**Assignment 3**

Show **all** your work for each question

This assignment is to be completed individually.  
Due: Nov. 12

Total: 45 points

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. Your model predicted the following labels on a given dataset:  1,1,2,2,2,3,3,4,4,4   The actual gold labels are:   1,1,1,2,2,2,3,3,4,4  Using this information calculate the following evaluation metrics of your mode’s performance. **Show** all your work including the confusion matrices.   |  |  | | --- | --- | | **Metric** | **Value** | | Mirco precision | 0.7 | | Micro Recall | 0.7 | | Micro F1-score | 0.7 | | Macro precision | 0.79 | | Macro Recall | 0.71 | | Macro F1-score | 0.74 |   Confusion matrix:   |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | Predicted 1 | Predicted 2 | Predicted 3 | Predicted 4 | | Actual 1 | 2 | 1 | 0 | 0 | | Actual 2 | 0 | 2 | 1 | 0 | | Actual 3 | 0 | 0 | 1 | 1 | | Actual 4 | 0 | 0 | 0 | 2 |   Total number of true positives = 7  Total predictions = 10  Micro precision = total true positives / total predictions = 0.7  Micro recall = 7/10 = 0.7  Micro F1-score = (2 x micro precision x micro recall)/(micro precision + micro recall)  = (2 x 0.7 x 0.7) / (0.7 + 0.7) = 0.7  For macro averaging:  Class 1:  TP = 2, FP = 0, FN = 1  Precision = 1.0  Recall = 0.67  F1 = 0.8  Class 2:  TP = 2, FP = 1, FN = 1  Precision = 0.67  Recall = 0.67  F1 = 0.67  Class 3:  TP = 1, FP = 1, FN = 1  Precision = 0.5  Recall = 0.5  F1 = 0.5  Class 4:  TP = 2, FP = 0, FN = 0  Precision = 1.0  Recall = 1.0  F1 = 1.0  Macro precision = (1.0 + 0.67 + 0.5 + 1.0) / 4 = 0.79  Macro recall = (0.67 + 0.67 + 0.5 + 1.0) / 4 = 0.71  Macro f1 score = (0.8 + 0.67 + 0.5 + 1.0) / 4 = 0.74 | 3 points |
| Answer the following questions regarding the given decision tree and the training set that it was trained on. A diagram of a diagram  Description automatically generated  **TRAINING**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Instance** | **A** | **B** | **C** | **Class** | | **1** | **0** | **0** | **0** | **+** | | **2** | **0** | **0** | **1** | **-** | | **3** | **0** | **1** | **0** | **-** | | **4** | **0** | **1** | **1** | **-** | | **5** | **1** | **0** | **0** | **+** | | **6** | **1** | **0** | **0** | **+** | | **7** | **1** | **1** | **0** | **+** | | **8** | **1** | **0** | **1** | **+** | | **9** | **1** | **1** | **0** | **+** | | **10** | **1** | **1** | **0** | **-** |   A. (2 points) Given the above decision tree that was trained on some dataset, calculate the pessimistic error with a penalty term Ω of 0.5 using the decision tree pessimistic error equation from class.   The decision tree splits the data based on attribute values A, B and C and makes predictions at the leaf nodes.  The question gave 10 input training examples and their actual classes.  Calculating errors:  --For node B(0):  Considers cases with A=0 and B=0 and predicts +  From the training set, 2 instances will be a part of node B(0):  Instance 1: correct prediction as it is in class +  Instance 2: incorrect prediction as it is in class -  Total number of misclassifications:1  --for node B(1):  Considers cases with A=0 and B=1 and predicts -  From the training set, 2 instances will be a part of node B(1):  Instance 3: correct prediction as it is in class -  Instance 4: correct prediction as it is in class -  Total number of misclassifications:0  --for node C(0):  Considers cases with A=1 and C=0 and predicts +  From the training set, 5 instances will be a part of node C(0):  Instance 5: correct prediction as it is in class +  Instance 6: correct prediction as it is in +  Instance 7: correct prediction as it is in class +  Instance 9: correct prediction as it is in class +  Instance 10: incorrect prediction as it is in class -  Total number of misclassifications:1  --for node C(1):  Considers cases with A=1 and C=1 and predicts -  From the training set, 1 instance will be a part of node C(1):  Instance 8: incorrect prediction as it is in class +  Total number of misclassifications:1  Using this info, we get total training error:    Using formula and training error to calculate pessimistic error:  B. (1 point) Using the same decision tree, calculate the optimistic error.  The optimistic error is the training error which we calculated earlier without using the penalty term. The optimistic error is err(T) which is 0.3 | 1. points |
| 1. Consider a binary classification problem with the following set of attributes   and attribute values:  • Air Conditioner = {Working, Broken}  • Engine = {Good, Bad}  • Mileage = {High, Medium, Low}  • Rust = {Yes, No}  Suppose a rule-based classifier produces the following rule set:  R1: Mileage = High *→* Value = Low  R2: Mileage = Low *→* Value = High  R3: Air Conditioner = Working, Engine = Good *→* Value = High  R4: Air Conditioner = Working, Engine = Bad *→* Value = Low  R5: Air Conditioner = Broken *→* Value = Low  **(a)** Are the rules mutually exclusive? If so, give a record that would be covered by multiple rules and state which rules are covering it. Remember that not all attributes of a record need to be included in a rule for that rule to cover that record. However, each conjunct in a rule needs to match the values of the record.  Rules are mutually exclusive if no record satisfies more than one rule at the same time. So we need to check if any record matches multiple rules. If we have a record where:  Mileage is high  AC is working  Engine is good  This will match 2 rules, R1 and R3, so thus, the rules are not mutually exclusive.  **(b)** Is the rule set exhaustive? If not, give a record that is not covered by the rule set.  Rule set is exhaustive if it covers all possible records. If we have a record where:  AC is working  Engine is bad  Mileage is medium  Rust is yes  This does not get covered by any rule, which means rule set is not exhaustive.  **(c)** Is ordering **needed** for this set of rules?  Ordering is needed when 2 or more rules cover the same record and we need to determine which rule to apply first. When rule sets are not mutually exclusive, ordering is needed to specify which rule takes precedence and as we have already determined that the rules in the question are not mutually exclusive as it fits multiple records, we need to determine which rule takes precedence so ordering is needed.  **(d)** Do you **need** a default class for the rule set? In other words, is there a time with this dataset that you **need** to rely on a default class?  Default classes are used to handle records where no rules apply to it. This is needed in cases where the rules are not exhaustive, i.e., do not cover all possible records. We already noted earlier that this question has a non exhaustive rule set so default class is needed in this case as it will be used in cases for records which do not fit into any rules. | 4 points |
| 1. Consider a training set that contains 100 positive examples and 400 negative   examples. For each of the following candidate rules,  *R*1: *A −→* + (covers 4 positive and 1 negative examples),  *R*2: *B −→* + (covers 30 positive and 10 negative examples),  *R*3: *C −→* + (covers 100 positive and 90 negative examples),   1. (2 points) Determine which is the best and worst candidate rule according to accuracy.   Accuracy = number of correctly classified examples / total number of examples covered by rule  For rule R1, total examples covered = 5 (4 positive, one negative), correctly classified = 4  Accuracy = 4/5 = 0.8  For rule R2, total examples covered = 40 (30 positive, 10 negative), correctly classified = 30  Accuracy = 30/40 = 0.75  For rule R3, total examples covered is 190 (100 positive 90 negative), correctly classified = 100  Accuracy = 100/190 = 0.526  Best rule according to accuracy is R1 as it has 80% accuracy while the worst is R3 which has a 52.6% accuracy.   1. (3 points) Determine which is the best and worst candidate rule according to FOIL’s Information Gain. Assume the initial rule is *∅ −→* + (always predict + class).     Best rule according to foil’s information gain: R3 (139.5)  Worst rule according to foil’s information gain: R1 (8) | 1. points |
| 5. (3 points) A. Given the following dataset, use majority vote KNN with Euclidean distance to predict the label of the test object: [1,1,1]. Assume K=3. **Show** all your work.   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Sample ID** | ***a*1** | ***a2*** | ***a3*** | **Label** | | *X1* | 2 | 3 | 3 | + | | *X2* | 2 | 1 | 1 | - | | *X3* | 3 | 3 | 3 | + | | *X4* | 3 | 2 | 4 | + | | *X5* | 4 | 4 | 4 | - | | *X6* | 5 | 6 | 4 | - |   Distance from test object to each sample:    The predicted label for this test object with be on the basis of the majority vote for the 2 nearest neighbours which we found above, x2 has label – while x1 and x3 have label +, thus, the label assigned to the test object is +  B. (2 points) Given the same dataset, use weighted KNN with Euclidean distance to predict the label of the test object: [1,1,1]. The objects are weighted with the equation , where d is the Euclidean distance between the test object and a training object.  If we are using weighted KNN, we calculate weights for the K nearest neighbours. We are given K is 3 in the first part of the question and we calculated the 3 nearest in the previous section, we need to find the weighted KNN for each of the 3:    As the weight of the – class is more than the + class, using weighted KNN, we assign class – to the test object. | 1. points |
| 6. Suppose the fraction of undergraduate students who play basketball is 15% and the fraction of graduate students who play basketball is 23%. If one-fifth of the college students are graduate students and the rest are undergraduates, what is the probability that a student who plays basketball is a graduate student?  P(G) be the probability that a student is a graduate student  P(U) is the probability that a student is an undergrad student  P(B|G) is the probability that a graduate student plays basketball  P(B|U) is the probability that an undergrad student plays basketball  And P(B) is the probability that a student plays basketball  In the question we are given:  P(B|G) = 0.23  P(B|U) = 0.15  P(G) = 0.2  P(U) = 1-P(G) = 0.8  using conditional probability for finding P(B):  P(B) = P(B|G) x P(G) + P(B|U) x P(U)  P(B) = (0.23 x 0.2) + (0.15 x 0.8)  P(b) = 0.166  Using this information with bayes theorem:  P(G|B) = (P(B|G) x P(G) ) / P(B)  = (0.23 x 0.2) / 0.166  = 0.277 the probability that a student who plays basketball is a grad student is 0.277 or 27.7% | 3 points |
| 7. Given the following dataset answer the questions A) and B:  A table with numbers and symbols  Description automatically generated   1. (2 points) Estimate the conditional probabilities for *P*(*A* = 1*|*+), *P*(*B* = 1*|*+), *P*(*C* = 1*|*+), *P*(*A* = 1*|−*), *P*(*B* = 1*|−*), and *P*(*C* = 1*|−*)  Total number of positive instances = 5   Total number of negative instances = 5  Conditional probabilities for positive class:  P(A= 1 | +) :  A = 1 occurs in 3/5 positive instances (instance 2, 5 and 10) Thus, probability = 3/5 which is 0.6  P(B= 1 | +) :  B = 1 occurs in 2/5 positive instances (instance 9 and 10) Thus, probability = 2/5 which is 0.4  P(C= 1 | +) :  C = 1 occurs in 4/5 positive instances (instance 2, 5, 6 and 10) Thus, probability = 4/5 which is 0.8  Conditional probabilities for negative class:  P(A= 1 | -) :  A = 1 occurs in 2/5 negative instances (instance 4 and 7) Thus, probability = 2/5 which is 0.4  P(B= 1 | -) :  B = 1 occurs in 2/5 positive instances (instance 3 and 7) Thus, probability = 2/5 which is 0.4  P(C= 1 | -) :  C = 1 occurs in 2/5 positive instances (instance 1 and 8) Thus, probability = 2/5 which is 0.4  So we have:  P(A= 1 | +) : 0.6  P(B= 1 | +) : 0.4  P(C= 1 | +) : 0.8  P(A= 1 | -) : 0.4  P(B= 1 | -) : 0.4  P(C= 1 | -) : 0.4   1. (3 points) Use the conditional probabilities in part (a) to predict the class label for a test sample (*A* = 1*, B* = 1*, C* = 1) using the naive Bayes approach. | 5 points |
|  |  |

## 

## Code (18 points)

Complete and submit the python script **a3.py** that reads in the file **cereal.csv**. The dataset is a modified version of the dataset from https://www.kaggle.com/datasets/semakulapaul/cereals-dataset?select=cereal.csv. Your code will create a KNN classifier and a naïve Bayes classifier along with functions to evaluate the performance of some model’s output. Your classifiers are going to predict the shelf number that a cereal will be displayed on, which is the rightmost column in cereal.csv. Your code must run from command line with the call:  
 **python a3.py cereal.csv**  
The classes, methods, and functions descriptions are given in the starter code and below:

|  |  |
| --- | --- |
| **Read\_data**(file\_path) | Reads in the file given from command line and returns the dataset as a list of tuples where each tuple is an instance with its class |

**Class KNN**

|  |  |
| --- | --- |
| **preprocess\_single\_sample(**self, curr) | Preprocesses a single given training sample **curr** by formatting it with the following:   * Convert values in the mfr attribute to a one-hot encoding, which is a binary sequence with a 1 in the position corresponding to the mfr ID. More detail in the starter code. * Remove the name attribute   Returns the formatted instance |
| **preprocess\_training\_set** (self) | update self.train\_samples with the new modified samples by calling preprocess\_single\_sample on each sample in training |
| **euclidean**(self, sample\_0, sample\_1) | Returns the euclidean distance of two vectors |
| **get\_k\_nearest**(self,k, test\_sample) | Gets the k nearest instances to the test samples based on the Euclidean distance. Returns a list of k tuples. Each tuple contains the values of an instance, its label, and the Euclidean distance from test\_sample |
| **predict\_majority\_vote(**self,k, test\_sample) | Predicts the class of test\_sample based on majority vote KNN. Returns a tuple with the predicted class and the number of votes for that class. |
| **predict\_weighted\_vote**(self, k, test\_sample) | Predicts the class of test\_sample based on majority vote KNN. Returns a tuple with the predicted class and the sum of weights for that class. Uses ,where d is the Euclidean distance. |
| **train**(self, dataset) | Stores the dataset is the variable self.train\_samples |
| **predict**(self, test\_instance=[], k=1, version=None) | Uses predict\_majority\_vote or predict\_weighted\_vote to perform KNN. |

**Class NB**

|  |  |
| --- | --- |
| **preprocess\_dataset**(self, dataset) | Preprocesses the entire training dataset by formatting each instance with the following:   * Remove the name attribute * Discretize all numerical attributes into 4 categories/bins using equal width * Name the bins values [0,1,2,3] * Allow the last bin to account for all values equal to or larger than the max value and first bin to account for all values smaller than the minimum value * mfr attribute remains unchanged from the original dataset |
| **calculate\_prior**(self, class\_input) | Calculates the prior of the given class  **Class\_input** based on the training set. |
| **calculate\_cond\_prob**(self,X\_attr,X\_value,Y) | Calculates the conditional probability of attribute X\_attr having value X\_value given class Y based on the training set.  Returns the conditional probability. |
| **calculate\_cond\_prob\_m\_estimate**(self,X\_attr,X\_value,Y) | Calculates the conditional probability of attribute X\_attr having value X\_value given class Y using m-estimate based on the training set.  Let m =3 and assume a uniform distribution of values for an attribute for p.  Returns the conditional probability using m-estimate. |
| **train**(self) | Calculates priors and store in self.priors |
| **preprocess\_test\_instance**(self, test) | Preprocesses the values in the test instance according to the same steps performed to preprocess the training set in **preprocess\_dataset**  Returns the formatted instance |
| **predict**(self, test\_instance, method) | Predicts the class of **test\_instance** using naïve Bayes with the given **method** to estimate the conditional probabilities (basic or m-estimate).  Returns a tuple that includes the label predicted and the probability for that label. Since you are performing a comparison to predict the labels, ignore the probability in the denominator. |

|  |  |
| --- | --- |
| **micro\_prec\_recall\_fscore**(golds, preds) | Calculates the micro precision, recall, and f1-score of a model’s output **preds** against the gold standard (ground truth) **golds.** Returns a tuple (micro-precision, micro-recall, micro-f1Score) |
| **macro\_prec\_recall\_fscore**(golds, preds) | Calculates the macro precision, recall, and f1-score of a model’s output **preds** against the gold standard (ground truth) **golds.** Returns a tuple (micro-precision, micro-recall, micro-f1Score) |

# IMPORTANT for your code!

* To run the code, you will need to run from command line:
  + python a3.py cereal.csv
* I will run the code in the same manner and therefore, it must work this way.
* The code submission must be a .py file and you don’t need to submit the data csv file.
* You may **NOT** import any additional libraries or packages that are not already imported with the starter code.
* The only function from the math library that you can use is the floor, function. You may also use the math.inf attribute.
* If you import additional libraries, your code will automatically be marked out of 50% and all code that uses the unapproved library will be marked as incorrect.
* Any attempt to modify the declaration of functions, such as the parameters that it uses, will be marked as incorrect.
* Any modification to the test code (all code below the line ###TESTS) will result in a **0** on the entire coding assignment.
* Code that can’t be run due to a syntax error will be marked out of 50%. If you are unable to get a function working, have it return a -1 for all expected int values and “none” for all expected string values. This should make your code not give a syntax error and not be marked out of 50% because of it.
* Your submitted code must **NOT** contain any additional print statements than what was given in the starter code.
* Have fun with the assignment and the data!