Assignment 5

Sophia J Quinton

Grand Canyon University

DSC-540: Machine Learning

Dr. Aiman Darwiche

01 December 2021

Assignment 5

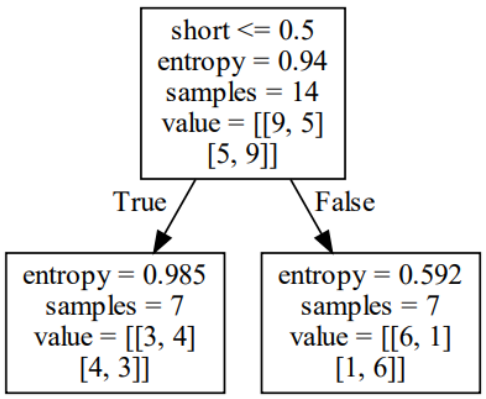
Part 1

For the decision tree part of the assignment, the information gain was calculated by hand in python. To get the information gain, AskPython (2020) was used as a guide. The information gain for the three x variables in terms of the y variable are given. The information gain reports

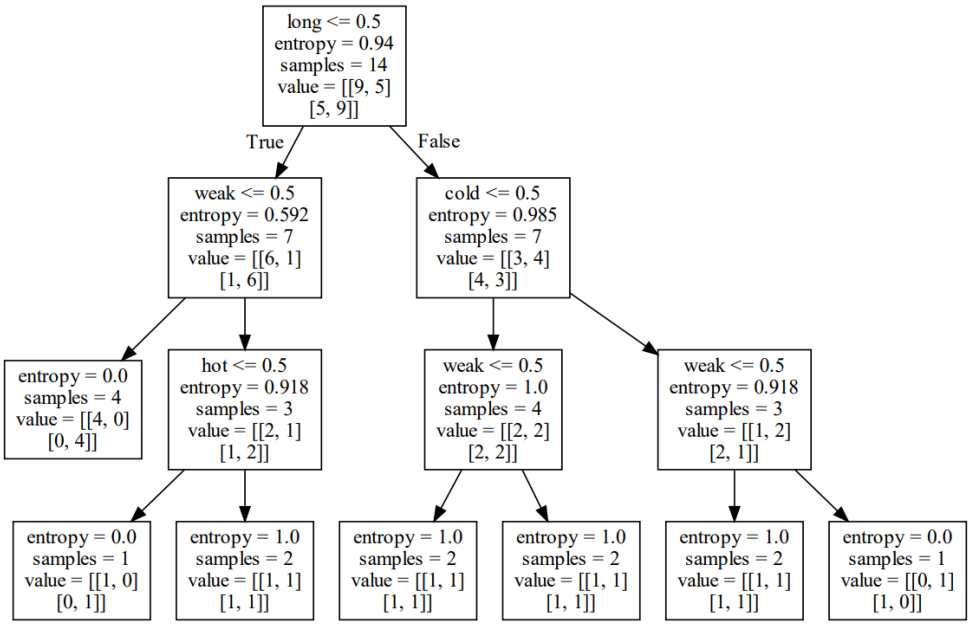
```
4          cool    weak          short         yes

In [5]: #calculate information gain
        ##temperature
        H_temp_hot = -1 * ((2/4) * np.log2((2/4)) + (2/4) * np.log2((2/4)))
        H_temp_mild = -1 * ((4/6) * np.log2((4/6)) + (2/6) * np.log2((2/6)))
        H_temp_cool = -1 * ((3/4) * np.log2((3/4)) + (1/4) * np.log2((1/4)))
        Net_Entropy = (4/14) * H_temp_hot + (6/14) * H_temp_mild + (4/14) * H_temp_cool
        Total_Reduction = 1-Net_Entropy
        print("The information gain for temperature is: ", Total_Reduction)

        The information gain for temperature is:  0.08893660698832373

In [6]: ##wind
        H_wind_weak = -1 * ((6/8) * np.log2((6/8)) + (2/8) * np.log2((2/8)))
        H_wind_strong = -1 * ((3/6) * np.log2((3/6)) + (3/6) * np.log2((3/6)))
        Net_Entropy_wind = (8/14) * H_wind_weak + (6/14) * H_wind_strong
        Total_Reduction_Wind = 1 - Net_Entropy_wind
        print("The information gain for wind is: ", Total_Reduction_Wind)

        The information gain for wind is:  0.10784107173763835

In [7]: ##traffic
        H_traffic_long = -1 * ((3/7) * np.log2((3/7)) + (4/7) * np.log2((4/7)))
        H_traffic_short = -1 * ((6/7) * np.log2((6/7)) + (1/7) * np.log2((1/7)))
        Net_Entropy_traffic = (7/14) * H_traffic_long + (7/14) * H_traffic_short
        Total_Reduction_traffic = 1 - Net_Entropy_traffic
        print("The information gain for traffic is: ", Total_Reduction_traffic)

        The information gain for traffic is:  0.21154954269171045
```

which variable is important for determining the overall prediction of the dependent variable (Gopal, 2019). In this case, the traffic score is 0.211 which is the highest. This means that the amount of traffic is likely to reveal more information about whether someone is likely to drive or not.

For the decision tree model. The C5.0 tree was chosen (this is determined by entropy as the criterion) (Pedregosa et. al., 2011). The C5.0 is a newer version of the C4.5 algorithm that uses information gain for the best attributes (AskPython, 2020). It is also useful with categorical variables (Larose & Larose, 2019).



For this tree, based on the information gain, the root node is the traffic and whether or not the traffic is short. The root was generated from the random function and is shown in the figure with the short tree. Then the tree was



fully grown in the graph with the large tree until the nodes are pure. No arguments were used in the second tree other than the criterion argument. The graphs were generated with the graphviz library (Graphviz, nd). The results indicate that the traffic length is the greatest predictor. The if the wind is weak the car will be driven. If the wind is not weak but it is hot, the car will be driven, but this is represented by one point. Then when the traffic is long, but the weather is cool and the wind is weak, the car will be driven. This is misleading because data is missing here, so the algorithm is making the assumption.

In terms of generating the tree until the nodes are pure is shown in the above decision tree; however, the terminal nodes do not reveal much information. The real terminal nodes are the third nodes. Looking at the table for this data reveals a small sample size with inconsistent results. Even though the traffic is the strongest predictor, there are still days in which there is no driving. The same inconsistent result is shown for wind and especially for temperature. This makes it difficult for the algorithm to design a successful tree. I did split the data to see how successful this model is, and due to the low number of samples, this setup is not accurate with 33%.

Part 2

```
In [11]: ##create a fuzzy tree by scratch
         ##using the above info gains, the traffic amount has the highest gain: this is t
         ##next select sub nodes
         short_traffic_table = table_driving[table_driving['Traffic-Jam_X3'] == "short"]
         short_traffic_table
         H_temp_cool = -1 * ((3/4) * np.log2(3/4) + (1/4) * np.log2(1/4))
         Net_Entropy_temp_2 = (4/7) * H_temp_cool
         Total_Reduction_temp_2 = 1 - Net_Entropy_temp_2
         print(Total_Reduction_temp_2)

         0.536412500309067

In [12]: H_wind_strong = -1 * ((2/3) * np.log2(2/3) + (1/3) * np.log2(1/3))
         Net_Entropy_wind_2 = (3/7) * H_wind_strong
         Total_Reduction_wind_2 = 1-Net_Entropy_wind_2
         print(Total_Reduction_wind_2)

         0.606444642548076

In [13]: long_traffic_table = table_driving[table_driving['Traffic-Jam_X3'] == "long"]
         long_traffic_table
         H_temp_hot = -1 * ((1/3) * np.log2(1/3) + (2/3) * np.log2(2/3))
         H_temp_mild = -1 * ((2/4) * np.log2(2/4) + (2/4) * np.log2(2/4))
         Net_Entropy_temp_3 = (4/7) * H_temp_mild + (3/7) * H_temp_hot
         Total_Reduction_temp_3 = 1-Net_Entropy_temp_3
         print(Total_Reduction_temp_3)

         0.035016071119504555

In [14]: h_wind_weak = -1 * ((2/4) * np.log2(2/4) + (2/4) * np.log2(2/4))
         h_wind_strong = -1 * ((1/3) * np.log2(1/3) + (2/3) * np.log2(2/3))
         Net_Entropy_wind_3 = (4/7) * h_wind_weak + (3/7) * h_wind_strong
         Total_Reduction_wind_3 = 1 - Net_Entropy_wind_3
         print(Total_Reduction_wind_3)

         0.035016071119504555
```

```
In [20]: short_traffic_table
```

Out[20]:

	Temperature_X1	Wind_X2	Traffic-Jam_X3	Car_Driving_y
4	cool	weak	short	yes
5	cool	strong	short	no
6	cool	strong	short	yes
8	cool	weak	short	yes
9	mild	weak	short	yes
10	mild	strong	short	yes
12	hot	weak	short	yes

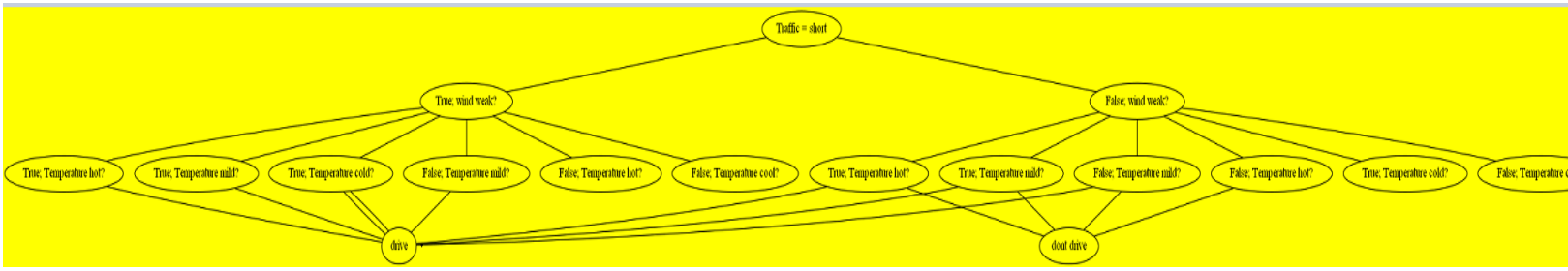
```
In [21]: long_traffic_table
```

Out[21]:

	Temperature_X1	Wind_X2	Traffic-Jam_X3	Car_Driving_y
0	hot	weak	long	no
1	hot	strong	long	no
2	hot	weak	long	yes
3	mild	weak	long	yes
7	mild	weak	long	no
11	mild	strong	long	yes
13	mild	strong	long	no

The solution to the above problem would be to build a fuzzified tree. To fuzzify this decision tree, the mathematical design in Gopal (2019) was used. The math was calculated by hand in python. It uses an adjusted information gain model that is similar to the above algorithm, but it does not make assumptions. Example 8.4 was followed to generate the new fuzzified tree. From the information in part 1, the traffic has the highest information gain and is used as the first node. Then the other variables are reevaluated for their information gain. The next is the wind then the temperature. The information gain results are printed in the image at the top.

Following the table to the left and the information gain result were then generated into a graph using the pydot package (Carrera, nd) (Thapa, 2020). The image will be attached as a separate pdf due to the size.



Article Review

In the article by Banakar et. al. (2017), the authors had a decision tree model that used a limited set of variables, so they fuzzified the decision tree. Overall, it was successful. In comparison to this dataset, there was enough data to split and test the tree models for their predictions. They used PCA and CFS to select their features. They tested three different trees included the C4.5, but the REP was the best preforming model. The REP uses reduced error pruning. This was different than what was done in the previous sections. Their fuzzy model was based on if then statements that are similar to what was done with the pydot algorithm above.

References

AskPython. (2020, December 7). *Decision trees in python - step-by-step implementation*. AskPython. Retrieved December 1, 2021, from https://www.askpython.com/python/examples/decision-trees.

Banakar, A., Zareiforoush, H., Baigvand, M., Montazeri, M., Khodaei, J., & Behroozi, K. N. (2017). Combined Application of Decision Tree and Fuzzy Logic Techniques for Intelligent Grading of Dried Figs. *Journal of Food Process Engineering*, *40*(3), n/a-N.PAG. https://doi-org.lopes.idm.oclc.org/10.1111/jfpe.12456

Carrera, E. (n.d.). *PYDOT*. PyPI. Retrieved December 1, 2021, from https://pypi.org/project/pydot/.

Gopal, M. (2019). *Applied machine learning*. McGraw-Hill Education.

graphviz. (n.d.). *User guide*. User Guide - graphviz 0.17 documentation. Retrieved September 20, 2021, from https://graphviz.readthedocs.io/en/stable/manual.html.

Larose, C. D., & Larose, D. T. (2019). *Data science using Python and R*. Wiley.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Thapa, M. (2020, October 16). *Visualisation with pydot for Beginners*. Medium. Retrieved December 1, 2021, from https://tmilan0604.medium.com/visualisation-with-pydot-for-beginners-ca99c9dc530b.

Github link: https://github.com/squinton-gcu/Data-Science/tree/main/DSC-540/Assignment5