B = attrb2Yes(strcmp(attrb2Yes.class, 'Negative'),:);

attrb2No = polyuriaYes(strcmp(polyuriaYes.suddenWeightLoss, 'No'),:);

attrb2C = attrb2No(strcmp(attrb2No.class, 'Positive'),:);

attrb2D = attrb2No(strcmp(attrb2No.class, 'Negative'),:);

attrb3Yes = polyuriaYes(strcmp(polyuriaYes.Polyphagia, 'Yes'),:);

attrb3A = attrb3Yes(strcmp(attrb3Yes.class, 'Positive'),:);

attrb3B = attrb3Yes(strcmp(attrb3Yes.class, 'Negative'),:);

attrb3No = polyuriaYes(strcmp(polyuriaYes.Polyphagia, 'No'),:);

attrb3C = attrb3No(strcmp(attrb3No.class, 'Positive'),:);

attrb3D = attrb3No(strcmp(attrb3No.class, 'Negative'),:);

attrb4Yes = polyuriaYes(strcmp(polyuriaYes.Itching, 'Yes'),:);

attrb4A = attrb4Yes(strcmp(attrb4Yes.class, 'Positive'),:);

attrb4B = attrb4Yes(strcmp(attrb4Yes.class, 'Negative'),:);

attrb4No = polyuriaYes(strcmp(polyuriaYes.Itching, 'No'),:);

attrb4C = attrb4No(strcmp(attrb4No.class, 'Positive'),:);

attrb4D = attrb4No(strcmp(attrb4No.class, 'Negative'),:);

attrb5Yes = polyuriaYes(strcmp(polyuriaYes.partialParesis, 'Yes'),:);

attrb5A = attrb5Yes(strcmp(attrb5Yes.class, 'Positive'),:);

attrb5B = attrb5Yes(strcmp(attrb5Yes.class, 'Negative'),:);

attrb5No = polyuriaYes(strcmp(polyuriaYes.partialParesis, 'No'),:);

attrb5C = attrb5No(strcmp(attrb5No.class, 'Positive'),:);

attrb5D = attrb5No(strcmp(attrb5No.class, 'Negative'),:);

% Polyuria Yes and Polydipsia No

var1Yes = attrb1No(strcmp(attrb1No.suddenWeightLoss, 'Yes'),:);

var1A = var1Yes(strcmp(var1Yes.class, 'Positive'),:);

var1B = var1Yes(strcmp(var1Yes.class, 'Negative'),:);

var1No = attrb1No(strcmp(attrb1No.suddenWeightLoss, 'No'),:);

var1C = var1No(strcmp(var1No.class, 'Positive'),:);

var1D = var1No(strcmp(var1No.class, 'Negative'),:);

var2Yes = attrb1No(strcmp(attrb1No.Polyphagia, 'Yes'),:);

var2A = var2Yes(strcmp(var2Yes.class, 'Positive'),:);

var2B = var2Yes(strcmp(var2Yes.class, 'Negative'),:);

var2No = attrb1No(strcmp(attrb1No.Polyphagia, 'No'),:);

var2C = var2No(strcmp(var2No.class, 'Positive'),:);

var2D = var2No(strcmp(var2No.class, 'Negative'),:);

var3Yes = attrb1No(strcmp(attrb1No.partialParesis, 'Yes'),:);

var3A = var3Yes(strcmp(var3Yes.class, 'Positive'),:);

var3B = var3Yes(strcmp(var3Yes.class, 'Negative'),:);

var3No = attrb1No(strcmp(attrb1No.partialParesis, 'No'),:);

var3C = var3No(strcmp(var3No.class, 'Positive'),:);

var3D = var3No(strcmp(var3No.class, 'Negative'),:);

var4Yes = attrb1No(strcmp(attrb1No.Itching, 'Yes'),:);

var4A = var4Yes(strcmp(var4Yes.class, 'Positive'),:);

var4B = var4Yes(strcmp(var4Yes.class, 'Negative'),:);

var4No = attrb1No(strcmp(attrb1No.Itching, 'No'),:);

var4C = var4No(strcmp(var4No.class, 'Positive'),:);

var4D = var4No(strcmp(var4No.class, 'Negative'),:);

%Polyuria yes, polydipsia No, Itching Yes

feat1Yes = var4Yes(strcmp(var4Yes.suddenWeightLoss, 'Yes'),:);

feat1A = feat1Yes(strcmp(feat1Yes.class, 'Positive'),:);

feat1B = feat1Yes(strcmp(feat1Yes.class, 'Negative'),:);

feat1No = var4Yes(strcmp(var4Yes.suddenWeightLoss, 'No'),:);

feat1C = feat1No(strcmp(feat1No.class, 'Positive'),:);

feat1D = feat1No(strcmp(feat1No.class, 'Negative'),:);

feat2Yes = var4Yes(strcmp(var4Yes.Polyphagia, 'Yes'),:);

feat2A = feat2Yes(strcmp(feat2Yes.class, 'Positive'),:);

feat2B = feat2Yes(strcmp(feat2Yes.class, 'Negative'),:);

feat2No = var4Yes(strcmp(var4Yes.Polyphagia, 'No'),:);

feat2C = feat2No(strcmp(feat2No.class, 'Positive'),:);

feat2D = feat2No(strcmp(feat2No.class, 'Negative'),:);

feat3Yes = var4Yes(strcmp(var4Yes.partialParesis, 'Yes'),:);

feat3A = feat3Yes(strcmp(feat3Yes.class, 'Positive'),:);

feat3B = feat3Yes(strcmp(feat3Yes.class, 'Negative'),:);

feat3No = var4Yes(strcmp(var4Yes.partialParesis, 'No'),:);

feat3C = feat3No(strcmp(feat3No.class, 'Positive'),:);

feat3D = feat3No(strcmp(feat3No.class, 'Negative'),:);

%Polyphagia yes

char1Yes = feat2Yes(strcmp(feat2Yes.partialParesis, 'Yes'),:);

char1No = feat2Yes(strcmp(feat2Yes.partialParesis, 'No'),:);

char2Yes= feat2Yes(strcmp(feat2Yes.suddenWeightLoss, 'Yes'),:);

char2No= feat2Yes(strcmp(feat2Yes.suddenWeightLoss, 'No'),:);

%Polyphagia No and partial paresis No

ky1Yes = feat2No(strcmp(feat2No.suddenWeightLoss, 'Yes'),:);

ky1No = feat2No(strcmp(feat2No.suddenWeightLoss, 'No'),:);

ky2Yes = feat2No(strcmp(feat2No.partialParesis, 'Yes'),:);

ky2No = feat2No(strcmp(feat2No.partialParesis, 'No'),:);

ky3Yes = ky2No(strcmp(ky2No.suddenWeightLoss, 'Yes'),:);

ky3NO = ky2No(strcmp(ky2No.suddenWeightLoss, 'No'),:);

%partialparesis yes

name1Yes = char1Yes(strcmp(char1Yes.suddenWeightLoss, 'Yes'),:);

name1No = char1Yes(strcmp(char1Yes.suddenWeightLoss, 'No'),:);

%partialParesis No

name2Yes = char1No(strcmp(char1No.suddenWeightLoss, 'Yes'),:);

name2No = char1No(strcmp(char1No.suddenWeightLoss, 'No'),:);

%END OF LEFT BRANCH

%Constructing the decision tree and comparing it to my results. My tree is

%the same as the built-in tree. Probably because the algorithm uses the

%same tactic as the built-in tree.

polyuriaone = table2array(trainSet(:,3));

polydipsiatwo = table2array(trainSet(:,4));

sWLthree = table2array(trainSet(:,5));

polyphagiafour = table2array(trainSet(:,7));

itchingfive = table2array(trainSet(:,10));

partialParalesissix = table2array(trainSet(:,13));

classseven = table2array(trainSet(:,17));

T = table(polyuriaone, polydipsiatwo, sWLthree, polyphagiafour, itchingfive, partialParalesissix, classseven);

decisionTree = fitctree(T,'classseven');

view(decisionTree)

view(decisionTree, 'Mode', 'Graph')

### %Holdout Method for Information Gain

T=dataset;

testset=T;

trainset=[];

x=[1:520]; %Generate vector with numbers 1 to 520

toremove=[];

for b=1:346 %Run for 2/3rds of the original dataset

r=round((length(x)-1)\*rand()+1); %Randomize number of the length of x vector

trainset=[trainset;T([x(r)],:)]; %Add T tuple at row x vector value at r

toremove=[toremove;x(r)]; %Store x(r) to vector

x(r)=[]; %Remove x value at position r from vector

end

testset([toremove],:)=[]; %Remove all the x(r) values from the test set to form 1/3rd test set

trainsetx=trainset(:,[3,4,5,7,10,13]); %Create trainsetx table with attributes 3, 4, 5, 7, 10, 13, %these will be used to train the model

testsetx=testset(:,[3,4,5,7,10,13]);

decisionTree = fitctree(trainsetx,trainset.class); %fit decision tree

w = predict(decisionTree, testsetx); %predict/test using test set

%convert values to numbers

w(find(w=="Positive"))={'1'};

w(find(w=="Negative"))={'0'};

testset(find(testset.(17)=="Positive"),"class")={'1'};

testset(find(testset.(17)=="Negative"),"class")={'0'};

TP=0;

FP=0;

TN=0;

FN=0;

for i=1:174 %run loop for the test set

comparew=cell2mat(w(i));

comparet=cell2mat(testset.class(i));

%compare and count results for True Positive, False Positive, True Negative, False Negative

if (comparew==comparet && (comparew=='1'))

TP=TP+1;

elseif (comparew==comparet && (comparew=='0'))

TN=TN+1;

else

if (comparew=='1')

FP=FP+1;

elseif (comparew=='0')

FN=FN+1;

end

end

end

%Calculate metrics

Accuracy = (TP+TN)/(174)

ErrorRate=(FP+FN)/(174)

Sensitivity=TP/(TP+FN)

Specificity=TN/(TN+FP)

Precision=TP/(TP+FP)

Recall=TP/(TP+FN)

Fmeasure=(2\*Precision\*Recall)/(Precision+Recall)

### %K=10 Fold Cross Validation Method for Information Gain

dataset = readtable('diabetes\_data\_upload.csv');

T=dataset;

Acc=[];

ER=[];

Sens=[];

Specif=[];

Prec=[];

Reca=[];

Fmeas=[];

%Randomize the

T = randperm(size(T,1));

T = dataset(T,:);

for b=1:10

trainset=[];

testset=[];

for x=1:520

trainset=[trainset;T(x,:)];

if ((x>=(52\*(b-1)+1) && (x<=(52\*b))))

testset=[testset;T(x,:)];

end

end

trainsetx=trainset(:,[3,4,5,7,10,13]);

testsetx=testset(:,[3,4,5,7,10,13]);

decisionTree = fitctree(trainsetx,trainset.class);

w = predict(decisionTree, testsetx);

w(find(w=="Positive"))={'1'};

w(find(w=="Negative"))={'0'};

testset(find(testset.(17)=="Positive"),"class")={'1'};

testset(find(testset.(17)=="Negative"),"class")={'0'};

TP=0;

FP=0;

TN=0;

FN=0;

for i=1:(520\*.1)

comparew=cell2mat(w(i));

comparet=cell2mat(testset.class(i));

if (comparew==comparet && (comparew=='1'))

TP=TP+1;

elseif (comparew==comparet && (comparew=='0'))

TN=TN+1;

else

if (comparew=='1')

FP=FP+1;

elseif (comparew=='0')

FN=FN+1;

end

end

end

Accuracy = (TP+TN)/(520\*0.1);

ErrorRate=(FP+FN)/(520\*0.1);

Sensitivity=TP/(TP+FN);

Specificity=TN/(TN+FP);

if (isnan(Specificity)==1)

Specificity=1;

end

Precision=TP/(TP+FP);

Recall=TP/(TP+FN);

Fmeasure=(2\*Precision\*Recall)/(Precision+Recall);

Acc=[Acc;Accuracy];

ER=[ER;ErrorRate];

Sens=[Sens;Sensitivity];

Specif=[Specif;Specificity];

Prec=[Prec;Precision];

Reca=[Reca;Recall];

Fmeas=[Fmeas;Fmeasure];

end

view(decisionTree, 'Mode', 'Graph')

meanAcc=mean(Acc)

meanER=mean(ER)

meanSens=mean(Sens)

meanSpecif=mean(Specif)

meanPrec=mean(Prec)

meanReca=mean(Reca)

meanFmeas=mean(Fmeas)

## %Gini Index Binary Decision Tree

### %Initial Code

% Read data from csv

data = readtable("diabetes\_data\_upload.csv");

%-------------------------------------------------------------------------

% Randomly divide data into training

%-------------------------------------------------------------------------

% Get data dimension (rows and columns)

[m,n] = size(data) ;

% Percentage of training data to get 347 samples in training set

P = 0.668;

% Generate random index for given number of rows

idx = randperm(m);

% Select Training samples based on set percentage

Training = data(idx(1:round(P\*m)),:);

% set predictors names

inputNames = {'Polyuria', 'Polydipsia', 'suddenWeightLoss', 'Itching', 'partialParesis'};

% Convert Training data into training inputs and outputs

trainX = Training(:, inputNames);

trainY = Training.class;

%-------------------------------------------------------------------------

% Train decision classifier

%-------------------------------------------------------------------------

% Create the decision tree classifer

classificationTree = fitctree(...

trainX, ...

trainY, ...

'SplitCriterion', 'gdi', ... % Gini index

'MaxNumSplits', 100, ...

'Surrogate', 'off', ...

'ClassNames', categorical({'Negative'; 'Positive'}))

% Create the result struct with predict function

predictorExtractionFcn = @(t) t(:, inputNames);

treePredictFcn = @(x) predict(classificationTree, x)

trainedClassifier.predictFcn = @(x) treePredictFcn(predictorExtractionFcn(x));

% Add additional fields to the result struct

trainedClassifier.RequiredVariables = inputNames;

trainedClassifier.ClassificationTree = classificationTree;

view(classificationTree, 'mode','graph'); % view decision tree

### %Holdout Method for Gini Index Tree

dataset = readtable('diabetes\_data\_upload.csv');

T=dataset;

testset=T;

trainset=[];

x=[1:520];

toremove=[];

for b=1:346

r=round((length(x)-1)\*rand()+1);

trainset=[trainset;T([x(r)],:)];

toremove=[toremove;x(r)];

x(r)=[];

end

testset([toremove],:)=[];

trainsetx=trainset(:,[3,4,5,10,13]);

testsetx=testset(:,[3,4,5,10,13]);

classificationTree = fitctree(...

trainsetx, ...

trainset.class, ...

'SplitCriterion', 'gdi', ... % Gini index

'MaxNumSplits', 100, ...

'Surrogate', 'off', ...

'ClassNames', categorical({'Negative'; 'Positive'}))

w = predict(classificationTree, testsetx);

w(find(w=="Positive"))={'1'};

w(find(w=="Negative"))={'0'};

testset(find(testset.(17)=="Positive"),"class")={'1'};

testset(find(testset.(17)=="Negative"),"class")={'0'};

TP=0;

FP=0;

TN=0;

FN=0;

for i=1:174

comparew=(w(i));

comparet=cell2mat(testset.class(i));

if (comparew==comparet && (comparew=='1'))

TP=TP+1;

elseif (comparew==comparet && (comparew=='0'))

TN=TN+1;

else

if (comparew=='1')

FP=FP+1;

elseif (comparew=='0')

FN=FN+1;

end

end

end

Accuracy = (TP+TN)/(174);

ErrorRate=(FP+FN)/(174);

Sensitivity=TP/(TP+FN);

Specificity=TN/(TN+FP);

Precision=TP/(TP+FP);

Recall=TP/(TP+FN);

Fmeasure=(2\*Precision\*Recall)/(Precision+Recall);

### %K=10 Fold Cross Validation for Gini Index Tree

dataset = readtable('diabetes\_data\_upload.csv');

T=dataset;

Acc=[];

ER=[];

Sens=[];

Specif=[];

Prec=[];

Reca=[];

Fmeas=[];

for b=1:10

trainset=[];

testset=[];

for x=1:520

trainset=[trainset;T(x,:)];

if ((x>=(52\*(b-1)+1) && (x<=(52\*b))))

testset=[testset;T(x,:)];

end

end

trainsetx=trainset(:,[3,4,5,10,13]);

testsetx=testset(:,[3,4,5,10,13]);

classificationTree = fitctree(...

trainsetx, ...

trainset.class, ...

'SplitCriterion', 'gdi', ... % Gini index

'MaxNumSplits', 100, ...

'Surrogate', 'off', ...

'ClassNames', categorical({'Negative'; 'Positive'}))

w = predict(classificationTree, testsetx);

w(find(w=="Positive"))={'1'};

w(find(w=="Negative"))={'0'};

testset(find(testset.(17)=="Positive"),"class")={'1'};

testset(find(testset.(17)=="Negative"),"class")={'0'};

TP=0;

FP=0;

TN=0;

FN=0;

for i=1:(520\*.1)

comparew=(w(i));

comparet=cell2mat(testset.class(i));

if (comparew==comparet && (comparew=='1'))

TP=TP+1;

elseif (comparew==comparet && (comparew=='0'))

TN=TN+1;

else

if (comparew=='1')

FP=FP+1;

elseif (comparew=='0')

FN=FN+1;

end

end

end

Accuracy = (TP+TN)/(520\*0.1);

ErrorRate=(FP+FN)/(520\*0.1);

Sensitivity=TP/(TP+FN);

Specificity=TN/(TN+FP);

if (isnan(Specificity)==1)

Specificity=1;

end

Precision=TP/(TP+FP);

Recall=TP/(TP+FN);

Fmeasure=(2\*Precision\*Recall)/(Precision+Recall);

Acc=[Acc;Accuracy];

ER=[ER;ErrorRate];

Sens=[Sens;Sensitivity];

Specif=[Specif;Specificity];

Prec=[Prec;Precision];

Reca=[Reca;Recall];

Fmeas=[Fmeas;Fmeasure];

end

meanAcc=mean(Acc)

meanER=mean(ER)

meanSens=mean(Sens)

meanSpecif=mean(Specif)

meanPrec=mean(Prec)

meanReca=mean(Reca)

meanFmeas=mean(Fmeas)

## %Naive Bayes Classifier

### %Code for to convert text to binary values

dataset=readtable('diabetes\_data\_upload.csv','Delimiter',',');

T=dataset;

T(find(T.(3)=="Yes"),"Polyuria")={'1'};

T(find(T.(3)=="No"),"Polyuria")={'0'};

T(find(T.(4)=="Yes"),"Polydipsia")={'1'};

T(find(T.(4)=="No"),"Polydipsia")={'0'};

T(find(T.(5)=="Yes"),"suddenWeightLoss")={'1'};

T(find(T.(5)=="No"),"suddenWeightLoss")={'0'};

T(find(T.(6)=="Yes"),"weakness")={'1'};

T(find(T.(6)=="No"),"weakness")={'0'};

T(find(T.(7)=="Yes"),"Polyphagia")={'1'};

T(find(T.(7)=="No"),"Polyphagia")={'0'};

T(find(T.(8)=="Yes"),"GenitalThrush")={'1'};

T(find(T.(8)=="No"),"GenitalThrush")={'0'};

T(find(T.(9)=="Yes"),"visualBlurring")={'1'};

T(find(T.(9)=="No"),"visualBlurring")={'0'};

T(find(T.(10)=="Yes"),"Itching")={'1'};

T(find(T.(10)=="No"),"Itching")={'0'};

T(find(T.(11)=="Yes"),"Irritability")={'1'};

T(find(T.(11)=="No"),"Irritability")={'0'};

T(find(T.(12)=="Yes"),"delayedHealing")={'1'};

T(find(T.(12)=="No"),"delayedHealing")={'0'};

T(find(T.(13)=="Yes"),"partialParesis")={'1'};

T(find(T.(13)=="No"),"partialParesis")={'0'};

T(find(T.(14)=="Yes"),"muscleStiffness")={'1'};

T(find(T.(14)=="No"),"muscleStiffness")={'0'};

T(find(T.(15)=="Yes"),"Alopecia")={'1'};

T(find(T.(15)=="No"),"Alopecia")={'0'};

T(find(T.(16)=="Yes"),"Obesity")={'1'};

T(find(T.(16)=="No"),"Obesity")={'0'};

testset=T;

trainset=[];

x=[1:520];

toremove=[];

for b=1:346

r=round((length(x)-1)\*rand()+1);

trainset=[trainset;T([x(r)],:)];

toremove=[toremove;x(r)];

x(r)=[];

end

testset([toremove],:)=[];

trainsetx=trainset(:,2:16);

testsetx=testset(:,2:16);

m=fitcnb(trainsetx,trainset.class) %Uses the built-in matlab function for naive bayes classifier

w=predict(m,testsetx)

errNB=loss(m,testsetx,testset.class)

disp([num2str(errNB),'-Naive Bayes loss']) %find loss in the model

### %Holdout Method for Naive Bayes Classifier

dataset = readtable('diabetes\_data\_upload.csv');

T=dataset;

testset=T;

trainset=[];

x=[1:520];

toremove=[];

for b=1:346

r=round((length(x)-1)\*rand()+1);

trainset=[trainset;T([x(r)],:)];

toremove=[toremove;x(r)];

x(r)=[];

end

testset([toremove],:)=[];

trainsetx=trainset(:,2:16);

testsetx=testset(:,2:16);

m=fitcnb(trainsetx,trainset.class)

w=predict(m,testsetx)

errNB=loss(m,testsetx,testset.class)

disp([num2str(errNB),'-Naive Bayes loss']) %find loss in the model

w(find(w=="Positive"))={'1'};

w(find(w=="Negative"))={'0'};

testset(find(testset.(17)=="Positive"),"class")={'1'};

testset(find(testset.(17)=="Negative"),"class")={'0'};

TP=0;

FP=0;

TN=0;

FN=0;

for i=1:174

comparew=cell2mat(w(i));

comparet=cell2mat(testset.class(i));

if (comparew==comparet && (comparew=='1'))

TP=TP+1;

elseif (comparew==comparet && (comparew=='0'))

TN=TN+1;

else

if (comparew=='1')

FP=FP+1;

elseif (comparew=='0')

FN=FN+1;

end

end

end

Accuracy = (TP+TN)/174

ErrorRate=(FP+FN)/174

Sensitivity=TP/(TP+FN)

Specificity=TN/(TN+FP)

Precision=TP/(TP+FP)

Recall=TP/(TP+FN)

Fmeasure=(2\*Precision\*Recall)/(Precision+Recall)

### %K=10 Fold Cross Validation for Naive Bayes Classifier

dataset = readtable('diabetes\_data\_upload.csv');

T=dataset;

Acc=[];

ER=[];

Sens=[];

Specif=[];

Prec=[];

Reca=[];

Fmeas=[];

T = randperm(size(T,1));

T = dataset(T,:);

for b=1:10

trainset=[];

testset=[];

for x=1:520

trainset=[trainset;T(x,:)];

if ((x>=(52\*(b-1)+1) && (x<=(52\*b))))

testset=[testset;T(x,:)];

end

end

trainsetx=trainset(:,2:16);

testsetx=testset(:,2:16);

m=fitcnb(trainsetx,trainset.class);

w=predict(m,testsetx);

w(find(w=="Positive"))={'1'};

w(find(w=="Negative"))={'0'};

testset(find(testset.(17)=="Positive"),"class")={'1'};

testset(find(testset.(17)=="Negative"),"class")={'0'};

TP=0;

FP=0;

TN=0;

FN=0;

for i=1:(520\*.1)

comparew=cell2mat(w(i));

comparet=cell2mat(testset.class(i));

if (comparew==comparet && (comparew=='1'))

TP=TP+1;

elseif (comparew==comparet && (comparew=='0'))

TN=TN+1;

else

if (comparew=='1')

FP=FP+1;

elseif (comparew=='0')

FN=FN+1;

end

end

end

Accuracy = (TP+TN)/(520\*0.1);

ErrorRate=(FP+FN)/(520\*0.1);

Sensitivity=TP/(TP+FN);

Specificity=TN/(TN+FP);

if (isnan(Specificity)==1)

Specificity=1;

end

Precision=TP/(TP+FP);

Recall=TP/(TP+FN);

Fmeasure=(2\*Precision\*Recall)/(Precision+Recall);

Acc=[Acc;Accuracy];

ER=[ER;ErrorRate];

Sens=[Sens;Sensitivity];

Specif=[Specif;Specificity];

Prec=[Prec;Precision];

Reca=[Reca;Recall];

Fmeas=[Fmeas;Fmeasure];

end

meanAcc=mean(Acc)

meanER=mean(ER)

meanSens=mean(Sens)

meanSpecif=mean(Specif)

meanPrec=mean(Prec)

meanReca=mean(Reca)

meanFmeas=mean(Fmeas)

## %T-Test For the Gini Decision Tree and InfoGain Tree %

dataset=readtable('diabetes\_data\_upload.csv','Delimiter',',');

f = dataset;

%creating a random permutation of our dataset

b = randperm(520);

%using two thirds for our training set

trainSet = f(b(1:347),:);

%Using one third for our test set

testSet = f(b(348:520),:);

%Calculating entropy of training set

info = trainSet(strcmp(trainSet.class, 'Positive'),:);

info2 = trainSet(strcmp(trainSet.class, 'Negative'),:);

%Calculating the info gain on splitting on each attribute, using matlab to get the

%numbers needed

polyuriaYes = trainSet(strcmp(trainSet.Polyuria, 'Yes'),:);

polyuria1 = polyuriaYes(strcmp(polyuriaYes.class, 'Positive'),:);

polyuria2 = polyuriaYes(strcmp(polyuriaYes.class, 'Negative'),:);

polyuriaNo = trainSet(strcmp(trainSet.Polyuria, 'No'),:);

polyuria3 = polyuriaNo(strcmp(polyuriaNo.class, 'Positive'),:);

polyuria4 = polyuriaNo(strcmp(polyuriaNo.class, 'Negative'),:);

polydipsiaYes = trainSet(strcmp(trainSet.Polydipsia, 'Yes'),:);

polydipsia1 = polydipsiaYes(strcmp(polydipsiaYes.class, 'Positive'),:);

polydipsia2 = polydipsiaYes(strcmp(polydipsiaYes.class, 'Negative'),:);

polydipsiaNo = trainSet(strcmp(trainSet.Polydipsia, 'No'),:);

polydipsia3 = polydipsiaNo(strcmp(polydipsiaNo.class, 'Positive'),:);

polydipsia4 = polydipsiaNo(strcmp(polydipsiaNo.class, 'Negative'),:);

suddenWeightLossYes = trainSet(strcmp(trainSet.suddenWeightLoss, 'Yes'),:);

suddenWeightLoss1 = suddenWeightLossYes(strcmp(suddenWeightLossYes.class, 'Positive'),:);

suddenWeightLoss2 = suddenWeightLossYes(strcmp(suddenWeightLossYes.class, 'Negative'),:);

suddenWeightLossNo = trainSet(strcmp(trainSet.suddenWeightLoss, 'No'),:);

suddenWeightLoss3 = suddenWeightLossNo(strcmp(suddenWeightLossNo.class, 'Positive'),:);

suddenWeightLoss4 = suddenWeightLossNo(strcmp(suddenWeightLossNo.class, 'Negative'),:);

PolyphagiaYes = trainSet(strcmp(trainSet.Polyphagia, 'Yes'),:);

Polyphagia1 = PolyphagiaYes(strcmp(PolyphagiaYes.class, 'Positive'),:);

Polyphagia2 = PolyphagiaYes(strcmp(PolyphagiaYes.class, 'Negative'),:);

PolyphagiaNo = trainSet(strcmp(trainSet.Polyphagia, 'No'),:);

Polyphagia3 = PolyphagiaNo(strcmp(PolyphagiaNo.class, 'Positive'),:);

Polyphagia4 = PolyphagiaNo(strcmp(PolyphagiaNo.class, 'Negative'),:);

ItchingYes = trainSet(strcmp(trainSet.Itching, 'Yes'),:);

Itching1 = ItchingYes(strcmp(ItchingYes.class, 'Positive'),:);

Itching2 = ItchingYes(strcmp(ItchingYes.class, 'Negative'),:);

ItchingNo = trainSet(strcmp(trainSet.Itching, 'No'),:);

Itching3 = ItchingNo(strcmp(ItchingNo.class, 'Positive'),:);

Itching4 = ItchingNo(strcmp(ItchingNo.class, 'Negative'),:);

partialParesisYes = trainSet(strcmp(trainSet.partialParesis, 'Yes'),:);

partialParesis1 = partialParesisYes(strcmp(partialParesisYes.class, 'Positive'),:);

partialParesis2 = partialParesisYes(strcmp(partialParesisYes.class, 'Negative'),:);

partialParesisNo = trainSet(strcmp(trainSet.partialParesis, 'No'),:);

partialParesis3 = partialParesisNo(strcmp(partialParesisNo.class, 'Positive'),:);

partialParesis4 = partialParesisNo(strcmp(partialParesisNo.class, 'Negative'),:);

%After determining Polyuria as the root, the rest of the code works on

%continuing the tree. most of the code is for me to get the numbers needed

%to do the calculation on paper

polyuriaYes = trainSet(strcmp(trainSet.Polyuria, 'Yes'),:);

polyuria1 = polyuriaYes(strcmp(polyuriaYes.class, 'Positive'),:);

polyuria2 = polyuriaYes(strcmp(polyuriaYes.class, 'Negative'),:);

attrb1Yes = polyuriaYes(strcmp(polyuriaYes.Polydipsia, 'Yes'),:);

attrb1A = attrb1Yes(strcmp(attrb1Yes.class, 'Positive'),:);

attrb1B = attrb1Yes(strcmp(attrb1Yes.class, 'Negative'),:);

attrb1No = polyuriaYes(strcmp(polyuriaYes.Polydipsia, 'No'),:);

attrb1C = attrb1No(strcmp(attrb1No.class, 'Positive'),:);

attrb1D = attrb1No(strcmp(attrb1No.class, 'Negative'),:);

attrb2Yes = polyuriaYes(strcmp(polyuriaYes.suddenWeightLoss, 'Yes'),:);

attrb2A = attrb2Yes(strcmp(attrb2Yes.class, 'Positive'),:);

attrb2B = attrb2Yes(strcmp(attrb2Yes.class, 'Negative'),:);

attrb2No = polyuriaYes(strcmp(polyuriaYes.suddenWeightLoss, 'No'),:);

attrb2C = attrb2No(strcmp(attrb2No.class, 'Positive'),:);

attrb2D = attrb2No(strcmp(attrb2No.class, 'Negative'),:);

attrb3Yes = polyuriaYes(strcmp(polyuriaYes.Polyphagia, 'Yes'),:);

attrb3A = attrb3Yes(strcmp(attrb3Yes.class, 'Positive'),:);

attrb3B = attrb3Yes(strcmp(attrb3Yes.class, 'Negative'),:);

attrb3No = polyuriaYes(strcmp(polyuriaYes.Polyphagia, 'No'),:);

attrb3C = attrb3No(strcmp(attrb3No.class, 'Positive'),:);

attrb3D = attrb3No(strcmp(attrb3No.class, 'Negative'),:);

attrb4Yes = polyuriaYes(strcmp(polyuriaYes.Itching, 'Yes'),:);

attrb4A = attrb4Yes(strcmp(attrb4Yes.class, 'Positive'),:);

attrb4B = attrb4Yes(strcmp(attrb4Yes.class, 'Negative'),:);

attrb4No = polyuriaYes(strcmp(polyuriaYes.Itching, 'No'),:);

attrb4C = attrb4No(strcmp(attrb4No.class, 'Positive'),:);

attrb4D = attrb4No(strcmp(attrb4No.class, 'Negative'),:);

attrb5Yes = polyuriaYes(strcmp(polyuriaYes.partialParesis, 'Yes'),:);

attrb5A = attrb5Yes(strcmp(attrb5Yes.class, 'Positive'),:);

attrb5B = attrb5Yes(strcmp(attrb5Yes.class, 'Negative'),:);

attrb5No = polyuriaYes(strcmp(polyuriaYes.partialParesis, 'No'),:);

attrb5C = attrb5No(strcmp(attrb5No.class, 'Positive'),:);

attrb5D = attrb5No(strcmp(attrb5No.class, 'Negative'),:);

% Polyuria Yes and Polydipsia No

var1Yes = attrb1No(strcmp(attrb1No.suddenWeightLoss, 'Yes'),:);

var1A = var1Yes(strcmp(var1Yes.class, 'Positive'),:);

var1B = var1Yes(strcmp(var1Yes.class, 'Negative'),:);

var1No = attrb1No(strcmp(attrb1No.suddenWeightLoss, 'No'),:);

var1C = var1No(strcmp(var1No.class, 'Positive'),:);

var1D = var1No(strcmp(var1No.class, 'Negative'),:);

var2Yes = attrb1No(strcmp(attrb1No.Polyphagia, 'Yes'),:);

var2A = var2Yes(strcmp(var2Yes.class, 'Positive'),:);

var2B = var2Yes(strcmp(var2Yes.class, 'Negative'),:);

var2No = attrb1No(strcmp(attrb1No.Polyphagia, 'No'),:);

var2C = var2No(strcmp(var2No.class, 'Positive'),:);

var2D = var2No(strcmp(var2No.class, 'Negative'),:);

var3Yes = attrb1No(strcmp(attrb1No.partialParesis, 'Yes'),:);

var3A = var3Yes(strcmp(var3Yes.class, 'Positive'),:);

var3B = var3Yes(strcmp(var3Yes.class, 'Negative'),:);

var3No = attrb1No(strcmp(attrb1No.partialParesis, 'No'),:);

var3C = var3No(strcmp(var3No.class, 'Positive'),:);

var3D = var3No(strcmp(var3No.class, 'Negative'),:);

var4Yes = attrb1No(strcmp(attrb1No.Itching, 'Yes'),:);

var4A = var4Yes(strcmp(var4Yes.class, 'Positive'),:);

var4B = var4Yes(strcmp(var4Yes.class, 'Negative'),:);

var4No = attrb1No(strcmp(attrb1No.Itching, 'No'),:);

var4C = var4No(strcmp(var4No.class, 'Positive'),:);

var4D = var4No(strcmp(var4No.class, 'Negative'),:);

%Polyuria yes, polydipsia No, Itching Yes

feat1Yes = var4Yes(strcmp(var4Yes.suddenWeightLoss, 'Yes'),:);

feat1A = feat1Yes(strcmp(feat1Yes.class, 'Positive'),:);

feat1B = feat1Yes(strcmp(feat1Yes.class, 'Negative'),:);

feat1No = var4Yes(strcmp(var4Yes.suddenWeightLoss, 'No'),:);

feat1C = feat1No(strcmp(feat1No.class, 'Positive'),:);

feat1D = feat1No(strcmp(feat1No.class, 'Negative'),:);

feat2Yes = var4Yes(strcmp(var4Yes.Polyphagia, 'Yes'),:);

feat2A = feat2Yes(strcmp(feat2Yes.class, 'Positive'),:);

feat2B = feat2Yes(strcmp(feat2Yes.class, 'Negative'),:);

feat2No = var4Yes(strcmp(var4Yes.Polyphagia, 'No'),:);

feat2C = feat2No(strcmp(feat2No.class, 'Positive'),:);

feat2D = feat2No(strcmp(feat2No.class, 'Negative'),:);

feat3Yes = var4Yes(strcmp(var4Yes.partialParesis, 'Yes'),:);

feat3A = feat3Yes(strcmp(feat3Yes.class, 'Positive'),:);

feat3B = feat3Yes(strcmp(feat3Yes.class, 'Negative'),:);

feat3No = var4Yes(strcmp(var4Yes.partialParesis, 'No'),:);

feat3C = feat3No(strcmp(feat3No.class, 'Positive'),:);

feat3D = feat3No(strcmp(feat3No.class, 'Negative'),:);

%Polyphagia yes

char1Yes = feat2Yes(strcmp(feat2Yes.partialParesis, 'Yes'),:);

char1No = feat2Yes(strcmp(feat2Yes.partialParesis, 'No'),:);

char2Yes= feat2Yes(strcmp(feat2Yes.suddenWeightLoss, 'Yes'),:);

char2No= feat2Yes(strcmp(feat2Yes.suddenWeightLoss, 'No'),:);

%Polyphagia No and partial paresis No

ky1Yes = feat2No(strcmp(feat2No.suddenWeightLoss, 'Yes'),:);

ky1No = feat2No(strcmp(feat2No.suddenWeightLoss, 'No'),:);

ky2Yes = feat2No(strcmp(feat2No.partialParesis, 'Yes'),:);

ky2No = feat2No(strcmp(feat2No.partialParesis, 'No'),:);

ky3Yes = ky2No(strcmp(ky2No.suddenWeightLoss, 'Yes'),:);

ky3NO = ky2No(strcmp(ky2No.suddenWeightLoss, 'No'),:);

%partialparesis yes

name1Yes = char1Yes(strcmp(char1Yes.suddenWeightLoss, 'Yes'),:);

name1No = char1Yes(strcmp(char1Yes.suddenWeightLoss, 'No'),:);

%partialParesis No

name2Yes = char1No(strcmp(char1No.suddenWeightLoss, 'Yes'),:);

name2No = char1No(strcmp(char1No.suddenWeightLoss, 'No'),:);

%END OF LEFT BRANCH

%Constructing the decision tree and comparing it to my results. My tree is

%the same as the built-in tree. Probably because the algorithm uses the

%same tactic as the built-in tree.

polyuriaone = table2array(trainSet(:,3));

polydipsiatwo = table2array(trainSet(:,4));

sWLthree = table2array(trainSet(:,5));

polyphagiafour = table2array(trainSet(:,7));

itchingfive = table2array(trainSet(:,10));

partialParalesissix = table2array(trainSet(:,13));

classseven = table2array(trainSet(:,17));

T = table(polyuriaone, polydipsiatwo, sWLthree, polyphagiafour, itchingfive, partialParalesissix, classseven);

decisionTree = fitctree(T,'classseven');

%view(decisionTree);

%view(decisionTree, 'Mode', 'Graph');

dataset = readtable('diabetes\_data\_upload.csv');

f = dataset;

%creating a random permutation of our dataset

k = randperm(520);

%using two thirds for our training set

trainSet = f(k(1:347),:);

%Using one third for our test set

testSet = f(k(348:520),:);

%Calculating entropy of training set

info = trainSet(strcmp(trainSet.class, 'Positive'),:);

info2 = trainSet(strcmp(trainSet.class, 'Negative'),:);

%Calculating the info gain on splitting on each attribute, using matlab to get the

%numbers needed

polyuriaYes = trainSet(strcmp(trainSet.Polyuria, 'Yes'),:);

polyuria1 = polyuriaYes(strcmp(polyuriaYes.class, 'Positive'),:);

polyuria2 = polyuriaYes(strcmp(polyuriaYes.class, 'Negative'),:);

polyuriaNo = trainSet(strcmp(trainSet.Polyuria, 'No'),:);

polyuria3 = polyuriaNo(strcmp(polyuriaNo.class, 'Positive'),:);

polyuria4 = polyuriaNo(strcmp(polyuriaNo.class, 'Negative'),:);

polydipsiaYes = trainSet(strcmp(trainSet.Polydipsia, 'Yes'),:);

polydipsia1 = polydipsiaYes(strcmp(polydipsiaYes.class, 'Positive'),:);

polydipsia2 = polydipsiaYes(strcmp(polydipsiaYes.class, 'Negative'),:);

polydipsiaNo = trainSet(strcmp(trainSet.Polydipsia, 'No'),:);

polydipsia3 = polydipsiaNo(strcmp(polydipsiaNo.class, 'Positive'),:);

polydipsia4 = polydipsiaNo(strcmp(polydipsiaNo.class, 'Negative'),:);

suddenWeightLossYes = trainSet(strcmp(trainSet.suddenWeightLoss, 'Yes'),:);

suddenWeightLoss1 = suddenWeightLossYes(strcmp(suddenWeightLossYes.class, 'Positive'),:);

suddenWeightLoss2 = suddenWeightLossYes(strcmp(suddenWeightLossYes.class, 'Negative'),:);

suddenWeightLossNo = trainSet(strcmp(trainSet.suddenWeightLoss, 'No'),:);

suddenWeightLoss3 = suddenWeightLossNo(strcmp(suddenWeightLossNo.class, 'Positive'),:);

suddenWeightLoss4 = suddenWeightLossNo(strcmp(suddenWeightLossNo.class, 'Negative'),:);

PolyphagiaYes = trainSet(strcmp(trainSet.Polyphagia, 'Yes'),:);

Polyphagia1 = PolyphagiaYes(strcmp(PolyphagiaYes.class, 'Positive'),:);

Polyphagia2 = PolyphagiaYes(strcmp(PolyphagiaYes.class, 'Negative'),:);

PolyphagiaNo = trainSet(strcmp(trainSet.Polyphagia, 'No'),:);

Polyphagia3 = PolyphagiaNo(strcmp(PolyphagiaNo.class, 'Positive'),:);

Polyphagia4 = PolyphagiaNo(strcmp(PolyphagiaNo.class, 'Negative'),:);

ItchingYes = trainSet(strcmp(trainSet.Itching, 'Yes'),:);

Itching1 = ItchingYes(strcmp(ItchingYes.class, 'Positive'),:);

Itching2 = ItchingYes(strcmp(ItchingYes.class, 'Negative'),:);

ItchingNo = trainSet(strcmp(trainSet.Itching, 'No'),:);

Itching3 = ItchingNo(strcmp(ItchingNo.class, 'Positive'),:);

Itching4 = ItchingNo(strcmp(ItchingNo.class, 'Negative'),:);

partialParesisYes = trainSet(strcmp(trainSet.partialParesis, 'Yes'),:);

partialParesis1 = partialParesisYes(strcmp(partialParesisYes.class, 'Positive'),:);

partialParesis2 = partialParesisYes(strcmp(partialParesisYes.class, 'Negative'),:);

partialParesisNo = trainSet(strcmp(trainSet.partialParesis, 'No'),:);

partialParesis3 = partialParesisNo(strcmp(partialParesisNo.class, 'Positive'),:);

partialParesis4 = partialParesisNo(strcmp(partialParesisNo.class, 'Negative'),:);

%After determining Polyuria as the root, the rest of the code works on

%continuing the tree. most of the code is for me to get the numbers needed

%to do the calculation on paper

polyuriaYes = trainSet(strcmp(trainSet.Polyuria, 'Yes'),:);

polyuria1 = polyuriaYes(strcmp(polyuriaYes.class, 'Positive'),:);

polyuria2 = polyuriaYes(strcmp(polyuriaYes.class, 'Negative'),:);

attrb1Yes = polyuriaYes(strcmp(polyuriaYes.Polydipsia, 'Yes'),:);

attrb1A = attrb1Yes(strcmp(attrb1Yes.class, 'Positive'),:);

attrb1B = attrb1Yes(strcmp(attrb1Yes.class, 'Negative'),:);

attrb1No = polyuriaYes(strcmp(polyuriaYes.Polydipsia, 'No'),:);

attrb1C = attrb1No(strcmp(attrb1No.class, 'Positive'),:);

attrb1D = attrb1No(strcmp(attrb1No.class, 'Negative'),:);

attrb2Yes = polyuriaYes(strcmp(polyuriaYes.suddenWeightLoss, 'Yes'),:);

attrb2A = attrb2Yes(strcmp(attrb2Yes.class, 'Positive'),:);

attrb2B = attrb2Yes(strcmp(attrb2Yes.class, 'Negative'),:);

attrb2No = polyuriaYes(strcmp(polyuriaYes.suddenWeightLoss, 'No'),:);

attrb2C = attrb2No(strcmp(attrb2No.class, 'Positive'),:);

attrb2D = attrb2No(strcmp(attrb2No.class, 'Negative'),:);

attrb3Yes = polyuriaYes(strcmp(polyuriaYes.Polyphagia, 'Yes'),:);

attrb3A = attrb3Yes(strcmp(attrb3Yes.class, 'Positive'),:);

attrb3B = attrb3Yes(strcmp(attrb3Yes.class, 'Negative'),:);

attrb3No = polyuriaYes(strcmp(polyuriaYes.Polyphagia, 'No'),:);

attrb3C = attrb3No(strcmp(attrb3No.class, 'Positive'),:);

attrb3D = attrb3No(strcmp(attrb3No.class, 'Negative'),:);

attrb4Yes = polyuriaYes(strcmp(polyuriaYes.Itching, 'Yes'),:);

attrb4A = attrb4Yes(strcmp(attrb4Yes.class, 'Positive'),:);

attrb4B = attrb4Yes(strcmp(attrb4Yes.class, 'Negative'),:);

attrb4No = polyuriaYes(strcmp(polyuriaYes.Itching, 'No'),:);

attrb4C = attrb4No(strcmp(attrb4No.class, 'Positive'),:);

attrb4D = attrb4No(strcmp(attrb4No.class, 'Negative'),:);

attrb5Yes = polyuriaYes(strcmp(polyuriaYes.partialParesis, 'Yes'),:);

attrb5A = attrb5Yes(strcmp(attrb5Yes.class, 'Positive'),:);

attrb5B = attrb5Yes(strcmp(attrb5Yes.class, 'Negative'),:);

attrb5No = polyuriaYes(strcmp(polyuriaYes.partialParesis, 'No'),:);

attrb5C = attrb5No(strcmp(attrb5No.class, 'Positive'),:);

attrb5D = attrb5No(strcmp(attrb5No.class, 'Negative'),:);

% Polyuria Yes and Polydipsia No

var1Yes = attrb1No(strcmp(attrb1No.suddenWeightLoss, 'Yes'),:);

var1A = var1Yes(strcmp(var1Yes.class, 'Positive'),:);

var1B = var1Yes(strcmp(var1Yes.class, 'Negative'),:);

var1No = attrb1No(strcmp(attrb1No.suddenWeightLoss, 'No'),:);

var1C = var1No(strcmp(var1No.class, 'Positive'),:);

var1D = var1No(strcmp(var1No.class, 'Negative'),:);

var2Yes = attrb1No(strcmp(attrb1No.Polyphagia, 'Yes'),:);

var2A = var2Yes(strcmp(var2Yes.class, 'Positive'),:);

var2B = var2Yes(strcmp(var2Yes.class, 'Negative'),:);

var2No = attrb1No(strcmp(attrb1No.Polyphagia, 'No'),:);

var2C = var2No(strcmp(var2No.class, 'Positive'),:);

var2D = var2No(strcmp(var2No.class, 'Negative'),:);

var3Yes = attrb1No(strcmp(attrb1No.partialParesis, 'Yes'),:);

var3A = var3Yes(strcmp(var3Yes.class, 'Positive'),:);

var3B = var3Yes(strcmp(var3Yes.class, 'Negative'),:);

var3No = attrb1No(strcmp(attrb1No.partialParesis, 'No'),:);

var3C = var3No(strcmp(var3No.class, 'Positive'),:);

var3D = var3No(strcmp(var3No.class, 'Negative'),:);

var4Yes = attrb1No(strcmp(attrb1No.Itching, 'Yes'),:);

var4A = var4Yes(strcmp(var4Yes.class, 'Positive'),:);

var4B = var4Yes(strcmp(var4Yes.class, 'Negative'),:);

var4No = attrb1No(strcmp(attrb1No.Itching, 'No'),:);

var4C = var4No(strcmp(var4No.class, 'Positive'),:);

var4D = var4No(strcmp(var4No.class, 'Negative'),:);

%Polyuria yes, polydipsia No, Itching Yes

feat1Yes = var4Yes(strcmp(var4Yes.suddenWeightLoss, 'Yes'),:);

feat1A = feat1Yes(strcmp(feat1Yes.class, 'Positive'),:);

feat1B = feat1Yes(strcmp(feat1Yes.class, 'Negative'),:);

feat1No = var4Yes(strcmp(var4Yes.suddenWeightLoss, 'No'),:);

feat1C = feat1No(strcmp(feat1No.class, 'Positive'),:);

feat1D = feat1No(strcmp(feat1No.class, 'Negative'),:);

feat2Yes = var4Yes(strcmp(var4Yes.Polyphagia, 'Yes'),:);

feat2A = feat2Yes(strcmp(feat2Yes.class, 'Positive'),:);

feat2B = feat2Yes(strcmp(feat2Yes.class, 'Negative'),:);

feat2No = var4Yes(strcmp(var4Yes.Polyphagia, 'No'),:);

feat2C = feat2No(strcmp(feat2No.class, 'Positive'),:);

feat2D = feat2No(strcmp(feat2No.class, 'Negative'),:);

feat3Yes = var4Yes(strcmp(var4Yes.partialParesis, 'Yes'),:);

feat3A = feat3Yes(strcmp(feat3Yes.class, 'Positive'),:);

feat3B = feat3Yes(strcmp(feat3Yes.class, 'Negative'),:);

feat3No = var4Yes(strcmp(var4Yes.partialParesis, 'No'),:);

feat3C = feat3No(strcmp(feat3No.class, 'Positive'),:);

feat3D = feat3No(strcmp(feat3No.class, 'Negative'),:);

%Polyphagia yes

char1Yes = feat2Yes(strcmp(feat2Yes.partialParesis, 'Yes'),:);

char1No = feat2Yes(strcmp(feat2Yes.partialParesis, 'No'),:);

char2Yes= feat2Yes(strcmp(feat2Yes.suddenWeightLoss, 'Yes'),:);

char2No= feat2Yes(strcmp(feat2Yes.suddenWeightLoss, 'No'),:);

%Polyphagia No and partial paresis No

ky1Yes = feat2No(strcmp(feat2No.suddenWeightLoss, 'Yes'),:);

ky1No = feat2No(strcmp(feat2No.suddenWeightLoss, 'No'),:);

ky2Yes = feat2No(strcmp(feat2No.partialParesis, 'Yes'),:);

ky2No = feat2No(strcmp(feat2No.partialParesis, 'No'),:);

ky3Yes = ky2No(strcmp(ky2No.suddenWeightLoss, 'Yes'),:);

ky3NO = ky2No(strcmp(ky2No.suddenWeightLoss, 'No'),:);

%partialparesis yes

name1Yes = char1Yes(strcmp(char1Yes.suddenWeightLoss, 'Yes'),:);

name1No = char1Yes(strcmp(char1Yes.suddenWeightLoss, 'No'),:);

%partialParesis No

name2Yes = char1No(strcmp(char1No.suddenWeightLoss, 'Yes'),:);

name2No = char1No(strcmp(char1No.suddenWeightLoss, 'No'),:);

%END OF LEFT BRANCH

%Constructing the decision tree and comparing it to my results. My tree is

%the same as the built-in tree. Probably because the algorithm uses the

%same tactic as the built-in tree.

polyuriaone = table2array(trainSet(:,3));

polydipsiatwo = table2array(trainSet(:,4));

sWLthree = table2array(trainSet(:,5));

polyphagiafour = table2array(trainSet(:,7));

itchingfive = table2array(trainSet(:,10));

partialParalesissix = table2array(trainSet(:,13));

classseven = table2array(trainSet(:,17));

T = table(polyuriaone, polydipsiatwo, sWLthree, polyphagiafour, itchingfive, partialParalesissix, classseven);

decisionTree = fitctree(T,'classseven');

%view(decisionTree);

%view(decisionTree, 'Mode', 'Graph');

%Code for information gain Tree, its a copy of the above code.

dataset = readtable('diabetes\_data\_upload.csv');

T=dataset;

Acc=[];

ER=[];

Sens=[];

Specif=[];

Prec=[];

Reca=[];

Fmeas=[];

T = randperm(520);

T = dataset(T,:);

for b=1:10

trainset=[];

testset=[];

for x=1:520

trainset=[trainset;T(x,:)];

if ((x>=(52\*(b-1)+1) && (x<=(52\*b))))

testset=[testset;T(x,:)];

end

end

trainsetx=trainset(:,[3,4,5,7,10,13]);

testsetx=testset(:,[3,4,5,7,10,13]);

decisionTree = fitctree(trainsetx,trainset.class);

w = predict(decisionTree, testsetx);

w(find(w=="Positive"))={'1'};

w(find(w=="Negative"))={'0'};

testset(find(testset.(17)=="Positive"),"class")={'1'};

testset(find(testset.(17)=="Negative"),"class")={'0'};

TP=0;

FP=0;

TN=0;

FN=0;

for i=1:(520\*.1)

comparew=cell2mat(w(i));

comparet=cell2mat(testset.class(i));

if (comparew==comparet && (comparew=='1'))

TP=TP+1;

elseif (comparew==comparet && (comparew=='0'))

TN=TN+1;

else

if (comparew=='1')

FP=FP+1;

elseif (comparew=='0')

FN=FN+1;

end

end

end

Accuracy = (TP+TN)/(520\*0.1);

ErrorRate=(FP+FN)/(520\*0.1);

Sensitivity=TP/(TP+FN);

Specificity=TN/(TN+FP);

if (isnan(Specificity)==1)

Specificity=1;

end

Precision=TP/(TP+FP);

Recall=TP/(TP+FN);

Fmeasure=(2\*Precision\*Recall)/(Precision+Recall);

Acc=[Acc;Accuracy];

ER=[ER;ErrorRate];

Sens=[Sens;Sensitivity];

Specif=[Specif;Specificity];

Prec=[Prec;Precision];

Reca=[Reca;Recall];

Fmeas=[Fmeas;Fmeasure];

end

M1 = mean(ER);

ER1=ER;

% Read data from csv

data = readtable("diabetes\_data\_upload.csv");

%-------------------------------------------------------------------------

% Randomly divide data into training

%-------------------------------------------------------------------------

% Get data dimension (rows and columns)

m=height(data);

% Percentage of training data to get 347 samples in training set

P = 0.668;

% Generate random index for given number of rows

idx = randperm(m);

% Select Training samples based on set percentage

Training = data(idx(1:round(P\*m)),:);

% set predictors names

inputNames = {'Polyuria', 'Polydipsia', 'suddenWeightLoss', 'Itching', 'partialParesis'};

% Convert Training data into training inputs and outputs

trainX = Training(:, inputNames);

trainY = Training.class;

%-------------------------------------------------------------------------

% Train decision classifier

%-------------------------------------------------------------------------

% Create the decision tree classifer

classificationTree = fitctree(...

trainX, ...

trainY, ...

'SplitCriterion', 'gdi', ... % Gini index

'MaxNumSplits', 100, ...

'Surrogate', 'off', ...

'ClassNames', categorical({'Negative'; 'Positive'}));

% Create the result struct with predict function

predictorExtractionFcn = @(t) t(:, inputNames);

treePredictFcn = @(x) predict(classificationTree, x);

trainedClassifier.predictFcn = @(x) treePredictFcn(predictorExtractionFcn(x));

% Add additional fields to the result struct

trainedClassifier.RequiredVariables = inputNames;

trainedClassifier.ClassificationTree = classificationTree;

%view(classificationTree, 'mode','graph'); % view decision tree

%Code for Gini Index, its a copy of the above code.

dataset = readtable('diabetes\_data\_upload.csv');

Acc=[];

ER=[];

Sens=[];

Specif=[];

Prec=[];

Reca=[];

Fmeas=[];

size = randperm(520);

T = dataset(size,:);

for b=1:10

trainset=[];

testset=[];

for x=1:520

trainset=[trainset;T(x,:)];

if ((x>=(52\*(b-1)+1) && (x<=(52\*b))))

testset=[testset;T(x,:)];

end

end

trainsetx=trainset(:,[3,4,5,10,13]);

testsetx=testset(:,[3,4,5,10,13]);

classificationTree = fitctree(...

trainsetx, ...

trainset.class, ...

'SplitCriterion', 'gdi', ... % Gini index

'MaxNumSplits', 100, ...

'Surrogate', 'off', ...

'ClassNames', categorical({'Negative'; 'Positive'}));

w = predict(classificationTree, testsetx);

w(find(w=="Positive"))={'1'};

w(find(w=="Negative"))={'0'};

testset(find(testset.(17)=="Positive"),"class")={'1'};

testset(find(testset.(17)=="Negative"),"class")={'0'};

TP=0;

FP=0;

TN=0;

FN=0;

for i=1:(520\*.1)

comparew=(w(i));

comparet=cell2mat(testset.class(i));

if (comparew==comparet && (comparew=='1'))

TP=TP+1;

elseif (comparew==comparet && (comparew=='0'))

TN=TN+1;

else

if (comparew=='1')

FP=FP+1;

elseif (comparew=='0')

FN=FN+1;

end

end

end

Accuracy = (TP+TN)/(520\*0.1);

ErrorRate=(FP+FN)/(520\*0.1);

Sensitivity=TP/(TP+FN);

Specificity=TN/(TN+FP);

if (isnan(Specificity)==1)

Specificity=1;

end

Precision=TP/(TP+FP);

Recall=TP/(TP+FN);

Fmeasure=(2\*Precision\*Recall)/(Precision+Recall);

Acc=[Acc;Accuracy];

ER=[ER;ErrorRate];

Sens=[Sens;Sensitivity];

Specif=[Specif;Specificity];

Prec=[Prec;Precision];

Reca=[Reca;Recall];

Fmeas=[Fmeas;Fmeasure];

end

ER2=ER;

errM1 = mean(ER);

errM1

%Actual Code and Calculation for T-Test

k = 10; %this represent how many time we ran the decision tree%

M1=ER1

M2=ER2

errM1=mean(M1)

errM2=mean(M2)

variance = (1/k)\*sum(abs((M1-M2 - (mean(M1) - mean(M2))).^2)); %this calculates how to find the variance of the 2 error rate%

TestT = (abs(mean(M1) - mean(M2)))/ sqrt (variance/k);% T-Test Formula

variance

TestT

## %T-Test For Information Gain and Naive Bayesian Classifier

dataset=readtable('diabetes\_data\_upload.csv','Delimiter',',');

f = dataset;

%creating a random permutation of our dataset

b = randperm(520);

%using two thirds for our training set

trainSet = f(b(1:347),:);

%Using one third for our test set

testSet = f(b(348:520),:);

%Calculating entropy of training set

info = trainSet(strcmp(trainSet.class, 'Positive'),:);

info2 = trainSet(strcmp(trainSet.class, 'Negative'),:);

%Calculating the info gain on splitting on each attribute, using matlab to get the

%numbers needed

polyuriaYes = trainSet(strcmp(trainSet.Polyuria, 'Yes'),:);

polyuria1 = polyuriaYes(strcmp(polyuriaYes.class, 'Positive'),:);

polyuria2 = polyuriaYes(strcmp(polyuriaYes.class, 'Negative'),:);

polyuriaNo = trainSet(strcmp(trainSet.Polyuria, 'No'),:);

polyuria3 = polyuriaNo(strcmp(polyuriaNo.class, 'Positive'),:);

polyuria4 = polyuriaNo(strcmp(polyuriaNo.class, 'Negative'),:);

polydipsiaYes = trainSet(strcmp(trainSet.Polydipsia, 'Yes'),:);

polydipsia1 = polydipsiaYes(strcmp(polydipsiaYes.class, 'Positive'),:);

polydipsia2 = polydipsiaYes(strcmp(polydipsiaYes.class, 'Negative'),:);

polydipsiaNo = trainSet(strcmp(trainSet.Polydipsia, 'No'),:);

polydipsia3 = polydipsiaNo(strcmp(polydipsiaNo.class, 'Positive'),:);

polydipsia4 = polydipsiaNo(strcmp(polydipsiaNo.class, 'Negative'),:);

suddenWeightLossYes = trainSet(strcmp(trainSet.suddenWeightLoss, 'Yes'),:);

suddenWeightLoss1 = suddenWeightLossYes(strcmp(suddenWeightLossYes.class, 'Positive'),:);

suddenWeightLoss2 = suddenWeightLossYes(strcmp(suddenWeightLossYes.class, 'Negative'),:);

suddenWeightLossNo = trainSet(strcmp(trainSet.suddenWeightLoss, 'No'),:);

suddenWeightLoss3 = suddenWeightLossNo(strcmp(suddenWeightLossNo.class, 'Positive'),:);

suddenWeightLoss4 = suddenWeightLossNo(strcmp(suddenWeightLossNo.class, 'Negative'),:);

PolyphagiaYes = trainSet(strcmp(trainSet.Polyphagia, 'Yes'),:);

Polyphagia1 = PolyphagiaYes(strcmp(PolyphagiaYes.class, 'Positive'),:);

Polyphagia2 = PolyphagiaYes(strcmp(PolyphagiaYes.class, 'Negative'),:);

PolyphagiaNo = trainSet(strcmp(trainSet.Polyphagia, 'No'),:);

Polyphagia3 = PolyphagiaNo(strcmp(PolyphagiaNo.class, 'Positive'),:);

Polyphagia4 = PolyphagiaNo(strcmp(PolyphagiaNo.class, 'Negative'),:);

ItchingYes = trainSet(strcmp(trainSet.Itching, 'Yes'),:);

Itching1 = ItchingYes(strcmp(ItchingYes.class, 'Positive'),:);

Itching2 = ItchingYes(strcmp(ItchingYes.class, 'Negative'),:);

ItchingNo = trainSet(strcmp(trainSet.Itching, 'No'),:);

Itching3 = ItchingNo(strcmp(ItchingNo.class, 'Positive'),:);

Itching4 = ItchingNo(strcmp(ItchingNo.class, 'Negative'),:);

partialParesisYes = trainSet(strcmp(trainSet.partialParesis, 'Yes'),:);

partialParesis1 = partialParesisYes(strcmp(partialParesisYes.class, 'Positive'),:);

partialParesis2 = partialParesisYes(strcmp(partialParesisYes.class, 'Negative'),:);

partialParesisNo = trainSet(strcmp(trainSet.partialParesis, 'No'),:);

partialParesis3 = partialParesisNo(strcmp(partialParesisNo.class, 'Positive'),:);

partialParesis4 = partialParesisNo(strcmp(partialParesisNo.class, 'Negative'),:);

%After determining Polyuria as the root, the rest of the code works on

%continuing the tree. most of the code is for me to get the numbers needed

%to do the calculation on paper

polyuriaYes = trainSet(strcmp(trainSet.Polyuria, 'Yes'),:);

polyuria1 = polyuriaYes(strcmp(polyuriaYes.class, 'Positive'),:);

polyuria2 = polyuriaYes(strcmp(polyuriaYes.class, 'Negative'),:);

attrb1Yes = polyuriaYes(strcmp(polyuriaYes.Polydipsia, 'Yes'),:);

attrb1A = attrb1Yes(strcmp(attrb1Yes.class, 'Positive'),:);

attrb1B = attrb1Yes(strcmp(attrb1Yes.class, 'Negative'),:);

attrb1No = polyuriaYes(strcmp(polyuriaYes.Polydipsia, 'No'),:);

attrb1C = attrb1No(strcmp(attrb1No.class, 'Positive'),:);

attrb1D = attrb1No(strcmp(attrb1No.class, 'Negative'),:);

attrb2Yes = polyuriaYes(strcmp(polyuriaYes.suddenWeightLoss, 'Yes'),:);

attrb2A = attrb2Yes(strcmp(attrb2Yes.class, 'Positive'),:);

attrb2B = attrb2Yes(strcmp(attrb2Yes.class, 'Negative'),:);

attrb2No = polyuriaYes(strcmp(polyuriaYes.suddenWeightLoss, 'No'),:);

attrb2C = attrb2No(strcmp(attrb2No.class, 'Positive'),:);

attrb2D = attrb2No(strcmp(attrb2No.class, 'Negative'),:);

attrb3Yes = polyuriaYes(strcmp(polyuriaYes.Polyphagia, 'Yes'),:);

attrb3A = attrb3Yes(strcmp(attrb3Yes.class, 'Positive'),:);

attrb3B = attrb3Yes(strcmp(attrb3Yes.class, 'Negative'),:);

attrb3No = polyuriaYes(strcmp(polyuriaYes.Polyphagia, 'No'),:);

attrb3C = attrb3No(strcmp(attrb3No.class, 'Positive'),:);

attrb3D = attrb3No(strcmp(attrb3No.class, 'Negative'),:);

attrb4Yes = polyuriaYes(strcmp(polyuriaYes.Itching, 'Yes'),:);

attrb4A = attrb4Yes(strcmp(attrb4Yes.class, 'Positive'),:);

attrb4B = attrb4Yes(strcmp(attrb4Yes.class, 'Negative'),:);

attrb4No = polyuriaYes(strcmp(polyuriaYes.Itching, 'No'),:);

attrb4C = attrb4No(strcmp(attrb4No.class, 'Positive'),:);

attrb4D = attrb4No(strcmp(attrb4No.class, 'Negative'),:);

attrb5Yes = polyuriaYes(strcmp(polyuriaYes.partialParesis, 'Yes'),:);

attrb5A = attrb5Yes(strcmp(attrb5Yes.class, 'Positive'),:);

attrb5B = attrb5Yes(strcmp(attrb5Yes.class, 'Negative'),:);

attrb5No = polyuriaYes(strcmp(polyuriaYes.partialParesis, 'No'),:);

attrb5C = attrb5No(strcmp(attrb5No.class, 'Positive'),:);

attrb5D = attrb5No(strcmp(attrb5No.class, 'Negative'),:);

% Polyuria Yes and Polydipsia No

var1Yes = attrb1No(strcmp(attrb1No.suddenWeightLoss, 'Yes'),:);

var1A = var1Yes(strcmp(var1Yes.class, 'Positive'),:);

var1B = var1Yes(strcmp(var1Yes.class, 'Negative'),:);

var1No = attrb1No(strcmp(attrb1No.suddenWeightLoss, 'No'),:);

var1C = var1No(strcmp(var1No.class, 'Positive'),:);

var1D = var1No(strcmp(var1No.class, 'Negative'),:);

var2Yes = attrb1No(strcmp(attrb1No.Polyphagia, 'Yes'),:);

var2A = var2Yes(strcmp(var2Yes.class, 'Positive'),:);

var2B = var2Yes(strcmp(var2Yes.class, 'Negative'),:);

var2No = attrb1No(strcmp(attrb1No.Polyphagia, 'No'),:);

var2C = var2No(strcmp(var2No.class, 'Positive'),:);

var2D = var2No(strcmp(var2No.class, 'Negative'),:);

var3Yes = attrb1No(strcmp(attrb1No.partialParesis, 'Yes'),:);

var3A = var3Yes(strcmp(var3Yes.class, 'Positive'),:);

var3B = var3Yes(strcmp(var3Yes.class, 'Negative'),:);

var3No = attrb1No(strcmp(attrb1No.partialParesis, 'No'),:);

var3C = var3No(strcmp(var3No.class, 'Positive'),:);

var3D = var3No(strcmp(var3No.class, 'Negative'),:);

var4Yes = attrb1No(strcmp(attrb1No.Itching, 'Yes'),:);

var4A = var4Yes(strcmp(var4Yes.class, 'Positive'),:);

var4B = var4Yes(strcmp(var4Yes.class, 'Negative'),:);

var4No = attrb1No(strcmp(attrb1No.Itching, 'No'),:);

var4C = var4No(strcmp(var4No.class, 'Positive'),:);

var4D = var4No(strcmp(var4No.class, 'Negative'),:);

%Polyuria yes, polydipsia No, Itching Yes

feat1Yes = var4Yes(strcmp(var4Yes.suddenWeightLoss, 'Yes'),:);

feat1A = feat1Yes(strcmp(feat1Yes.class, 'Positive'),:);

feat1B = feat1Yes(strcmp(feat1Yes.class, 'Negative'),:);

feat1No = var4Yes(strcmp(var4Yes.suddenWeightLoss, 'No'),:);

feat1C = feat1No(strcmp(feat1No.class, 'Positive'),:);

feat1D = feat1No(strcmp(feat1No.class, 'Negative'),:);

feat2Yes = var4Yes(strcmp(var4Yes.Polyphagia, 'Yes'),:);

feat2A = feat2Yes(strcmp(feat2Yes.class, 'Positive'),:);

feat2B = feat2Yes(strcmp(feat2Yes.class, 'Negative'),:);

feat2No = var4Yes(strcmp(var4Yes.Polyphagia, 'No'),:);

feat2C = feat2No(strcmp(feat2No.class, 'Positive'),:);

feat2D = feat2No(strcmp(feat2No.class, 'Negative'),:);

feat3Yes = var4Yes(strcmp(var4Yes.partialParesis, 'Yes'),:);

feat3A = feat3Yes(strcmp(feat3Yes.class, 'Positive'),:);

feat3B = feat3Yes(strcmp(feat3Yes.class, 'Negative'),:);

feat3No = var4Yes(strcmp(var4Yes.partialParesis, 'No'),:);

feat3C = feat3No(strcmp(feat3No.class, 'Positive'),:);

feat3D = feat3No(strcmp(feat3No.class, 'Negative'),:);

%Polyphagia yes

char1Yes = feat2Yes(strcmp(feat2Yes.partialParesis, 'Yes'),:);

char1No = feat2Yes(strcmp(feat2Yes.partialParesis, 'No'),:);

char2Yes= feat2Yes(strcmp(feat2Yes.suddenWeightLoss, 'Yes'),:);

char2No= feat2Yes(strcmp(feat2Yes.suddenWeightLoss, 'No'),:);

%Polyphagia No and partial paresis No

ky1Yes = feat2No(strcmp(feat2No.suddenWeightLoss, 'Yes'),:);

ky1No = feat2No(strcmp(feat2No.suddenWeightLoss, 'No'),:);

ky2Yes = feat2No(strcmp(feat2No.partialParesis, 'Yes'),:);

ky2No = feat2No(strcmp(feat2No.partialParesis, 'No'),:);

ky3Yes = ky2No(strcmp(ky2No.suddenWeightLoss, 'Yes'),:);

ky3NO = ky2No(strcmp(ky2No.suddenWeightLoss, 'No'),:);

%partialparesis yes

name1Yes = char1Yes(strcmp(char1Yes.suddenWeightLoss, 'Yes'),:);

name1No = char1Yes(strcmp(char1Yes.suddenWeightLoss, 'No'),:);

%partialParesis No

name2Yes = char1No(strcmp(char1No.suddenWeightLoss, 'Yes'),:);

name2No = char1No(strcmp(char1No.suddenWeightLoss, 'No'),:);

%END OF LEFT BRANCH

%Constructing the decision tree and comparing it to my results. My tree is

%the same as the built-in tree. Probably because the algorithm uses the

%same tactic as the built-in tree.

polyuriaone = table2array(trainSet(:,3));

polydipsiatwo = table2array(trainSet(:,4));

sWLthree = table2array(trainSet(:,5));

polyphagiafour = table2array(trainSet(:,7));

itchingfive = table2array(trainSet(:,10));

partialParalesissix = table2array(trainSet(:,13));

classseven = table2array(trainSet(:,17));

T = table(polyuriaone, polydipsiatwo, sWLthree, polyphagiafour, itchingfive, partialParalesissix, classseven);

decisionTree = fitctree(T,'classseven');

%view(decisionTree);

%view(decisionTree, 'Mode', 'Graph');

dataset = readtable('diabetes\_data\_upload.csv');

f = dataset;

%creating a random permutation of our dataset

k = randperm(520);

%using two thirds for our training set

trainSet = f(k(1:347),:);

%Using one third for our test set

testSet = f(k(348:520),:);

%Calculating entropy of training set

info = trainSet(strcmp(trainSet.class, 'Positive'),:);

info2 = trainSet(strcmp(trainSet.class, 'Negative'),:);

%Calculating the info gain on splitting on each attribute, using matlab to get the

%numbers needed

polyuriaYes = trainSet(strcmp(trainSet.Polyuria, 'Yes'),:);

polyuria1 = polyuriaYes(strcmp(polyuriaYes.class, 'Positive'),:);

polyuria2 = polyuriaYes(strcmp(polyuriaYes.class, 'Negative'),:);

polyuriaNo = trainSet(strcmp(trainSet.Polyuria, 'No'),:);

polyuria3 = polyuriaNo(strcmp(polyuriaNo.class, 'Positive'),:);

polyuria4 = polyuriaNo(strcmp(polyuriaNo.class, 'Negative'),:);

polydipsiaYes = trainSet(strcmp(trainSet.Polydipsia, 'Yes'),:);

polydipsia1 = polydipsiaYes(strcmp(polydipsiaYes.class, 'Positive'),:);

polydipsia2 = polydipsiaYes(strcmp(polydipsiaYes.class, 'Negative'),:);

polydipsiaNo = trainSet(strcmp(trainSet.Polydipsia, 'No'),:);

polydipsia3 = polydipsiaNo(strcmp(polydipsiaNo.class, 'Positive'),:);

polydipsia4 = polydipsiaNo(strcmp(polydipsiaNo.class, 'Negative'),:);

suddenWeightLossYes = trainSet(strcmp(trainSet.suddenWeightLoss, 'Yes'),:);

suddenWeightLoss1 = suddenWeightLossYes(strcmp(suddenWeightLossYes.class, 'Positive'),:);

suddenWeightLoss2 = suddenWeightLossYes(strcmp(suddenWeightLossYes.class, 'Negative'),:);

suddenWeightLossNo = trainSet(strcmp(trainSet.suddenWeightLoss, 'No'),:);

suddenWeightLoss3 = suddenWeightLossNo(strcmp(suddenWeightLossNo.class, 'Positive'),:);

suddenWeightLoss4 = suddenWeightLossNo(strcmp(suddenWeightLossNo.class, 'Negative'),:);

PolyphagiaYes = trainSet(strcmp(trainSet.Polyphagia, 'Yes'),:);

Polyphagia1 = PolyphagiaYes(strcmp(PolyphagiaYes.class, 'Positive'),:);

Polyphagia2 = PolyphagiaYes(strcmp(PolyphagiaYes.class, 'Negative'),:);

PolyphagiaNo = trainSet(strcmp(trainSet.Polyphagia, 'No'),:);

Polyphagia3 = PolyphagiaNo(strcmp(PolyphagiaNo.class, 'Positive'),:);

Polyphagia4 = PolyphagiaNo(strcmp(PolyphagiaNo.class, 'Negative'),:);

ItchingYes = trainSet(strcmp(trainSet.Itching, 'Yes'),:);

Itching1 = ItchingYes(strcmp(ItchingYes.class, 'Positive'),:);

Itching2 = ItchingYes(strcmp(ItchingYes.class, 'Negative'),:);

ItchingNo = trainSet(strcmp(trainSet.Itching, 'No'),:);

Itching3 = ItchingNo(strcmp(ItchingNo.class, 'Positive'),:);

Itching4 = ItchingNo(strcmp(ItchingNo.class, 'Negative'),:);

partialParesisYes = trainSet(strcmp(trainSet.partialParesis, 'Yes'),:);

partialParesis1 = partialParesisYes(strcmp(partialParesisYes.class, 'Positive'),:);

partialParesis2 = partialParesisYes(strcmp(partialParesisYes.class, 'Negative'),:);

partialParesisNo = trainSet(strcmp(trainSet.partialParesis, 'No'),:);

partialParesis3 = partialParesisNo(strcmp(partialParesisNo.class, 'Positive'),:);

partialParesis4 = partialParesisNo(strcmp(partialParesisNo.class, 'Negative'),:);

%After determining Polyuria as the root, the rest of the code works on

%continuing the tree. most of the code is for me to get the numbers needed

%to do the calculation on paper

polyuriaYes = trainSet(strcmp(trainSet.Polyuria, 'Yes'),:);

polyuria1 = polyuriaYes(strcmp(polyuriaYes.class, 'Positive'),:);

polyuria2 = polyuriaYes(strcmp(polyuriaYes.class, 'Negative'),:);

attrb1Yes = polyuriaYes(strcmp(polyuriaYes.Polydipsia, 'Yes'),:);

attrb1A = attrb1Yes(strcmp(attrb1Yes.class, 'Positive'),:);

attrb1B = attrb1Yes(strcmp(attrb1Yes.class, 'Negative'),:);

attrb1No = polyuriaYes(strcmp(polyuriaYes.Polydipsia, 'No'),:);

attrb1C = attrb1No(strcmp(attrb1No.class, 'Positive'),:);

attrb1D = attrb1No(strcmp(attrb1No.class, 'Negative'),:);

attrb2Yes = polyuriaYes(strcmp(polyuriaYes.suddenWeightLoss, 'Yes'),:);

attrb2A = attrb2Yes(strcmp(attrb2Yes.class, 'Positive'),:);

attrb2B = attrb2Yes(strcmp(attrb2Yes.class, 'Negative'),:);

attrb2No = polyuriaYes(strcmp(polyuriaYes.suddenWeightLoss, 'No'),:);

attrb2C = attrb2No(strcmp(attrb2No.class, 'Positive'),:);

attrb2D = attrb2No(strcmp(attrb2No.class, 'Negative'),:);

attrb3Yes = polyuriaYes(strcmp(polyuriaYes.Polyphagia, 'Yes'),:);

attrb3A = attrb3Yes(strcmp(attrb3Yes.class, 'Positive'),:);

attrb3B = attrb3Yes(strcmp(attrb3Yes.class, 'Negative'),:);

attrb3No = polyuriaYes(strcmp(polyuriaYes.Polyphagia, 'No'),:);

attrb3C = attrb3No(strcmp(attrb3No.class, 'Positive'),:);

attrb3D = attrb3No(strcmp(attrb3No.class, 'Negative'),:);

attrb4Yes = polyuriaYes(strcmp(polyuriaYes.Itching, 'Yes'),:);

attrb4A = attrb4Yes(strcmp(attrb4Yes.class, 'Positive'),:);

attrb4B = attrb4Yes(strcmp(attrb4Yes.class, 'Negative'),:);

attrb4No = polyuriaYes(strcmp(polyuriaYes.Itching, 'No'),:);

attrb4C = attrb4No(strcmp(attrb4No.class, 'Positive'),:);

attrb4D = attrb4No(strcmp(attrb4No.class, 'Negative'),:);

attrb5Yes = polyuriaYes(strcmp(polyuriaYes.partialParesis, 'Yes'),:);

attrb5A = attrb5Yes(strcmp(attrb5Yes.class, 'Positive'),:);

attrb5B = attrb5Yes(strcmp(attrb5Yes.class, 'Negative'),:);

attrb5No = polyuriaYes(strcmp(polyuriaYes.partialParesis, 'No'),:);

attrb5C = attrb5No(strcmp(attrb5No.class, 'Positive'),:);

attrb5D = attrb5No(strcmp(attrb5No.class, 'Negative'),:);

% Polyuria Yes and Polydipsia No

var1Yes = attrb1No(strcmp(attrb1No.suddenWeightLoss, 'Yes'),:);

var1A = var1Yes(strcmp(var1Yes.class, 'Positive'),:);

var1B = var1Yes(strcmp(var1Yes.class, 'Negative'),:);

var1No = attrb1No(strcmp(attrb1No.suddenWeightLoss, 'No'),:);

var1C = var1No(strcmp(var1No.class, 'Positive'),:);

var1D = var1No(strcmp(var1No.class, 'Negative'),:);

var2Yes = attrb1No(strcmp(attrb1No.Polyphagia, 'Yes'),:);

var2A = var2Yes(strcmp(var2Yes.class, 'Positive'),:);

var2B = var2Yes(strcmp(var2Yes.class, 'Negative'),:);

var2No = attrb1No(strcmp(attrb1No.Polyphagia, 'No'),:);

var2C = var2No(strcmp(var2No.class, 'Positive'),:);

var2D = var2No(strcmp(var2No.class, 'Negative'),:);

var3Yes = attrb1No(strcmp(attrb1No.partialParesis, 'Yes'),:);

var3A = var3Yes(strcmp(var3Yes.class, 'Positive'),:);

var3B = var3Yes(strcmp(var3Yes.class, 'Negative'),:);

var3No = attrb1No(strcmp(attrb1No.partialParesis, 'No'),:);

var3C = var3No(strcmp(var3No.class, 'Positive'),:);

var3D = var3No(strcmp(var3No.class, 'Negative'),:);

var4Yes = attrb1No(strcmp(attrb1No.Itching, 'Yes'),:);

var4A = var4Yes(strcmp(var4Yes.class, 'Positive'),:);

var4B = var4Yes(strcmp(var4Yes.class, 'Negative'),:);

var4No = attrb1No(strcmp(attrb1No.Itching, 'No'),:);

var4C = var4No(strcmp(var4No.class, 'Positive'),:);

var4D = var4No(strcmp(var4No.class, 'Negative'),:);

%Polyuria yes, polydipsia No, Itching Yes

feat1Yes = var4Yes(strcmp(var4Yes.suddenWeightLoss, 'Yes'),:);

feat1A = feat1Yes(strcmp(feat1Yes.class, 'Positive'),:);

feat1B = feat1Yes(strcmp(feat1Yes.class, 'Negative'),:);

feat1No = var4Yes(strcmp(var4Yes.suddenWeightLoss, 'No'),:);

feat1C = feat1No(strcmp(feat1No.class, 'Positive'),:);

feat1D = feat1No(strcmp(feat1No.class, 'Negative'),:);

feat2Yes = var4Yes(strcmp(var4Yes.Polyphagia, 'Yes'),:);

feat2A = feat2Yes(strcmp(feat2Yes.class, 'Positive'),:);

feat2B = feat2Yes(strcmp(feat2Yes.class, 'Negative'),:);

feat2No = var4Yes(strcmp(var4Yes.Polyphagia, 'No'),:);

feat2C = feat2No(strcmp(feat2No.class, 'Positive'),:);

feat2D = feat2No(strcmp(feat2No.class, 'Negative'),:);

feat3Yes = var4Yes(strcmp(var4Yes.partialParesis, 'Yes'),:);

feat3A = feat3Yes(strcmp(feat3Yes.class, 'Positive'),:);

feat3B = feat3Yes(strcmp(feat3Yes.class, 'Negative'),:);

feat3No = var4Yes(strcmp(var4Yes.partialParesis, 'No'),:);

feat3C = feat3No(strcmp(feat3No.class, 'Positive'),:);

feat3D = feat3No(strcmp(feat3No.class, 'Negative'),:);

%Polyphagia yes

char1Yes = feat2Yes(strcmp(feat2Yes.partialParesis, 'Yes'),:);

char1No = feat2Yes(strcmp(feat2Yes.partialParesis, 'No'),:);

char2Yes= feat2Yes(strcmp(feat2Yes.suddenWeightLoss, 'Yes'),:);

char2No= feat2Yes(strcmp(feat2Yes.suddenWeightLoss, 'No'),:);

%Polyphagia No and partial paresis No

ky1Yes = feat2No(strcmp(feat2No.suddenWeightLoss, 'Yes'),:);

ky1No = feat2No(strcmp(feat2No.suddenWeightLoss, 'No'),:);

ky2Yes = feat2No(strcmp(feat2No.partialParesis, 'Yes'),:);

ky2No = feat2No(strcmp(feat2No.partialParesis, 'No'),:);

ky3Yes = ky2No(strcmp(ky2No.suddenWeightLoss, 'Yes'),:);

ky3NO = ky2No(strcmp(ky2No.suddenWeightLoss, 'No'),:);

%partialparesis yes

name1Yes = char1Yes(strcmp(char1Yes.suddenWeightLoss, 'Yes'),:);

name1No = char1Yes(strcmp(char1Yes.suddenWeightLoss, 'No'),:);

%partialParesis No

name2Yes = char1No(strcmp(char1No.suddenWeightLoss, 'Yes'),:);

name2No = char1No(strcmp(char1No.suddenWeightLoss, 'No'),:);

%END OF LEFT BRANCH

%Constructing the decision tree and comparing it to my results. My tree is

%the same as the built-in tree. Probably because the algorithm uses the

%same tactic as the built-in tree.

polyuriaone = table2array(trainSet(:,3));

polydipsiatwo = table2array(trainSet(:,4));

sWLthree = table2array(trainSet(:,5));

polyphagiafour = table2array(trainSet(:,7));

itchingfive = table2array(trainSet(:,10));

partialParalesissix = table2array(trainSet(:,13));

classseven = table2array(trainSet(:,17));

T = table(polyuriaone, polydipsiatwo, sWLthree, polyphagiafour, itchingfive, partialParalesissix, classseven);

decisionTree = fitctree(T,'classseven');

%view(decisionTree);

%view(decisionTree, 'Mode', 'Graph');

%Code for Cross Validation for information gain Tree, its a copy of the above code.

dataset = readtable('diabetes\_data\_upload.csv');

T=dataset;

Acc=[];

ER=[];

Sens=[];

Specif=[];

Prec=[];

Reca=[];

Fmeas=[];

T = randperm(520);

T = dataset(T,:);

for b=1:10

trainset=[];

testset=[];

for x=1:520

trainset=[trainset;T(x,:)];

if ((x>=(52\*(b-1)+1) && (x<=(52\*b))))

testset=[testset;T(x,:)];

end

end

trainsetx=trainset(:,[3,4,5,7,10,13]);

testsetx=testset(:,[3,4,5,7,10,13]);

decisionTree = fitctree(trainsetx,trainset.class);

w = predict(decisionTree, testsetx);

w(find(w=="Positive"))={'1'};

w(find(w=="Negative"))={'0'};

testset(find(testset.(17)=="Positive"),"class")={'1'};

testset(find(testset.(17)=="Negative"),"class")={'0'};

TP=0;

FP=0;

TN=0;

FN=0;

for i=1:(520\*.1)

comparew=cell2mat(w(i));

comparet=cell2mat(testset.class(i));

if (comparew==comparet && (comparew=='1'))

TP=TP+1;

elseif (comparew==comparet && (comparew=='0'))

TN=TN+1;

else

if (comparew=='1')

FP=FP+1;

elseif (comparew=='0')

FN=FN+1;

end

end

end

Accuracy = (TP+TN)/(520\*0.1);

ErrorRate=(FP+FN)/(520\*0.1);

Sensitivity=TP/(TP+FN);

Specificity=TN/(TN+FP);

if (isnan(Specificity)==1)

Specificity=1;

end

Precision=TP/(TP+FP);

Recall=TP/(TP+FN);

Fmeasure=(2\*Precision\*Recall)/(Precision+Recall);

Acc=[Acc;Accuracy];

ER=[ER;ErrorRate];

Sens=[Sens;Sensitivity];

Specif=[Specif;Specificity];

Prec=[Prec;Precision];

Reca=[Reca;Recall];

Fmeas=[Fmeas;Fmeasure];

end

M1 = mean(ER);

ER1=ER;

dataset=readtable('diabetes\_data\_upload.csv','Delimiter',',');

T=dataset;

T(find(T.(3)=="Yes"),"Polyuria")={'1'};

T(find(T.(3)=="No"),"Polyuria")={'0'};

T(find(T.(4)=="Yes"),"Polydipsia")={'1'};

T(find(T.(4)=="No"),"Polydipsia")={'0'};

T(find(T.(5)=="Yes"),"suddenWeightLoss")={'1'};

T(find(T.(5)=="No"),"suddenWeightLoss")={'0'};

T(find(T.(6)=="Yes"),"weakness")={'1'};

T(find(T.(6)=="No"),"weakness")={'0'};

T(find(T.(7)=="Yes"),"Polyphagia")={'1'};

T(find(T.(7)=="No"),"Polyphagia")={'0'};

T(find(T.(8)=="Yes"),"GenitalThrush")={'1'};

T(find(T.(8)=="No"),"GenitalThrush")={'0'};

T(find(T.(9)=="Yes"),"visualBlurring")={'1'};

T(find(T.(9)=="No"),"visualBlurring")={'0'};

T(find(T.(10)=="Yes"),"Itching")={'1'};

T(find(T.(10)=="No"),"Itching")={'0'};

T(find(T.(11)=="Yes"),"Irritability")={'1'};

T(find(T.(11)=="No"),"Irritability")={'0'};

T(find(T.(12)=="Yes"),"delayedHealing")={'1'};

T(find(T.(12)=="No"),"delayedHealing")={'0'};

T(find(T.(13)=="Yes"),"partialParesis")={'1'};

T(find(T.(13)=="No"),"partialParesis")={'0'};

T(find(T.(14)=="Yes"),"muscleStiffness")={'1'};

T(find(T.(14)=="No"),"muscleStiffness")={'0'};

T(find(T.(15)=="Yes"),"Alopecia")={'1'};

T(find(T.(15)=="No"),"Alopecia")={'0'};

T(find(T.(16)=="Yes"),"Obesity")={'1'};

T(find(T.(16)=="No"),"Obesity")={'0'};

testset=T;

trainset=[];

x=[1:520];

toremove=[];

for b=1:346

r=round((length(x)-1)\*rand()+1);

trainset=[trainset;T([x(r)],:)];

toremove=[toremove;x(r)];

x(r)=[];

end

testset([toremove],:)=[];

trainsetx=trainset(:,2:16);

testsetx=testset(:,2:16);

m=fitcnb(trainsetx,trainset.class);

w=predict(m,testsetx);

errNB=loss(m,testsetx,testset.class);

disp([num2str(errNB),'-Naive Bayes loss']) %find loss in the model

%10-Fold Cross Validation Method for Naive Bayes

dataset=readtable('diabetes\_data\_upload.csv','Delimiter',',');

T=dataset;

Acc=[];

ER=[];

Sens=[];

Specif=[];

Prec=[];

Reca=[];

Fmeas=[];

T = randperm(520);

T = dataset(T,:);

for b=1:10

trainset=[];

testset=[];

for x=1:520

trainset=[trainset;T(x,:)];

if ((x>=(52\*(b-1)+1) && (x<=(52\*b))))

testset=[testset;T(x,:)];

end

end

trainsetx=trainset(:,2:16);

testsetx=testset(:,2:16);

m=fitcnb(trainsetx,trainset.class);

w=predict(m,testsetx);

w(find(w=="Positive"))={'1'};

w(find(w=="Negative"))={'0'};

testset(find(testset.(17)=="Positive"),"class")={'1'};

testset(find(testset.(17)=="Negative"),"class")={'0'};

TP=0;

FP=0;

TN=0;

FN=0;

for i=1:(520\*.1)

comparew=cell2mat(w(i));

comparet=cell2mat(testset.class(i));

if (comparew==comparet && (comparew=='1'))

TP=TP+1;

elseif (comparew==comparet && (comparew=='0'))

TN=TN+1;

else

if (comparew=='1')

FP=FP+1;

elseif (comparew=='0')

FN=FN+1;

end

end

end

Accuracy = (TP+TN)/(520\*0.1);

ErrorRate=(FP+FN)/(520\*0.1);

Sensitivity=TP/(TP+FN);

Specificity=TN/(FP+TN);

if (isnan(Specificity)==1)

Specificity=1;

end

Precision=TP/(TP+FP);

Recall=TP/(TP+FN);

Fmeasure=(2\*Precision\*Recall)/(Precision+Recall);

Acc=[Acc;Accuracy];

ER=[ER;ErrorRate];

Sens=[Sens;Sensitivity];

Specif=[Specif;Specificity];

Prec=[Prec;Precision];

Reca=[Reca;Recall];

Fmeas=[Fmeas;Fmeasure];

end

ER2=ER;

errM2=mean(ER);

%Actual Code and Calculation for T-Test

k = 10; %this represent how many time we ran the decision tree%

M1=ER1

M2=ER2

errM1=mean(M1)

errM2=mean(M2)

variance = (1/k)\*sum(abs((M1-M2 - (mean(M1) - mean(M2))).^2)); %this calculates how to find the variance of the 2 error rate%

TestT = (abs(mean(M1) - mean(M2)))/ sqrt (variance/k);% T-Test Formula

variance

TestT