COMP307/AIML420 — Fundamentals of AI

Assignment 1: Basic Machine Learning Algorithms

Part 1: k-Nearest Neighbour Method

(a) Output results for k=1

```
Class predictions of test wines using k-Nearest Neighbour,
where k = 1:

Wine 1: Prediction = 3, Actual = 3    Success!
Wine 2: Prediction = 3, Actual = 3    Success!
Wine 3: Prediction = 3, Actual = 3    Success!
Wine 4: Prediction = 1, Actual = 1    Success!
Wine 5: Prediction = 1, Actual = 1    Success!
Wine 6: Prediction = 1, Actual = 1    Success!
Wine 7: Prediction = 1, Actual = 2    Fail...
Wine 8: Prediction = 2, Actual = 2    Success!
Wine 9: Prediction = 1, Actual = 1    Success!
Wine 10: Prediction = 2, Actual = 2    Success!
Wine 11: Prediction = 2, Actual = 2    Success!
Wine 12: Prediction = 3, Actual = 2    Fail...
Wine 13: Prediction = 3, Actual = 3    Success!
Wine 14: Prediction = 3, Actual = 3    Success!
Wine 15: Prediction = 1, Actual = 1    Success!
Wine 16: Prediction = 2, Actual = 2    Success!
Wine 17: Prediction = 3, Actual = 3    Success!
Wine 18: Prediction = 3, Actual = 3    Success!
Wine 19: Prediction = 1, Actual = 1    Success!
Wine 20: Prediction = 1, Actual = 1    Success!
Wine 21: Prediction = 3, Actual = 3    Success!
Wine 22: Prediction = 2, Actual = 2    Success!
Wine 23: Prediction = 2, Actual = 2    Success!
Wine 24: Prediction = 3, Actual = 3    Success!
Wine 25: Prediction = 2, Actual = 2    Success!
Wine 26: Prediction = 3, Actual = 2    Fail...
Wine 27: Prediction = 2, Actual = 2    Success!
Wine 28: Prediction = 3, Actual = 3    Success!
Wine 29: Prediction = 2, Actual = 2    Success!
Wine 30: Prediction = 1, Actual = 1    Success!
Wine 31: Prediction = 2, Actual = 2    Success!
Wine 32: Prediction = 1, Actual = 1    Success!
Wine 33: Prediction = 2, Actual = 2    Success!
Wine 34: Prediction = 1, Actual = 1    Success!
Wine 35: Prediction = 2, Actual = 2    Success!
Wine 36: Prediction = 2, Actual = 2    Success!
Wine 37: Prediction = 2, Actual = 2    Success!
Wine 38: Prediction = 2, Actual = 2    Success!
Wine 39: Prediction = 2, Actual = 2    Success!
Wine 40: Prediction = 1, Actual = 1    Success!
Wine 41: Prediction = 2, Actual = 2    Success!
Wine 42: Prediction = 2, Actual = 2    Success!
Wine 43: Prediction = 3, Actual = 3    Success!
Wine 44: Prediction = 1, Actual = 1    Success!
Wine 45: Prediction = 2, Actual = 2    Success!
Wine 46: Prediction = 1, Actual = 1    Success!
Wine 47: Prediction = 3, Actual = 3    Success!
Wine 48: Prediction = 2, Actual = 2    Success!
Wine 49: Prediction = 2, Actual = 2    Success!
Wine 50: Prediction = 1, Actual = 1    Success!
Wine 51: Prediction = 3, Actual = 3    Success!
Wine 52: Prediction = 1, Actual = 1    Success!
Wine 53: Prediction = 1, Actual = 1    Success!
Wine 54: Prediction = 3, Actual = 3    Success!
Wine 55: Prediction = 3, Actual = 3    Success!
Wine 56: Prediction = 1, Actual = 1    Success!
Wine 57: Prediction = 1, Actual = 1    Success!
Wine 58: Prediction = 3, Actual = 3    Success!
Wine 59: Prediction = 1, Actual = 1    Success!
Wine 60: Prediction = 3, Actual = 3    Success!
Wine 61: Prediction = 3, Actual = 3    Success!
Wine 62: Prediction = 1, Actual = 2    Fail...
Wine 63: Prediction = 2, Actual = 2    Success!
Wine 64: Prediction = 3, Actual = 3    Success!
Wine 65: Prediction = 2, Actual = 2    Success!
Wine 66: Prediction = 3, Actual = 3    Success!
Wine 67: Prediction = 3, Actual = 3    Success!
Wine 68: Prediction = 1, Actual = 1    Success!
Wine 69: Prediction = 1, Actual = 1    Success!
Wine 70: Prediction = 2, Actual = 2    Success!
Wine 71: Prediction = 1, Actual = 2    Fail...
Wine 72: Prediction = 3, Actual = 3    Success!
Wine 73: Prediction = 2, Actual = 2    Success!
Wine 74: Prediction = 2, Actual = 2    Success!
Wine 75: Prediction = 1, Actual = 1    Success!
Wine 76: Prediction = 1, Actual = 1    Success!
Wine 77: Prediction = 1, Actual = 1    Success!
Wine 78: Prediction = 3, Actual = 3    Success!
Wine 79: Prediction = 1, Actual = 1    Success!
Wine 80: Prediction = 1, Actual = 1    Success!
Wine 81: Prediction = 2, Actual = 2    Success!
Wine 82: Prediction = 2, Actual = 2    Success!
Wine 83: Prediction = 3, Actual = 3    Success!
Wine 84: Prediction = 1, Actual = 1    Success!
Wine 85: Prediction = 2, Actual = 2    Success!
Wine 86: Prediction = 1, Actual = 1    Success!
Wine 87: Prediction = 1, Actual = 1    Success!
Wine 88: Prediction = 2, Actual = 2    Success!
Wine 89: Prediction = 1, Actual = 1    Success!

Successfully predicted 84 out of 89 wines

Accuracy: 94.38203%
```

(b) kNN where k=3

The classification accuracy of the test set for the k-Nearest Neighbour method where k=3 was 95.505615%. This was ever so slightly higher to the accuracy of around 94.4% for k=1 with one more successful prediction. It's hard to say if the change in k is significant, especially after 1 test. I'd feel more confident if I applied the k-fold cross validation, however. Another potential concern for k=3 is that there's 3 possible classifications for each wine. There is a potential for a tie. Otherwise, yes, there is a small improvement on the predictive performance for the k=3 classifier compared to the k=1 classifier. Probably due to it reducing overfitting to the training data.

(c) Advantages and disadvantages to kNN

kNN is very easy to understand compared to other ML languages. No real model is needed to be generated with the training data, we just need to store it. It's very good at handling noise as it can take into account several data points before determining the classification. It's pretty flexible with most distributions and can handle several different classes too. For disadvantages, predicting is costly. It can take a lot of memory as we need to store the training data for each prediction. The need to check every training instance for each prediction, as well as every feature for each instance can be computationally costly. The high number of dimensions also obscures the concept of "distance" and may not guarantee results. Increasing the dimensions also increases the costs exponentially.

Finding a good k value is important too. The lower the k value, the more vulnerable it is to overfitting. The higher the k value, we decrease the likelihood the algorithm will find a useful pattern. If k was the same as the size of the dataset, it would only pick the most common class every time. The prediction will never change. Picking a k value that avoids tie breakers is important too. Interestingly for the provided training and data set, the accuracy peaked when k = 11.

(d) k-fold cross validation

To apply the k-fold cross validation method for this dataset, we need to combine the training and test data and shuffle the instance order. For a 5-fold CV we split the data into 5 equal sets. 4 of the sets will be used as training data, and the last remaining set to be used as test data. Then apply kNN and measure the result. Then we assign a different test set and try again. Let’s say, Set 1 is the test set and Sets 2-5 are the training data. For round 2, Set 2 will be the test set with the remaining sets (set 1, 3, 4, and 5) being the training data. Then it’s set 3 and so on until all of the data has had the opportunity to be the test data once. If the evaluation metric was the accuracy, we can take the average result from those 5 folds.

Part 2: Decision Tree Learning Method

During this section, I'm using the terms "category" and "class" interchangably.

(a) Results of the DTL program

Applying the Decision Tree Learning method to the provided *hepatitis-training* and *hepatitis-test* resulted with a clarification accuracy of 76%. Here's the decision tree classifier printed by the program:

```
ASCITES = True:
    SPIDERS = True:
        VARICES = True:
            STEROID = True:
                Class live, prob=1.00
            STEROID = False:
                SPLEENPALPABLE = True:
                    FIRMLIVER = True:
                        Class live, prob=1.00
                    FIRMLIVER = False:
                        BIGLIVER = True:
                            SGOT = True:
                                Class live, prob=1.00
                            SGOT = False:
                                FEMALE = True:
                                    Class live, prob=1.00
                                FEMALE = False:
                                    ANOREXIA = True:
                                        Class die, prob=1.00
                                    ANOREXIA = False:
                                        Class live, prob=1.00
                        BIGLIVER = False:
                            Class live, prob=1.00
                SPLEENPALPABLE = False:
                    ANOREXIA = True:
                        Class live, prob=1.00
                    ANOREXIA = False:
                        Class die, prob=1.00
        VARICES = False:
            Class die, prob=1.00
    SPIDERS = False:
        FIRMLIVER = True:
            ANOREXIA = True:
                SGOT = True:
                    Class live, prob=1.00
                SGOT = False:
                    Class die, prob=1.00
            ANOREXIA = False:
                Class live, prob=1.00
        FIRMLIVER = False:
            SGOT = True:
                BIGLIVER = True:
                    Class live, prob=1.00
                BIGLIVER = False:
                    Class die, prob=1.00
            SGOT = False:
                Class live, prob=1.00
ASCITES = False:
    BIGLIVER = True:
        VARICES = True:
            FIRMLIVER = True:
                STEROID = True:
                    Class die, prob=1.00
                STEROID = False:
                    BILIRUBIN = True:
                        Class live, prob=1.00
                    BILIRUBIN = False:
                        Class die, prob=1.00
            FIRMLIVER = False:
                Class live, prob=1.00
        VARICES = False:
            Class die, prob=1.00
    BIGLIVER = False:
        Class live, prob=1.00
```

Using the baseline classifier (which always predicts the most frequent class in the training set) we get an accuracy of 80%. Compared to the accuracy of 76% using the decision tree, the DTL method yields worse results than simply assuming the class will be “live” every time. It’s possible the test data is biassed having 80% of cases is “live”, but looking at the training data the “live” categories is around 80% too. So that is not the case, at least within the scope of both data sets. Therefore, the worse result is either due to the decision tree overfitting to the training data; or there isn’t enough data to reach a useful conclusion. I suspect it’s due to overfitting because every classification (every leaf node) gives the probability of 100%

(b) Pruning

i. Pruning criteria

After some experimenting, I would use the impurity as criteria for pruning. Instead of only creating a leaf node when the impurity is zero, I would make a small threshold for a little impurity. In other words, if the impurity for the given instances was below a particular value, then build a leaf. The measure of impurity essentially determines how high of a percentage the dominant class takes up in a set. There’s a danger of overfitting when expecting a set to be pure before constructing a leaf. So a little bit of impurity, within reason, allows the tree to generalise more. Which is what we want.

ii. Why pruning reduces accuracy on the training set?

Another way to visualise pruning is that we're combining 2 or more leaves to create one big leaf. Some of those leaves may contain different categories. Not all instances associated with that leaf will fit the chosen category. Therefore, if we used the training data on the decision tree, the accuracy will no longer be 100%

iii. When pruning, should we expect the accuracy on the test set to decrease as well?

In theory, no. When should expect a higher accuracy now that the decision tree model is more generalised it should hopefully be more accurate with the test data.

Part 3: Perceptron

(a) Accuracy

Unfortunately, my perceptron didn’t converge. The accuracy increased with each interaction initially, but after most iterations it hovered around 90% accuracy. So therefore it didn’t find the correct set of weights. The performance didn’t change much between runs. This is probably due to all the weights being initially set to zero, instead of being random. However, I did check if randomising weights made a difference. While there was some variation, on average it still hovered around 90% accuracy.

(b) Performance

Evaluating the perceptron’s performance on the training data alone increases the likelyhood over overfitting. Neither would it confirm if future tests guarantee the same success rate (90% in this case). There’s no way to tell how actually effective it is because it’s not making any predictions before we compare the results.

To avoid confusion, I’ve included separate files in a separate folder. This contains the source code and executable for a perceptron program that splits in half the data into training and test sets. It develops the weights from the training set, and tests the final weights by making predictions using the test set. That way it’s a fairer evaluation. In this case around 83% of the test instances were classified correctly. One could say the performance is worse, but this test is probably a more accurate reflection if used in the real world. Obviously, implementing k-fold cross validation would provide a more useful evaluation metric.