DATA MINING: CAR EVALUATION

Juan Lopez

University of Miami
34
jxl1581@miami.edu

ABSTRACT

This project is to determine the viability of a car in terms of different attributes including the buying price, maintenance price, etc. We are first going to determine the type of attributes we are dealing with. Do a simple classification learning algorithm in terms of a Decision Tree on a training:testing split 80:20. After this using the Apriori Algorithm we will generate association rules and based on these findings we report the rules and now we have a ruleset to determine the viability of the cars.

Index Terms— One, two, three, four, five

1. CLASSIFYING ATTRIBUTES

The car_evaluation.csv is split into 7 attributes in terms of buying with values (vhigh,high,med,low) , maintenance with values (vhigh,high,med,low) , # of doors (2,3,4,5more) , # of capacity (2,4,more) , size of Luggage capacity (small,med, big), How safe the car is (low,med,high). And then we have our class label (unnacceptable, acceptable, good, very good). As we can see here Each attribute is a bit different from each other. Buying (ordinal, interval) because it gives a specific values to the price of a car if it is very high or low with a difference in order. Maintenance is also (ordinal, interval) because it does the same with the buying price though it is in terms of maintenance. # of doors (ordinal, interval) because it does have an order with no true 0. Capacity is (nomial) because I believe there is no true # of people that can actually fit so the number is a recommendation to how many can fit but not a true ordinal value of it. Luggage capacity is (ordinal, interval) with no true 0. Safety (ordinal, interval) for same reason. Class label (nomial) because there is no true way to measure the difference between them.

2. DATA PROCESSING

This data is given to us in values in which a computer might find it hard to understand… English. This is why it is very important for ordinal attributes to given them values in terms of #. An example of this can be for buying price instead of vhigh we can give this value 4 and for low we can give that the value of 1. In terms of Pre-Processing data I can sit and rewrite all the instances with this but using Python I automated this system and created a new the csv file named. IntegerCarEvaluation.csv. An instance example from this Data Set can look like. There is also no missing values so professor Shy made it easier for us. Usually in a real life example it would be a little more complicated than this when it comes to also filling up missing values or data that could be bad. In turn we would have to add a further step before this and fix all the missing values with appropriate values this can be a NONE value or even the value that is most common. Though it can lead to a certain problem if all of them are equally common. There could be a skew in the training set. An example on how we turned all the values into integer format is given below.

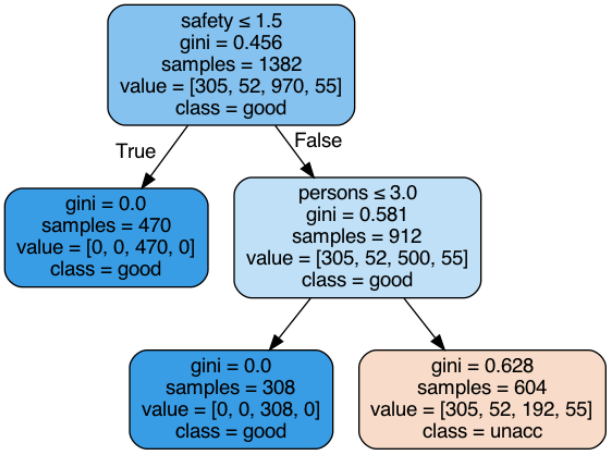
Buying	Maint	Doors	Persons	Lug-Boot	Safety	Class
4	4	2	2	1	1	unacc

As we can see from before we can the values for buying, maintenance, capacity, and safety have changed instead of the values being strings like buyings (vhigh,high,med,low) or maintenance being (vhigh,high,med,low) we changed these values into more appropriate fitting values for the Decision Tree.

3. DECISION TREE CLASSIFIER

Using libraries like numpy, pandas, I was able to create a datagram reading from the .csv file that I just created based off integer values. After this I tested out a simple xtrain and ytrain from the specific columns in which I know was the class labels and the other that described what that class label consisted of. After this using the train_test_split function I created 4 different variables called xtrain ytrain xtest ytest in which I used these to train and test my Decision Tree model. Using the Decision tree with maxLeafNodes as 3 and the random state to be 0. I trained my Dataset with my x and y training sets. I predicted what my x test would classify based on this data set and and compared it to the actual values. I discovered that this gave me a 75.4335%

which is an accurate prediction. We can improve this accuracy by tuning the decision tree parameters. Decreasing the data split into 70:30 provided better results at 77.26% accuracy. Increasing it to a 40:60 has turned the accuracy to a 77.02% accuracy. 50:50 split made this to a 77.3% accuracy but once we passed the 50:50 mark and jumped into 40:60 the accuracy began to decrease. After this I used export_graphviz to provide me with a visual that I could display here.



Analyzing these visuals we can clearly see that the first node in the tree says that if the safety<= 1.5 the class is good. Else we will do the next check for persons<=3.0. The only real issue I've discovered is the consistence based on logic. I don't think that there is a value 1.5 safety as it was more of a nominal attribute before hand with a string for a value. Now using integers it lacks the whole number sense. But it still make's sense to know that the values are either 1 or 2 which stays consistent with Decision Tree.

4. ASSOCIATION LEARNING

For my project 1B) I've decided to take upon the task of applying an association rule algorithm to the data set above. This is used to create "if-then" rules for this data set. Instead of simply just looking at the decision tree above and saying the if and then rules we can humanly see. I will be using the Apriori Algorithm to establish these rules. There is some prior mathematics we have to include before we are able to talk about the implementation of the algorithm. These are

$$support(x) = P(x)$$

$$support(x \rightarrow y) = P(x \rightarrow y)$$

$$confidence(x \rightarrow y) = \frac{P(x \rightarrow y)}{P(x)}$$

$$lift(x \rightarrow y) = \frac{confidence(x \rightarrow y)}{support(x)}$$

These are all used to calculate the association rules that ultimately would be able to determined the the rules of the dataset provided. The difference between having to do the decision tree and the association rules came to the use of the different learning libraries already available in Python formulating the csv file from strings to integers for decisions trees was something not had to be done for the association rules. It was able to use the string values to promote a significant more clear rule set. Running through the algorithm it was easy to decipher how the appropriate equations at the time provided these rules sets. Using numpy, pandas, matplotlib, and apyori libraries I was able to turn the datagram data from the csv file into a list. So we can pass it to the apriori function. Running the algorithm we came up with these rules and their calculations based on their being a minimum support of 3% minimum confidence of 20% and minimum lift of 3. There were also a set of lengths of attributes that could describe these rules which I also set from range of (0,4) this means that a rule can have be any number from 0-4. This means we can look at the example rule of. Sam likes the colors green and eggs -> Sam likes Ham. As we can see two things could provide a rule of 1 outcome or so. The following rules were the results

Rule: big -> vgood
Support: 0.00578368999421631
Confidence: 1.0
Lift: 3.001736111111111
==========

Rule: big -> high
Support: 0.00578368999421631
Confidence: 1.0
Lift: 4.802777777777778
==========

Rule: big -> low
Support: 0.004626951995373048
Confidence: 0.7999999999999999
Lift: 3.842222222222222
==========

Rule: big -> med
Support: 0.003470213996529786
Confidence: 1.0
Lift: 3.001736111111111
==========

Rule: big -> 3
Support: 0.00578368999421631
Confidence: 0.6666666666666666
Lift: 3.2018518518518517

==========
Rule: 3 -> vgood
Support: 0.00578368999421631
Confidence: 0.6666666666666666
Lift: 3.2018518518518517
==========
Rule: big -> 4
Support: 0.014459224985540775
Confidence: 0.38461538461538464
Lift: 3.694444444444445
==========
Rule: small -> good
Support: 0.008675534991324466
Confidence: 0.5
Lift: 3.001736111111111
==========
Rule: small -> good
Support: 0.003470213996529786
Confidence: 1.0
Lift: 3.001736111111111
==========
Rule: big -> good
Support: 0.013880855986119144
Confidence: 0.34782608695652173
Lift: 3.1322463768115942
==========
Rule: big -> high
Support: 0.01850780798149219
Confidence: 0.4923076923076923
Lift: 4.433333333333333
==========
Rule: big -> med
Support: 0.013880855986119144
Confidence: 0.36923076923076925
Lift: 3.325
==========
Rule: big -> high
Support: 0.01156737998843262
Confidence: 0.3076923076923077
Lift: 4.433333333333334
==========
Rule: big -> low
Support: 0.009253903990746095
Confidence: 0.24615384615384614
Lift: 3.5466666666666664
==========
Rule: small -> good
Support: 0.012145748987854251
Confidence: 0.7
Lift: 3.3619444444444446
==========
Rule: med -> good
Support: 0.01735106998264893
Confidence: 0.4347826086956522
Lift: 3.0312061711079945
==========
Rule: high -> vgood
Support: 0.016194331983805668
Confidence: 0.4307692307692308
Lift: 3.879166666666667
==========

After this I was able to either return a data frame .csv file with the following rules or the printed version.

5. RESULTS

Analyzing these results we can say from a human perspective that it makes almost perfect sense. We can see that for example a rule is seen that if it has a big luggage room -> 4. This idea is seen if there is a big luggage room the car should also contain the same amount of people it needs to have luggage for. This makes complete sense. The Support is 1.4% with a confidence of 38% and a lift of 3.694. Honestly something I can do to improve the human ability to read this information is changing the format so you can understand what category the rules are talking about when it says med for example which is in 4 different categories or 2,4 which are in 2 other categories. Having a difference between which specific attribute we are looking at would make this easier.

6. CONCLUSION

To run the file there are 3 different programs in which we normalize the data from strings into integers this is due to the data pre-processing we have to do. This dataset specifically doesn't have missing values so domain knowledge is not as applicable. But to analyze the data afterwards we can use domain knowledge to verify our results intuitively. After applying the pre-process I was able to split the data to be able to run the decision tree function based on certain % splits. I tried different percent splits and got about an average of 75% correctness on testing. This is sufficient enough to call this decision tree classifier as correct. Depending on the data this can change and % correct testing could be altered. To provide a better association learning I used the string .csv file to provide better looking results to analyze. The algorithm I used to create these association rules was the apriori algorithm which uses support(x->y) confidence (x->y) and lift (x->y) to appropriately determine rule sets. This is seen by my results being posted with relations. To further talk about my results I would like to say that these results could be altered to provide more general relations or more specific relations. Changing the parameters of minimum confidence or minimum support we can eliminate relations that might not meet the minimums. I believe the association rules intuitively are what a normal family Household would look for in a car and evaluate it by before purchasing. Focusing on these relations car manufactures can prioritize these models and build more of them.