Machine Learning Final Project

A11 -

Vinay Agarwal

Amber Huang

Elaine Jennings

Joseph Jiang

Aysouda Vatanparast

BANA 271: Machine Learning for Analytics

Dec 2, 2022

Table of Contents

Introduction

Horses are a beloved working animals as well as the occasional pet. With any animal that people take care of, we want them to live a long healthy life. There can be a lot of factors that affect the longevity of a horse's life, especially once they're taken to the vet for a present issue. On the website Kaggle, there's a dataset titled "Horse Survival Dataset" that outlines various medical conditions and whether or not the horse was able to survive once it was taken to the vet. Some of the factors include age, whether the horse was treated with surgery, mucous membrane color, abdomen fluid color, and 23 other attributes that all contribute to whether or not the horse survived. This is important data for veterinarians who rely on their success rate of helping horses survive in order to get and keep clients.

The question we are trying to answer is, based on this dataset, which factor(s) link to horse survival the most. Specifically, how do factor(s) associate with the survival rates for horses. We are planning to clean the dataset: exclude all impossible values (may replace them with column mean or other data cleaning method). Then analyze the above topic and predict certain future events.

The Data Set

After the data was cleaned we were left with 21 columns that described the horse's condition and an additional column with what the outcome was. Five of the columns are numerical while the remaining 16 columns are categorical. Since the data describing the horse's condition is in the context of a veterinary exam, there are multiple values that are technical and related to the exam conducted by the veterinarian. The categorical data can be briefly described as follows:

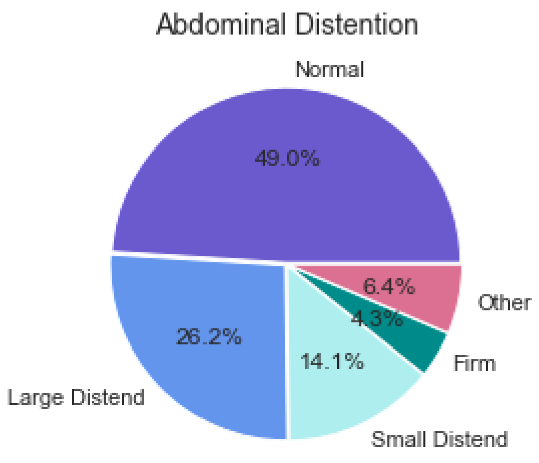
Attribute	Values
Surgery	Yes/No
Age	Young/Adult
Temp of Extremities	Normal, Warm, Cool, Cold
Peripheral Pulse	Normal, Absent, Reduced, Increased

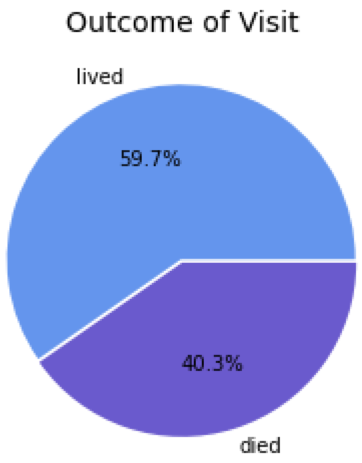
Mucous Membrane	Normal Pink, Pale Pink, Pale Cyanotic, Bright Pink, Bright Red, Dark Cyanotic
Capillary Refill Time	> 3 seconds, 3 seconds, < 3 seconds
Pain	Alert, Mild Pain, Depressed, Extreme Pain, Severe Pain
Peristalsis	Hypomotile, Hypermotile, Absent, Normal
Abdominal Distention	None, Moderate, Slight, Severe
Rectal Exam	Normal, Absent, Increased, Decreased
Abdomen	Normal, Distend Large, Distend Small, Firm, Other
Outcome	Lived, Died
Surgical Lesion	Yes, No
Lesion	1, 0
Lesion Site	0-11, each number representing a different area
Lesion Type	0-4, each number representing a different type
Lesion Subtype	0-3, each number representing a different subtype
Lesion Code	0-10, each number representing a different code

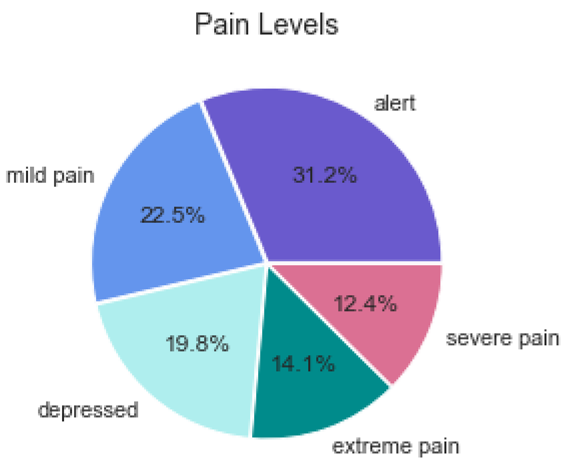
The numerical data can be similarly explained below:

Attribute	**Min**	**Max**
Rectal Temp	35.4	40.8
Pulse	30	184
Respiratory Rate	8	96
Packed Cell Volume	23	75
Total Protein	3.3	89

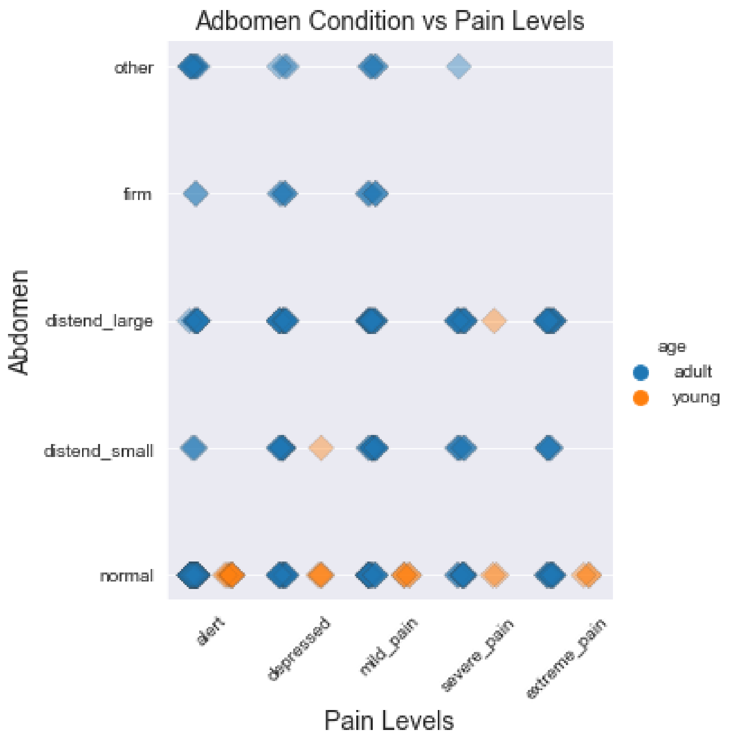
In order to better visualize the spread of some attributes, we created a number of visuals. From the pie charts we can see that the pain levels reported were relatively evenly distributed, with the largest section being 'alert'. There's also an almost even split of outcomes, with 59.7% living and the other 40.3% passing away. As for abdominal distention, almost half of the data was normal with the next largest section showing 'large'.







In an effort to visualize additional relationships between the attributes, a category chart was made. This graph shows the spread of data points when it comes to Abdominal condition and pain levels with a split between age groups. While there are fewer data points for the young age group, it's interesting to see that they're almost exclusively in the normal abdominal condition section. Another interesting take away from this visualization is that there isn't a clear correlation between the two variables. The data points seem just as saturated across pain levels when it comes to different abdominal conditions.



Analysis

Naive Bayes

We proceeded to run Naive Bayes classification method in weka on the subject dataset. The initial result is shown below:

```
=== Summary ===
Correctly Classified Instances        224              75.1678 %
Incorrectly Classified Instances       74              24.8322 %
Kappa statistic                    0.4974
Mean absolute error                   0.2471
Root mean squared error                 0.4513
Relative absolute error              51.3577 %
Root relative squared error            92.0175 %
Total Number of Instances              298
=== Detailed Accuracy By Class ===
                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
Class
              0.775    0.264    0.664     0.775    0.715     0.502    0.849     0.765    0
              0.736    0.225    0.829     0.736    0.780     0.502    0.849     0.899    1
Weighted Avg.    0.752    0.241    0.763     0.752    0.754     0.502    0.849    0.845
=== Confusion Matrix ===
   a   b   <-- classified as
  93  27 |   a = 0
  47 131 |   b = 1
```

After getting the initial results we attempted to increase the accuracy of the model by running Weka with various numbers of bins for each attribute, depending on the parameter. The new updated results can be seen below:

```
=== Summary ===
Correctly Classified Instances        228              76.5101 %
Incorrectly Classified Instances       70              23.4899 %
Kappa statistic                    0.5221
Mean absolute error                   0.2384
Root mean squared error                 0.4447
Relative absolute error              49.5441 %
Root relative squared error            90.6681 %
Total Number of Instances              298
=== Detailed Accuracy By Class ===
                  TP Rate   FP Rate   Precision   Recall    F-Measure   MCC       ROC Area  PRC Area  
Class
                0.775    0.242    0.684     0.775    0.727     0.525    0.856     0.777     '(-inf-0.083333]'
                0.758    0.225    0.833     0.758    0.794     0.525    0.856     0.906     '(0.083333-inf)'
Weighted Avg.    0.765    0.232    0.773     0.765    0.767     0.525    0.856     0.854
=== Confusion Matrix ===
   a   b   <-- classified as
  93  27 |   a = '(-inf-0.083333]'
  43 135 |   b = '(0.083333-inf)'
```

We thought Naive Bayes classification would be a good idea to start the analysis on the subject dataset since it analyzes every pair of features being classified that is independent of each other. In terms of binning, we pre-processed the dataset several times before proceeding to analysis in Weka. We first excluded all irrelevant (and/or redundant) attributes and also, numericalized the ‘Outcome’ attribute (1: Lived; 0: Died, Euthanized). In Weka, we further manipulated the data by binning all numerical variables. In this step, we adopted the “findNumBins” function in Weka while setting “bins” to 12, which automatically finds the optimal numbers of bins to discretize the numerical attributes. For instance, the resulting bins happen to be (-inf,0.083333] and (0.083333-inf) for the attribute “Outcome”. This is optimal because the resulting bins perfectly cover both possible values 1 and 0 for these attributes. During the analysis, 10-fold cross-validations were performed by default. So both results above are with 10-fold cross-validations performed. We further investigated this binning option for all other numerical attributes and made sure there were boned correctly then proceeded to run Naive Bayes classification. According to the above Confusion Matrix and other results, Naive Bayes classification produces a fairly good model. Binning to various numbers of bins for all numeric attributes, depending on the parameter of the subject attribute, produces a 76.5% accuracy. With type I error 27/298 = 0.091 and type II error 0.144. Compared to the 75.2% accuracy without binning with the same method, the results show binning is not only a necessary step, but also more accurate.

<u>Association Rule/Apriori</u>

In order to get a more full picture of the data, we also ran the Apriori analysis through Weka. Firstly, we had to discretize our attributes and then run the association. For discretization, we considered 5 bins and also we changed the car to True, to make sure that the focus is on desired class.

```
=== Run information ===
Scheme: weka.associations.Apriori -N 20 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -A -c -1
Apriori
=======
Minimum support: 0.1 (30 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 18
Generated sets of large itemsets:
Size of set of large itemsets L(1): 44
Size of set of large itemsets L(2): 172
Size of set of large itemsets L(3): 253
Size of set of large itemsets L(4): 166
Size of set of large itemsets L(5): 45
Size of set of large itemsets L(6): 2

Best rules found:
 1. surgery=no temp_of_extremities=normal mucous_membrane=normal_pink 31 ==>
outcome_binary='(0.5-inf)' 30    conf:(0.97)
 2. surgery=no temp_of_extremities=normal peripheral_pulse=normal
mucous_membrane=normal_pink 31 ==> outcome_binary='(0.5-inf)' 30    conf:(0.97)
 3. surgery=no pulse='(-inf-58]' mucous_membrane=normal_pink 38 ==>
outcome_binary='(0.5-inf)' 36    conf:(0.95)
 4. surgery=no age=adult pulse='(-inf-58]' mucous_membrane=normal_pink 38 ==>
outcome_binary='(0.5-inf)' 36    conf:(0.95)
 5. surgery=no age=adult temp_of_extremities=normal capillary_refill_time=less_3_sec 37 ==>
outcome_binary='(0.5-inf)' 35    conf:(0.95)
 6. surgery=no pulse='(-inf-58]' peripheral_pulse=normal mucous_membrane=normal_pink 37
==> outcome_binary='(0.5-inf)' 35    conf:(0.95)
 7. surgery=no age=adult pulse='(-inf-58]' peripheral_pulse=normal
mucous_membrane=normal_pink 37 ==> outcome_binary='(0.5-inf)' 35    conf:(0.95)
 8. surgery=no age=adult temp_of_extremities=normal peripheral_pulse=normal
capillary_refill_time=less_3_sec 36 ==> outcome_binary='(0.5-inf)' 34    conf:(0.94)
 9. surgery=no mucous_membrane=normal_pink abdominal_distention=none 35 ==>
outcome_binary='(0.5-inf)' 33    conf:(0.94)
```

```
10. surgery=no age=adult peripheral_pulse=normal peristalsis=hypomotile 35 ==>
outcome_binary='(0.5-inf)' 33    conf:(0.94)
11. surgery=no peripheral_pulse=normal mucous_membrane=normal_pink
abdominal_distention=none 35 ==> outcome_binary='(0.5-inf)' 33    conf:(0.94)
12. surgery=no pulse='(-inf-58]' temp_of_extremities=normal 33 ==> outcome_binary='(0.5-inf)'
31    conf:(0.94)
13. surgery=no age=adult pulse='(-inf-58]' temp_of_extremities=normal 33 ==>
outcome_binary='(0.5-inf)' 31    conf:(0.94)
14. surgery=no pulse='(-inf-58]' mucous_membrane=normal_pink
capillary_refill_time=less_3_sec 33 ==> outcome_binary='(0.5-inf)' 31    conf:(0.94)
15. surgery=no age=adult pulse='(-inf-58]' mucous_membrane=normal_pink
capillary_refill_time=less_3_sec 33 ==> outcome_binary='(0.5-inf)' 31    conf:(0.94)
16. surgery=no pulse='(-inf-58]' temp_of_extremities=normal peripheral_pulse=normal 32 ==>
outcome_binary='(0.5-inf)' 30    conf:(0.94)
17. pulse='(-inf-58]' temp_of_extremities=normal mucous_membrane=normal_pink
capillary_refill_time=less_3_sec 32 ==> outcome_binary='(0.5-inf)' 30    conf:(0.94)
18. surgery=no age=adult pulse='(-inf-58]' temp_of_extremities=normal
peripheral_pulse=normal 32 ==> outcome_binary='(0.5-inf)' 30    conf:(0.94)
19. surgery=no pulse='(-inf-58]' peripheral_pulse=normal mucous_membrane=normal_pink
capillary_refill_time=less_3_sec 32 ==> outcome_binary='(0.5-inf)' 30    conf:(0.94)
20. age=adult pulse='(-inf-58]' temp_of_extremities=normal mucous_membrane=normal_pink
capillary_refill_time=less_3_sec 32 ==> outcome_binary='(0.5-inf)' 30    conf:(0.94)
```

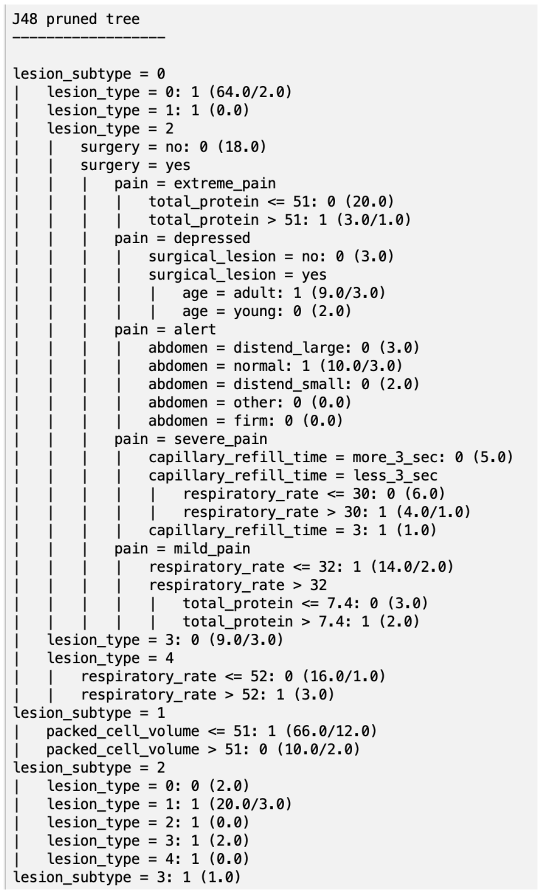
According to the above association model, the confidence levels for most of the top-generating rules are over 94%, which we can assume is good, However, for rules 1 and 2, the confidence level is 97% which is a high level of accuracy.

According to the first and second rules, if the horse had no surgery, the temp of extremities is normal and the mucous membrane is normal_pink, with a 97% confidence level the binary outcome is '(0.5-inf)', which means the animal will survive. (30 associations)
According to the 3rd and 4th rules, if the horse had no surgery and the pulse is '(-inf-58]' and the mucous membrane is normal_pink with a 95% confidence level the binary outcome is '(0.5-inf)', on the other word the horse will survive. (36 associations)

According to the 5th rule, if the horse had no surgery, the anima is adult and the temp of extremities is normal, and capillary refill time is less than 3 seconds the binary outcome with 95% confidence is '(0.5-inf)', so, the animal survives.(35 associations)

Decision Tree

First, all 22 attributes were taken into account to build a decision tree to predict the outcome of the horses. Ten folds cross-validation was used to train and test the model, since experiments have already proved that it provides the best estimate. We pruned the decision tree and set the minimum number of instances per leaf as 2, which is the number that is defaulted in Weka. The following decision tree is generated:



```
J48 pruned tree
------------------

lesion_subtype = 0
|   lesion_type = 0: 1 (64.0/2.0)
|   lesion_type = 1: 1 (0.0)
|   lesion_type = 2
|   |   surgery = no: 0 (18.0)
|   |   surgery = yes
|   |   |   pain = extreme_pain
|   |   |   |   total_protein <= 51: 0 (20.0)
|   |   |   |   total_protein > 51: 1 (3.0/1.0)
|   |   |   pain = depressed
|   |   |   |   surgical_lesion = no: 0 (3.0)
|   |   |   |   surgical_lesion = yes
|   |   |   |   |   age = adult: 1 (9.0/3.0)
|   |   |   |   |   age = young: 0 (2.0)
|   |   |   pain = alert
|   |   |   |   abdomen = distend_large: 0 (3.0)
|   |   |   |   abdomen = normal: 1 (10.0/3.0)
|   |   |   |   abdomen = distend_small: 0 (2.0)
|   |   |   |   abdomen = other: 0 (0.0)
|   |   |   |   abdomen = firm: 0 (0.0)
|   |   |   pain = severe_pain
|   |   |   |   capillary_refill_time = more_3_sec: 0 (5.0)
|   |   |   |   capillary_refill_time = less_3_sec
|   |   |   |   |   respiratory_rate <= 30: 0 (6.0)
|   |   |   |   |   respiratory_rate > 30: 1 (4.0/1.0)
|   |   |   |   capillary_refill_time = 3: 1 (1.0)
|   |   |   pain = mild_pain
|   |   |   |   respiratory_rate <= 32: 1 (14.0/2.0)
|   |   |   |   respiratory_rate > 32
|   |   |   |   |   total_protein <= 7.4: 0 (3.0)
|   |   |   |   |   total_protein > 7.4: 1 (2.0)
|   lesion_type = 3: 0 (9.0/3.0)
|   lesion_type = 4
|   |   respiratory_rate <= 52: 0 (16.0/1.0)
|   |   respiratory_rate > 52: 1 (3.0)
lesion_subtype = 1
|   packed_cell_volume <= 51: 1 (66.0/12.0)
|   packed_cell_volume > 51: 0 (10.0/2.0)
lesion_subtype = 2
|   lesion_type = 0: 0 (2.0)
|   lesion_type = 1: 1 (20.0/3.0)
|   lesion_type = 2: 1 (0.0)
|   lesion_type = 3: 1 (2.0)
|   lesion_type = 4: 1 (0.0)
lesion_subtype = 3: 1 (1.0)
```

This decision tree has an accuracy of 69.1275%, where 206 instances are correctly classified and 92 are incorrectly classified. The confusion matrix of this model is shown below:

```
=== Confusion Matrix ===
  a   b   <-- classified as
 71   49 |   a = 0 (died)
 43  135 |   b = 1 (lived)
```

114 instances are classified as "died", while 43 are incorrectly classified. 184 instances are classified as "lived", and 49 are incorrectly classified. The accuracy for "died" is 59.17%, while the accuracy for "lived" is 75.84%. The confusion matrix shows that this model is better at predicting the horse which survived but cannot accurately predict those that died.

We set the minimum number of instances per leaf as 3, which is about 1% of our entire dataset, and the accuracy of the decision tree increased to 71.4765%, with 213 instances being correctly classified and 85 being incorrectly classified.

```
=== Confusion Matrix ===
   a   b   <-- classified as
  72   48 |   a = 0 (died)
  37  141 |   b = 1 (lived)
```

The confusion matrix shows that the accuracy for "died" improved, but still a lot lower than the accuracy of "lived".

Although 71.48% of accuracy isn't that bad, it'll be good to see what adjustments can be made to make the model better. As a result, a number of pre-processing steps were performed to see whether the accuracy of the model will increase.

Discretize Numeric Attributes

Decision tree algorithms can split continuous attributes but can be cleaner when split points are chosen between bins or predefined groups. 5 of the 22 attributes are continuous. By applying unsupervised discretization and converting each of them into 15 bins using equal-width-binning, a new decision tree is generated. The accuracy of the model is 72.1477%, where 215 instances are correctly classified and 83 are incorrectly classified. This result is a bit better than the previous one without binning, which could be concluded that discretizing the attributes does help in improving the accuracy of the model. The confusion matrix is as follows:

```
=== Confusion Matrix ===
   a   b   <-- classified as
  91   29 |   a = 0 (died)
  54  124 |   b = 1 (lived)
```

145 instances are classified as "died", while 54 are incorrectly classified. 153 instances are classified as "lived", and 29 are incorrectly classified. The confusion matrix shows that the accuracy for "died" became worse, but the accuracy for predicting "lived" improved. This model is better at predicting the horse which survived but cannot that accurately predict those that died.

Bagging

Since discretization of numerical attributes does help in improving the accuracy of the model, the dataset with binning is used for bagging. Decision tree is an unstable classifier, as a result, bagging might help in improving the performance. It was set to repeatedly generate 10 training sets with the same size of the original data. The accuracy of the model has dropped to 71.8121%. The result is as follows:

```
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances          214               71.8121 %
Incorrectly Classified Instances        84                28.1879 %
Kappa statistic                   0.4156
Mean absolute error                     0.3183
Root mean squared error               0.4139
Relative absolute error                 66.1541 %
Root relative squared error             84.3969 %
Total Number of Instances               298
```

And the confusion matrix is as follows:

```
=== Confusion Matrix ===
  a   b   <-- classified as
 79   41 |   a = 0
 43  135 |   b = 1
```

Random Forest

Overfitting problems can be seen in a single tree, which can result in poor predicting power. To address this problem, random forest is tried to gain more accurate results. 100 trees are constructed in a random forest. All 22 attributes were taken into account to build the model to predict the outcome of the horses. This random forest has an accuracy of 78.8591%, where 235 instances are correctly classified and 63 are incorrectly classified. The confusion matrix of this model is shown below:

```
=== Confusion Matrix ===
  a   b   <-- classified as
 85   35 |   a = 0
 28  150 |   b = 1
```

113 instances are classified as "died", while 28 are incorrectly classified. 185 instances are classified as "lived", and 35 are incorrectly classified. It could be seen that this model is better at predicting the horse which survived, but this isn't such a big difference compared to the previous models.

Discretize Numeric Attributes

5 of the 22 attributes are continuous. By applying unsupervised discretization and converting each of them into 5, 10, 15, 20 bins using equal-width-binning, we found that binning into 15 bins result in a higher accuracy. After binning continuous attributes into 15 bins, a new model is generated. The accuracy of the model is 83.2215%, where 248 instances are correctly classified and 50 are incorrectly classified. This result is better than the previous one without binning, which could be concluded that discretizing the attributes does help in improving the accuracy of the model. The confusion matrix is as follows:

```
=== Confusion Matrix ===
  a   b   <-- classified as
 89   31 |   a = 0
 19  159 |   b = 1
```

108 instances are classified as “died”, while 19 are incorrectly classified. 190 instances are classified as “lived”, and 31 are incorrectly classified. The accuracy of class “died” and “lived” both improved and reach 82.4% and 83.7% each. This model is better at predicting the outcomes of the horses.

K Nearest Neighbor Classification

```
Test mode: 10-fold cross-validation
=== Classifier model (full training set) ===
IB1 instance-based classifier
using 12 nearest neighbor(s) for classification
Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances        223               74.8322 %
Incorrectly Classified Instances       75               25.1678 %
Kappa statistic                         0.4818
Mean absolute error                     0.3125
Root mean squared error                 0.3979
Relative absolute error                64.9404 %
Root relative squared error            81.1213 %
Total Number of Instances             298
```

```
=== Confusion Matrix ===
   a   b   <-- classified as
  86  34 |   a = died
  41 137 |   b = lived
```

K-nearest neighbors algorithm is a supervised learning classifier that uses proximity to make classifications or predictions about the grouping of an individual point. This algorithm is a proud member of the “lazy learning” models which means that it only stores a training dataset instead of undergoing a training stage. Since it heavily relies on memory it is also called Instance-based (IB1) or memory-based learning. The primary goal of this classifier is to determine the nearest neighbors of a given query point so we can assign a class label to that point.

The value of the k defines the neighbors that will be checked to determine the classification of a specific query point. For building our model, we initially take k=1, here the query point will be classified to the same class as its single nearest neighbor. Using 10-fold cross-validation, the accuracy that we get using 1 nearest neighbor for classification is 72.48%. Choosing the value of k is a crucial step, as it can lead to overfitting or underfitting of the model depending on the data. Having lower values of k means high variance and low bias whereas higher values of k leads to having high bias and lower variance.

Upon increasing the value of k from 1 to 12, we see that the accuracy of the model increases to 74.83%. On analyzing the confusion matrix for the kNN algorithm we understand that the 127 instances are classified as “died” out of which 41 are incorrectly classified. Whereas, 171 instances are classified as “lived” out of which 34 are incorrectly classified. However, running this updated model with a higher k value we achieve higher accuracies.

The key takeaways from the kNN algorithm are, the simplicity and accuracy of the classification make it easier to be used by early data scientists. The algorithm is well equipped to account for any training data since all of it is stored in the memory. It is a very minimalist algorithm in terms of the hyperparameters required, since it only requires a k value and distance metric which is relatively low from other machine learning algorithms. On the other hand, it is very time and resource intensive for large data in terms of computation since it takes up more memory and storage space. In our use case, it was fairly easy to run the algorithm with the small size of instances we had.

Conclusion

After completing the various data analysis described previously, we were able to make a handful of conclusions. Starting with binning Naive Bayes we could create a model to predict the outcome of the animals with 76.5% accuracy. This is predicted with type I error 27/298 = 0.091 and type II error 0.144. Since a 76.5% accuracy wasn't as high as we wanted we also conducted a decision tree and random forest analysis as well as K nearest neighbors. After completing these various analyses, the highest accuracy we achieved was 83.2% through the random forest by binning continuous attributes into 15 bins. No matter which data analytics we conducted, the accuracy of each increased after further processing the data and adjusting the parameters of the analysis.

Through Apriori and association rules we classified 0 as 'died' and 1 as 'lived'. The rules we gathered through WEKA almost exclusively predicted the outcome as 0.5-infinity, otherwise known as 'lived'. Through the rules, we were able to conclude that the animals that didn't have surgery, were adults, and had a pulse less than 58 were surviving their veterinarian visits. While the vets can't control the ages of the horses they see, they can take steps to avoid unnecessary surgery, and lower the animal's pulse rate.

Veterinarians can use the information we gathered as guidelines to prepare for the likely outcomes of each animal. Since the stakes are high, with the outcome being each animal's survival, we wouldn't recommend the veterinarian rely too heavily on our data analysis. Each situation is different, but the information described here can help them prepare for what next steps they need to take.

References

IBM, KNN. “What Is the K-Nearest Neighbors Algorithm?” IBM, 2022, https://www.ibm.com/topics/knn.