Data Science Certificate Diploma

DS650 Data Analytics

**Homework 1 (20 points)**

**You are expected to deliver a PDF document (this document) and one Colab jupyter notebook file with your Python code.**

You will implement a Decision Tree based on the eyes.csv dataset. This dataset consists of a header row, followed by 24 rows of training data as shown below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Age** | **Vision** | **Astigmatism** | **Use of Glasses** | **Class** |
| 1 | Young | Farsightedness | Yes | Rare | A |
| 2 | Young | Myopia | Yes | Often | A |
| 3 | Young | Myopia | Yes | Often | B |
| 4 | Young | Myopia | No | Rare | A |
| 5 | Young | Farsightedness | Yes | Rare | B |
| 6 | Young | Myopia | Yes | Often | A |
| 7 | Young | Farsightedness | No | Rare | B |
| 8 | Young | Myopia | No | Rare | B |
| 9 | Young | Farsightedness | Yes | Often | A |
| 10 | Young | Myopia | No | Rare | B |
| 11 | Young | Farsightedness | Yes | Often | B |
| 12 | Young | Myopia | No | Rare | B |
| 13 | Middle-aged | Myopia | No | Rare | A |
| 14 | Middle-aged | Farsightedness | No | Rare | B |
| 15 | Middle-aged | Farsightedness | No | Often | B |
| 16 | Middle-aged | Myopia | Yes | Rare | B |
| 17 | Middle-aged | Farsightedness | Yes | Rare | B |
| 18 | Middle-aged | Farsightedness | No | Rare | B |
| 19 | Middle-aged | Myopia | Yes | Rare | B |
| 20 | Middle-aged | Myopia | Yes | Often | B |
| 21 | Middle-aged | Farsightedness | Yes | Often | B |
| 22 | Elderly | Myopia | No | Rare | A |
| 23 | Elderly | Farsightedness | Yes | Often | A |
| 24 | Elderly | Farsightedness | Yes | Rare | A |

The .csv file contains five categorical attributes: *Age, Vision, Astigmatism, UseOfGlasses, and Class. Class* categorizes patients into two groups: *Class A* indicates a potential glaucoma risk of less than 5% in the near future, while *Class B* indicates a risk greater than 5%, requiring periodic examinations for patients. So, *Class* would be the output variable (or the predicted class), while the rest attributes would be the input variables.

1. **(2 points).** Calculate by hand, the information gain after the dataset has been split according to the values of Vision and provide the result and the calculations in the following spaces, respectively.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Answer:**   |  |  |  |  | | --- | --- | --- | --- | | **Entropy (before)** | **Entropy1 (myopia)** | **Entropy2 (Farsightedness)** | **Entropy (after)** | | **?** | **?** | **?** | **?** |   **Information Gain = ?** |

1. **(5 points).** In the code that follows, replace the {insert your code here} placeholder with the appropriate Python code, so as to create a function with name “eye\_attr\_entropy\_calc” for calculating the entropy left after the split of dataframe x on attribute attr, for the generated subset with attribute value subsetParam. For example, when we call the function with the following parameters eye\_attr\_entropy\_calc(eye\_data,'Vision','Myopia'), it will return the entropy left for subset ‘Myopia’, after splitting on attribute ‘Vision’. Afterwards, fill the respective table, using this function.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Answer:**  def eye\_attr\_entropy\_calc(data, attr, subset\_param):  subset = data[data[attr] == subset\_param]  s = subset.shape[0]  s1 = \_\_\_\_\_{insert your code here}\_\_\_\_\_  s2 = \_\_\_\_\_{insert your code here}\_\_\_\_\_  y1 = s1 / s  y2 = s2 / s  if (s1 == 0 or s2 == 0):  return 0  else:  return \_\_\_\_\_{insert your code here}\_\_\_\_\_   |  |  |  | | --- | --- | --- | | **attr** | **subsetParam** | **Output (entropy)** | | Age | Young | 0.980 | | Age | Middle-aged | **?** | | Vision | Myopia | **?** | | Astigmatism | Yes | **?** | | Astigmatism | No | 0.881 | | UseOfGlasses | Often | 0.991 | | UseOfGlasses | Rare | **?** | |

1. **(5 points).** How does the root node split for the given dataset, if we use the information gain? Use the function that you implemented in question (b). Justify your answer.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Answer:**   |  |  |  |  | | --- | --- | --- | --- | | **Information Gain** | | | | | **Age** | **Vision** | **Astigmatism** | **UseOfGlasses** | | **?** | **?** | 0.013 | **?** |   **Justification**: …. |

1. **(5 points).** In the code that follows, replace the {insert your code here} placeholder with the appropriate Python code, so as to fit a decision tree using the sklearn library. Use the “entropy” as the splitting method, and set the maximum depth to be 5. Use a random seed of 2. Do not add additional parameters to the function beyond those that are requested (2 points). Afterwards, plot the tree (1 points), and programmatically check (writing the respective Python code) what is the class of the following two records (2 points):

|  |  |  |  |
| --- | --- | --- | --- |
| **Age** | **Vision** | **Astigmatism** | **UseOfGlasses** |
| Young | Myopia | No | Often |
| Elderly | Myopia | Yes | Often |

|  |
| --- |
| **Answer:**  import pandas as pd  eyes = pd.read\_csv("eyes.csv")  X = eyes.drop("Class", axis = 1)  y = eyes["Class"]  X\_dum = pd.get\_dummies(X, drop\_first = True)  tree = \_\_\_\_\_{insert your code here}\_\_\_\_\_  Decision Tree diagram |

**e**. **(3 points).** Perform 10-fold cross validation to find the optimal number for the hyperparameter max\_depth. Implement using GridSearchCV function, a grid search for the values [2, 3, 4, 5, 6, 7]. (2 points) Report the best value and the 10-fold cross validation accuracy for this value. (1 point)

|  |
| --- |
| **Answer:** |