Julia Tzu-Ya Weng U07487022

### ***Data Mining for Scientific Applications***

Course No. CSE-40770

***Laboratory Assignment II:***



***To download additional .arff data sets go to:***

[**http://repository.seasr.org/Datasets/UCI/arff/**](http://repository.seasr.org/Datasets/UCI/arff/)

or alternatively

<http://www.hakank.org/weka/>

1. Use the following learning schemes to analyze the zoo data (in zoo.arff):

|  |  |
| --- | --- |
| OneR | - weka.classifiers.OneR |
| Decision table | - weka.classifiers.DecisionTable -R |
| C4.5 | - weka.classifiers.j48.J48 |
| K-means | - weka.clusterers.SimpleKMeans |

Try using reduced error pruning for the C4.5. Did it change the produced model? Why?

For K-means, for the first run, set ***k***=10. Adjust as needed. What was the final number of ***k***? Why?

|  |
| --- |
| For this question, I used the NumericToNominal filter under “unsupervised” on the leg attribute. I split the data into 60% train; 40% test because for the last assignment I used 10-fold cross validation, and for this question I would like to practice using 60-40.  **OneR:**   * uses the “animal” attribute to determine the type of animal an instance belongs to. * performance on the test dataset   + classifier correctly classifies 17 of the 40 test instances (42.5%) and misclassifies the remaining 23 (57.5%)   + kappa statistics = 0.0564, mean absolute error = 0.1643; precision and recall are 0 for fish, birds, invertebrates, insects, and reptiles. The confusion matrix table shows that most of these are incorrectly classified as mammals. * OneR has very poor performance on this dataset.   **Decision Table:**   * uses the “milk”, “fins”, “legs”, “tails” attribute values to form 15 rules to determine the type of animal an instance belongs to. * performance on the test dataset:   + the classifier correctly classifies 35 of the 40 test instances (87.5%) and misclassifies the remaining 5 (12.5%)   + kappa statistic= 0.8215, mean absolute error=0.1743; weighted area under ROC (AUC) is 0.92 * Decision table looks like good news at first, but precision and recall are 0 for invertebrates. This model is unreliable for classifying invertebrates from other types of animals.   **C4.5** (Fig. 1):   * The tree has 9 leaves, starting with the “feather” attribute at the top. * performance on the test dataset   + classifier correctly classifies 38 of the 40 test instances (95%) and misclassifies the remaining 2 (5%)   + kappa statistic= 0.9345, mean absolute error = 0.0157; the only thing keeping the AUC at 0.971 seems to be the 2 insect instances that are misclassified into invertebrates. * Best performing model so far; however, the tree looks big at the bottom, particularly at the branches from the legs node.   Screen Shot 2017-07-21 at 4.12.39 PM.png  Figure 1. Decision tree model for the classification of the zoo.arff dataset. Reduced error pruning was not applied.  **C4.5 with reduced error pruning** (Fig. 2):   * numFolds set at the default 3, one fold of training data for pruning and the rest for growing the tree. * The tree has reduced to 7 leaves, still with the “feather” attribute being the most important attribute. The model has been changed because the “airborne” and “legs” attributes were not included in the tree. The tree is smaller and less complex. * performance on the test dataset:   + classifier correctly classifies 34 of the 40 test instances (85%) and misclassifies the remaining 6 (5%)   + kappa statistic= 0.04, mean absolute error = 0.1454; classification precision and recall for invertebrates have improved, the overall AUC is increased to 0.978, but 6 insects are misclassified as invertebrates. Precision and recall are 0 for the insect class. * There seems to be a trade-off between pruning and accuracy with this model. It is like, in order to simplify the model, it has “given up” on correctly making a decision about whether the animal is an insect or not.   Screen Shot 2017-07-21 at 4.15.25 PM.png  Figure 2. Decision tree model with reduced error pruning for the classification of the zoo.arff dataset.  **k-means clustering:**  side note: There is a classes to clusters evaluation setting for k-means clustering, but that does not give me the option to use training and testing, so I still chose to use 60-40 split of training and testing, respectively, in this case. I also removed the class attribute “type”.   * I tried k = 10 down to k = 2 * For evaluation, I look at the number of instances assigned to each cluster and use the within cluster sum of squared error (SSE) as a reference. * Fig. 3 shows how SSE changes with k .   Chart  Chart  Figure 3. A plot within group SSE vs. k in the training (top) and testing (bottom) sets.   * I think I would choose k = 4 as the final number of k because it looks like the “elbow” in the plot. We do not want k to be too big because the data will be too separated to allow us to make meaningful interpretations. When k = 4, the four clusters are roughly equally represented, cluster 0 has 11 instances, cluster 1 has 11 instances, cluster 2 has 15 instances, and cluster 3 has 23 instances. * Another potential k is 6. It looks like another elbow, and within group SSE is smaller in this case, but one of the clusters in this case is very underrepresented, having only 3 instances. This makes me worry about potential overfitting problem. |

1. Use the following learning schemes to analyze the Automobile (http://archive.ics.uci.edu/ml/datasets/Automobile) data set.

|  |  |
| --- | --- |
| Linear regression | - weka.classifiers.LinearRegression |
| M5' | - weka.classifiers.M5' |
| Regression Tree | - weka.classifiers.M5' |
| K-means clustering | - weka.clusterers.SimpleKMeans |

A) How many leaves did the Model tree produce? Regression Tree? What happens if you change the pruning factor?

How many clusters did you choose for the K-means method? Was that a good choice? Did you try a different value for ***k***?

|  |
| --- |
| For this question, I used 60-40 training to testing split. The .csv file was converted to .arff by manually entering the attributes according to the [imports-85.names](http://archive.ics.uci.edu/ml/machine-learning-databases/autos/imports-85.names) file found on the attached link in the question, and follow the .arff format. The question does not say which class attribute we are looking at, so I assumed that we are predicting the price of the car because this is how the dataset was used in the past as indicated by the document.  For different pruning factors, I changed either the unpruned = T/F or the minNumInstances. Everything else was kept at default. For k-means clustering, I changed the numClusters. EVerything else was kept at default.  **M5P:**  The model tree produces four leaves, each corresponding to the result generated by a LM model:  curb-weight is the most important attribute  curb-weight <= 2665.5 : LM1 (125/11.377%)  curb-weight > 2665.5 :  | engine-size <= 182 :  | | make=audi,volvo,bmw,porsche,mercedes-benz,jaguar <= 0.5 : LM2 (40/25.399%)  | | make=audi,volvo,bmw,porsche,mercedes-benz,jaguar > 0.5 : LM3 (19/15.729%)  | engine-size > 182 : LM4 (17/26.355%)  ===Result Summary on Test ===  Correlation coefficient 0.9561  Mean absolute error 1553.3364  Root mean squared error 2614.2608  Relative absolute error 26.2064 %  Root relative squared error 30.0693 %  Total Number of Instances 82  **Linear regression:**  Prediction of price was derived from a linear regression model that considers, in order of importance:  symboling, make, fuel-type, aspiration, body-style, engine-location, wheel-base, length, width, height, curb-weight, engine-type, num-of-cylinders, engine-size, fuel-system, bore, stroke, compression-ratio, horsepower, peak-rpm, highway-mpg  The model did not include normalized-losses, city-mpg, num-of-doors, and drive-wheels.  === Result Summary on Test===  Correlation coefficient 0.9419  Mean absolute error 2285.8637  Root mean squared error 3101.5284  Relative absolute error 38.5649 %  Root relative squared error 35.6738 %  Total Number of Instances 82  **Regression Tree:** 9 leaves!  curb-weight <= 2665.5 :  | curb-weight <= 2291.5 :  | | curb-weight <= 2121 :  | | | length <= 160.75 : LM1 (27/8.596%)  | | | length > 160.75 : LM2 (17/7.78%)  | | curb-weight > 2121 : LM3 (27/11.735%)  | curb-weight > 2291.5 :  | | fuel-system=spfi,4bbl,mfi,idi,mpfi <= 0.5 : LM4 (21/13.739%)  | | fuel-system=spfi,4bbl,mfi,idi,mpfi > 0.5 :  | | | make=volkswagen,nissan,mazda,saab,peugot,alfa-romero,mercury,audi,volvo,bmw,porsche,mercedes-benz,jaguar <= 0.5 : LM5 (16/13.76%)  | | | make=volkswagen,nissan,mazda,saab,peugot,alfa-romero,mercury,audi,volvo,bmw,porsche,mercedes-benz,jaguar > 0.5 : LM6 (17/28.342%)  curb-weight > 2665.5 :  | engine-size <= 182 :  | | make=audi,volvo,bmw,porsche,mercedes-benz,jaguar <= 0.5 : LM7 (40/32.152%)  | | make=audi,volvo,bmw,porsche,mercedes-benz,jaguar > 0.5 : LM8 (19/40.389%)  | engine-size > 182 : LM9 (17/62.212%)  === Result Summary on Test===  Correlation coefficient 0.8422  Mean absolute error 3419.2143  Root mean squared error 5396.8391  Relative absolute error 57.6858 %  Root relative squared error 62.0745 %  Total Number of Instances 82  **Changing the pruning factor in the M5P tree model**:  I do not understand the question because M5P does not offer a pruning factor as an option. Do you mean “minNumInstances” or the “unpruned”? Anyways, I did both, and changed minNumInstances from 2.0 to 10.0, each change incremented by 1. Default is 4.0.  Below is the result of an unpruned M5P tree using 74 rules!  === Result Summary on Test ===  Correlation coefficient 0.9491  Mean absolute error 1695.6078  Root mean squared error 2718.8298  Relative absolute error 28.6067 %  Root relative squared error 31.272 %  Total Number of Instances 82  The performance is better (slightly increased correlation coefficient and decreased mean absolute error) compared to the previously pruned tree. However, the model incorporates 74 rules, so the tree is deep and is likely to have overfitting issues.  As the minimum number of instances allowed at a leaf node increases from 2 to 5, correlation coefficient and mean absolute error increases and decreases, respectively,. Then, at minNumInstances = 6, mean absolute error starts to increase, whereas correlation coefficient starts to decrease.  **Changing the pruning factor in the regression tree model**:  Below is the result summary of the unpruned tree. Compared to the M5P tree. This tree uses 74 rules as well, but there are only slight changes in correlation coefficient and mean absolute error.  === Result Summary on Test===  Correlation coefficient 0.8451  Mean absolute error 3388.1121  Root mean squared error 5373.7179  Relative absolute error 57.161 %  Root relative squared error 61.8086 %  Total Number of Instances 82  The regression tree seems to be less “sensitive to changes” in pruning and the minimum number of instances required in a leaf node. The number of rules stays the same as minNumInstances increases from 2 to 10. Correlation coefficient decreases a little starting at minNumInstances = 8, and the error is also decreased slightly.    **k-means clustering:**  I removed the price attribute. I tried k = 2 to k = 13. Figure 4 shows the relationship between the within group SSE and k in the testing and training set.  Chart  Chart  Figure 4. A plot within group SSE vs. k in the training (top) and testing (bottom) sets.  There seems to be two elbows from the testing data, one at k = 4, and the other one at k = 6. k = 7 has a roughly equal level with k = 6. After k = 7, the line begin to level off relatively slowly. Based on results of the testing data, it seems that SSE decreases at a slower rate after k = 7. k = 7 might be a good choice for the model. |

B) Now perform the same analysis on the bodyfat.arff data set.

|  |
| --- |
| **M5P:**  The model tree uses 6 rules, but focused on different levels of the density attribute value.  Density <= 1.056 : LM1 (130/2.308%)  Density > 1.056 :  | Density <= 1.066 : LM2 (43/0.408%)  | Density > 1.066 :  | | Density <= 1.078 : LM3 (46/16.847%)  | | Density > 1.078 :  | | | Density <= 1.083 : LM4 (15/0.394%)  | | | Density > 1.083 :  | | | | Density <= 1.092 : LM5 (14/0.295%)  | | | | Density > 1.092 : LM6 (4/12.453%)  ===Result Summary on Test ===  Correlation coefficient 0.9911  Mean absolute error 0.5665  Root mean squared error 1.0652  Relative absolute error 8.695 %  Root relative squared error 13.4171 %  Total Number of Instances 101    **unpruned M5P** uses 33 rules based on the density, weight, and chest attributes. Though correlation coefficient is increased, and mean absolute error, decreased, the tree is deep and complex and may run into overfitting problem. Below is the result summary on the testing dataset.  Correlation coefficient 0.993  Mean absolute error 0.326  Root mean squared error 0.9624  Relative absolute error 5.003 %  Root relative squared error 12.1216 %  Total Number of Instances 101  Ichanged minNumInstances from 2.0 to 10.0, each change incremented by 1. When minNumInstances = 2 and 3, the result is the same as when minNumInstances = 4. When minNumInstances >= 5, the number of rules decreases to 2 and the model just looks at whether density <= 1.056 or >1.056. Correlation coefficient increases and mean absolute error decreases compared to when minNumInstance < 5.  **Linear regression:**  Prediction of class was derived from the following linear regression model:  class = -410.2167 \* Density + 0.0124 \* Age + 0.0253 \* Chest + 0.0314 \* Abdomen + 446.1513  === Result Summary on Test===  Correlation coefficient 0.9907  Mean absolute error 0.5634  Root mean squared error 1.0852  Relative absolute error 8.6476 %  Root relative squared error 13.6685 %  Total Number of Instances 101  **Regression Tree:** 16 rules that seem to chop up the density into many different ranges as rules.  Density <= 1.056 :  | Density <= 1.045 :  | | Density <= 1.035 : LM1 (37/43.294%)  | | Density > 1.035 :  | | | Density <= 1.039 :  | | | | Density <= 1.038 : LM2 (10/4.606%)  | | | | Density > 1.038 : LM3 (5/0.896%)  | | | Density > 1.039 :  | | | | Density <= 1.042 : LM4 (14/4.008%)  | | | | Density > 1.042 : LM5 (10/2.836%)  | Density > 1.045 :  | | Density <= 1.05 :  | | | Density <= 1.047 : LM6 (9/4.148%)  | | | Density > 1.047 : LM7 (13/2.933%)  | | Density > 1.05 :  | | | Density <= 1.053 :  | | | | Density <= 1.052 : LM8 (12/2.993%)  | | | | Density > 1.052 : LM9 (10/1.71%)  | | | Density > 1.053 : LM10 (10/3.05%)  Density > 1.056 :  | Density <= 1.066 :  | | Density <= 1.062 :  | | | Density <= 1.059 : LM11 (12/3.737%)  | | | Density > 1.059 : LM12 (14/4.351%)  | | Density > 1.062 :  | | | Density <= 1.063 : LM13 (4/0.847%)  | | | Density > 1.063 : LM14 (13/4.135%)  | Density > 1.066 :  | | Density <= 1.078 : LM15 (46/23.385%)  | | Density > 1.078 : LM16 (33/35.635%)  === Result Summary on Test===  Correlation coefficient 0.9436  Mean absolute error 2.0838  Root mean squared error 2.9225  Relative absolute error 31.982 %  Root relative squared error 36.811 %  Total Number of Instances 101  **The unpruned regression tree** uses 33 rules based on the density, chest, and weight attributes, and has slightly increased correlation coefficient and decreased mean absolute error, but the tree is deep and complex.  === Result Summary on Test===  Correlation coefficient 0.9587  Mean absolute error 1.951  Root mean squared error 2.6927  Relative absolute error 29.9437 %  Root relative squared error 33.9167 %  Total Number of Instances 101  The number of rules stays the same as minNumInstances increases from 2 to 10. Correlation coefficient increases slightly when minNumInstances = 10, and the mean absolute error is decreased by a little. There is not much change to the overall performance when the setting for minNumInstances is changed    **k-means clustering:**  I removed the class attribute. I tried k = 2 to k = 13. Figure 5 shows the relationship between the within group SSE and k in the testing and training set.  Chart  Chart  Figure 5. A plot within group SSE vs. k in the training (top) and testing (bottom) sets.  From the training result, it seems that k = 6 and 8 could be the elbows. Looking at the testing result, within group SSE decreases at a slower rate after k = 6. k = 6 might be a choice for the model. |

1. Use a k-means clustering technique to analyze the iris data set. What did you set the ***k*** value to be? Try several different values. What was the random seed value? Experiment with different random seed values. How did changing of these values influence the produced models?

|  |
| --- |
| For this question, I tried k = 2 to k = 10 for seed = 10. Figure 6 shows the relationship between the within group SSE and k in the testing and training set. From the training set result, it seems that the elbows are at k = 4 and 6. Looking at the training and testing set results, the two agree on k = 4 being the elbow. Therefore, k = 4 is probably a good choice of k for clustering the iris data set.  Chart  Chart  Figure 6. A plot within group SSE vs. k in the training (top) and testing (bottom) sets.  For k = 4, I tried to run k-means with seed = 1, 10, 100. Seed values change the initial cluster center of each cluster. This changes the clustering results in terms of the number of iterations to build the model, percentage of clustered instances in each cluster, the within cluster SSE. It seems like keeping the seed constant ensures reproducibility of results. |

1. Produce a hierarchical clustering (COBWEB) model for iris data. How many clusters did it produce? Why? Does it make sense? What did you expect?

Change the acuity and cutoff parameters in order to produce a model similar to the one obtained in the book. Use the classes to cluster evaluation – what does that tell you?

|  |
| --- |
| I removed the class attribute before I did the clustering, I used 60-40%, train vs. test split. Everything else was kept at default: seed = 42, acuity = 1, cutoff ~0.00282  === Run information ===  Scheme: weka.clusterers.Cobweb -A 1.0 -C 0.0028209479177387815 -S 42  Relation: iris-weka.filters.unsupervised.attribute.Remove-R5  Instances: 150  Attributes: 4  sepallength  sepalwidth  petallength  petalwidth  Test mode: split 60% train, remainder test  === Clustering model (full training set) ===  Number of merges: 0  Number of splits: 0  Number of clusters: 3  node 0 [150]  | leaf 1 [97]  node 0 [150]  | leaf 2 [53]  Time taken to build model (full training data) : 0 seconds  === Model and evaluation on test split ===  Number of merges: 0  Number of splits: 0  Number of clusters: 3  node 0 [90]  | leaf 1 [35]  node 0 [90]  | leaf 2 [55]  I do not understand why I set it up for 60-40 split, the training process still dump all the full training data into building the model. There are essentially 2 clusters (leaf 1 and leaf 3). There is a root node (node 0) that split into two subclusters: leaf 1 and leaf 2. Subcluster 1 has 97 instances; subcluster 2 has 53 instances.  The clustering result does not make sense to me. I expect there to be at least three subclusters because there are three different classes of iris.  **Changing the acuity and cutoff parameters:**  Decreasing acuity increases the number of clusters, but each cluster has fewer number of instances. Decreasing the cutoff parameters does not have much of an effect until the value approaches 0, at which point the tree suddenly got really big.  I decided the best I could get as a similar match to Fig. 6.18(B) in first edition of the textbook is the hierarchical cluster in Fig. 7, where acuity = 0.4 and cutoff = 0.00282.  Screen Shot 2017-07-25 at 3.28.42 PM.png  **Classes to cluster evaluation:** For this step, I put the class attribute back in the preprocessing, so that WEKA can assign classes to the clusters after the clustering. This setting shows me the classification error:  Class attribute: class  Classes to Clusters:  3 4 6 7 8 9 <-- assigned to cluster  0 0 0 0 2 48 | Iris-setosa  23 23 4 0 0 0 | Iris-versicolor  1 13 24 12 0 0 | Iris-virginica  Cluster 3 <-- No class  Cluster 4 <-- Iris-versicolor  Cluster 6 <-- Iris-virginica  Cluster 7 <-- No class  Cluster 8 <-- No class  Cluster 9 <-- Iris-setosa  Incorrectly clustered instances : 55.0 36.6667 % |