Olivia Carline 300334926

Comp 307 Assignment One Report

Part One (K Nearest Neighbours)

K Nearest Neighbour algorithm is used for both classification and regression, in this assignment it is used for classification. The algorithm is very simple to implement while also being very versatile. KNN is referred to as a non parametric lazy learning algorithm which essentially means it does not make any assumptions on the underlying data distribution and will not use the training points to do any generalization.

1. Report the class labels of each instance in the test set predicted by the basic nearest neighbour method (k=1), and the classification accuracy on the test set of the basic nearest neighbour method;

Classname	KNN Predicted (K = 1)	Classname	KNN Predicted (k = 1)
Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa
Iris-virginica	Iris-virginica	Iris-setosa	Iris-setosa
Iris-setosa	Iris-setosa	Iris-versicolor	Iris-versicolor
Iris-setosa	Iris-setosa	Iris-versicolor	Iris-versicolor
Iris-versicolor	Iris-versicolor	Iris-setosa	Iris-setosa
Iris-virginica	Iris-virginica	Iris-virginica	Iris-virginica
Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa
Iris-versicolor	Iris-versicolor	Iris-virginica	Iris-virginica
Iris-versicolor	Iris-versicolor	Iris-virginica	Iris-virginica
Iris-versicolor	Iris-versicolor	Iris-versicolor	Iris-versicolor
Iris-virginica	Iris-virginica	Iris-setosa	Iris-setosa
Iris-versicolor	Iris-versicolor	Iris-versicolor	Iris-versicolor
Iris-virginica	Iris-virginica	Iris-setosa	Iris-setosa
Iris-versicolor	Iris-versicolor	Iris-setosa	Iris-setosa
Iris-versicolor	Iris-versicolor	Iris-versicolor	Iris-versicolor
Iris-setosa	Iris-setosa	Iris-virginica	Iris-virginica
Iris-virginica	Iris-virginica	Iris-virginica	Iris-virginica
Iris-virginica	Iris-virginica	Iris-virginica	Iris-virginica
Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa
Iris-versicolor	Iris-versicolor	Iris-virginica	Iris-virginica
Iris-virginica	Iris-virginica	Iris-virginica	Iris-virginica
Iris-versicolor	Iris-versicolor	Iris-virginica	Iris-virginica
Iris-virginica	Iris-virginica	Iris-virginica	Iris-virginica
Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa
Iris-virginica	Iris-virginica	Iris-setosa	Iris-setosa
Iris-setosa	Iris-setosa	Iris-versicolor	Iris-versicolor
Iris-setosa	Iris-setosa	Iris-virginica	Iris-virginica
Iris-setosa	Iris-setosa	Iris-virginica	Iris-virginica
Iris-virginica	Iris-virginica	Iris-versicolor	Iris-versicolor
Iris-versicolor	Iris-versicolor	Iris-setosa	Iris-setosa
Iris-versicolor	Iris-versicolor	Iris-versicolor	Iris-versicolor
Iris-virginica	Iris-virginica	Iris-versicolor	Iris-versicolor
Iris-versicolor	Iris-versicolor	Iris-versicolor	Iris-versicolor
Iris-virginica	Iris-virginica	Iris-setosa	Iris-setosa
Iris-setosa	Iris-setosa	Iris-versicolor	Iris-versicolor
Iris-versicolor	Iris-versicolor	Iris-versicolor	Iris-versicolor
Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa
Iris-virginica	Iris-virginica		

Showing 75 instances, using K =1, all 75 instances are classified accurately
Algorithm Accuracy: 100.00%

2. Report the classification accuracy on the test set of the k-nearest neighbour method where k=3, and compare and comment the performance of the two classifiers (k=1 and k=3);

Classname	KNN Predicted (K = 3)	Classname	KNN Predicted (k = 3)
Iris-setosa	Iris-setosa	Iris-virginica	Iris-virginica
Iris-virginica	Iris-virginica	Iris-setosa	Iris-setosa
Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa
Iris-setosa	Iris-setosa	Iris-versicolor	Iris-versicolor
Iris-versicolor	Iris-versicolor	Iris-versicolor	Iris-versicolor
Iris-virginica	Iris-virginica	Iris-setosa	Iris-setosa
Iris-setosa	Iris-setosa	Iris-virginica	Iris-versicolor
Iris-versicolor	Iris-versicolor	Iris-setosa	Iris-setosa
Iris-versicolor	Iris-versicolor	Iris-virginica	Iris-virginica
Iris-versicolor	Iris-versicolor	Iris-virginica	Iris-virginica
Iris-virginica	Iris-virginica	Iris-versicolor	Iris-versicolor
Iris-versicolor	Iris-versicolor	Iris-setosa	Iris-setosa
Iris-virginica	Iris-virginica	Iris-versicolor	Iris-versicolor
Iris-versicolor	Iris-versicolor	Iris-setosa	Iris-setosa
Iris-versicolor	Iris-versicolor	Iris-setosa	Iris-setosa
Iris-setosa	Iris-setosa	Iris-versicolor	Iris-versicolor
Iris-virginica	Iris-virginica	Iris-virginica	Iris-virginica
Iris-setosa	Iris-setosa	Iris-virginica	Iris-virginica
Iris-virginica	Iris-versicolor	Iris-virginica	Iris-virginica
Iris-versicolor	Iris-virginica	Iris-setosa	Iris-setosa
Iris-virginica	Iris-versicolor	Iris-virginica	Iris-virginica
Iris-versicolor	Iris-versicolor	Iris-virginica	Iris-virginica
Iris-virginica	Iris-virginica	Iris-virginica	Iris-virginica
Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa
Iris-virginica	Iris-virginica	Iris-setosa	Iris-setosa
Iris-setosa	Iris-setosa	Iris-versicolor	Iris-versicolor
Iris-setosa	Iris-setosa	Iris-virginica	Iris-virginica
Iris-setosa	Iris-virginica	Iris-virginica	Iris-virginica
Iris-versicolor	Iris-versicolor	Iris-versicolor	Iris-versicolor
Iris-versicolor	Iris-versicolor	Iris-setosa	Iris-setosa
Iris-virginica	Iris-virginica	Iris-versicolor	Iris-versicolor
Iris-versicolor	Iris-virginica	Iris-versicolor	Iris-versicolor
Iris-virginica	Iris-virginica	Iris-versicolor	Iris-versicolor
Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa
Iris-versicolor	Iris-versicolor	Iris-versicolor	Iris-versicolor
Iris-setosa	Iris-setosa	Iris-versicolor	Iris-versicolor
		Iris-setosa	Iris-setosa

Showing 75 instances, using K =3, 73 instances are classified accurately (wrong classifications are highlighted)
Algorithm Accuracy: 96.00%

3. Discuss the main advantages and disadvantages of this classification method.

Advantages	Disadvantages
Simple algorithm to implement, easy to use Cost of learning process is 0 Effective with large training data sets	Algorithm's lazy learning can lead to the algorithm not generalizing well and not being robust for noisy data KNN can become computationally expensive to implement when the dataset is very large

4. Assuming that you are asked to apply the k-fold cross validation method for the above problem and k=5, what would you do? State the major steps.

    a. The main idea for K-fold Cross-Validation is to chop the instances into K equal chunks, in this case splitting 75 instances into 5 equal sets of 15.
    b. For every chunk in turn, treat it as the test data set and use the remaining K-1 chunks as the training data set, use the the training set to train the classifier and apply the test set.
    c. You then need to repeat the process k (5) times, with each of the chunks used exactly once as the test data set.
    d. Finally, average/combine the results to produce a single estimate

5. In the above problem, assuming that the class labels are not available in the training set and the test set, and that there are three cluters, which method would you use to group the examples in the data set? State the major steps.

K means clustering is used when data is unlabelled, it is a method of cluster analysis which partitions m instances into k clusters where each instance belongs to the cluster with the nearest mean. It requires some sort of distance measure (typically Euclidean distance) and assumes the number of clusters.

a. You first need to set k initial means, randomly from the data (in this instance k = 3)
b. Set k initial “means” (in this case k=3) randomly from the data set (shown in color).
c. Create k clusters by associating every instance with the nearest mean based on a distance measure.
d. Replace the old means with the centroid of each of the k clusters (as the new means).
e. Repeat the above two steps until convergence (no change in each cluster center).

Part Two (Decision Tree Learning)

1. You should first apply your program to the hepatitis-training.dat and hepatitis-test.dat files and report the classification accuracy in terms of the fraction of the test instances that it classified correctly. Report the learned decision tree classifier. Compare the accuracy of your Decision Tree program to the baseline classifier, and comment on it.

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

With my decision tree algorithm, 21 out of 24 classifications are correct to give an accuracy of 77.78%. Compared to the 85.19% accuracy obtained using the baseline accuracy, which always predicts the most frequent class in the dataset.

While the baseline has better accuracy than my sorting algorithm this does not mean that using the baseline classification is better than the decision tree.

Because baseline only refers to the division of classes over a dataset using a baseline could have lower accuracy in different experiments and is considered not consistent enough to be an accurate classification method.

We can assume that using the decision tree will be more accurate than the baseline and have relatively the same level of accuracy on each data set.

2. You should then construct 10 other pairs of training/test files, train and test your classifiers on each pair, and calculate the average accuracy of the classifiers over the 10 trials.

Test	**Decision Tree Accuracy (%)**	**Baseline Accuracy (%)**
1	79.03	79.03
2	100	50
3	100	100
4	71.43	71.43
5	85.71	85.71
6	100	0
7	100	100
8	85.04	80.31
9	72.34	76.60
10	85.71	71.43
Average:	87.93	79.99

As the table shows, over the 10 tests, decision tree accuracy is, on average, greater than baseline accuracy. Because the data is random and contains either high or low quantity, it tends to favour the decision tree which is more robust and consistent with a much lower range of accuracy.

3. “Pruning” (removing) some of leaves of the decision tree would always make the decision tree less accurate on the training set. Explain (a) How you could prune leaves from the decision tree; (b) Why it would reduce accuracy on the training set, and (c)Why it might improve accuracy on the test set.

a. There are many implementations of decision tree pruning however the two most common are pre pruning and post pruning. Pre-pruning refers to stopping the growth of the tree early before it perfectly classifies the training set while post-pruning defines the process of perfectly classifying the training set and then pruning the tree. With pruning, the most important aspect is defining a criterion that determines the correct final tree size in one of the following ways:
    - Using a validation set to evaluate the effect of post pruning nodes from the tree
    - Building the tree using the training set and then applying a statistical test to estimate whether pruning or expanding a particular node will produce improvement (e.g. error estimation)
    - Using reduced error pruning in which starting at leaf nodes, each node is replaced by its most popular class and if accuracy is not affected, changes are retained otherwise nodes are changed back to their previous state.

b. Pruning will always reduce accuracy on the training set as the tree is originally generated by training set data so any change made without training set involvement or reduction in classifications will of course make it less accurate compared to having more classifications.

c. Because pruning reduces the complexity of the final classifier it can improve predictive accuracy through the reduction of overfitting. Overfitting occurs when the algorithm tightly fits the training set and perfectly fits all samples in training data so much that it becomes inaccurate on untrained data.

4. Explain why the impurity measure is not a good measure if there are three or more possible classes that the decision tree must distinguish.

With impurity measure,there are two simple criteria. if a class contains all of the instances, it should be zero, if there is a mixture of instances in the classes, more tests should be done.

If the decision tree has more than two possible classes, situations can arise that can cause complications. Because the decision tree can split multiple ways with one branch for each value we won't have generalisation, instead we should use information theoretical purity measures, such as the entropy impurity (shown below) which will work for multiple classes.

(Entropy) impurity = $-\sum_{class}[P(class)\log(1/P(class))]$

Part Three (Perceptron)

1. Report on the performance of your perceptron. For example, did it find a correct set of weights?

My perceptron successfully found a set of weights that classified all 100 images found inside the image.data file. Below is the feature weights and connections of features of running with 51 features, it took 392 cycles to be complete. It is interesting to note that increasing the number of features reduced the number of cycles it took to achieve success. As such, increasing the number of features from 51 to 80 reduces the cycle number down to 140 cycles.

Feature weight: 0.20352281699418867
[{5, 8, False}, {3, 4, False}, {4, 6, True}, {8, 9, False}]
Feature weight: -0.06778860065128739
[{2, 4, True}, {2, 6, False}, {6, 2, False}, {9, 4, False}]
Feature weight: -0.22267088405507662
[{7, 2, True}, {4, 0, False}, {3, 5, False}, {0, 5, True}]
Feature weight: 0.18078793794074077
[{7, 7, False}, {2, 0, False}, {0, 7, False}, {3, 6, False}]
Feature weight: 0.0030484171571952545
[{8, 9, False}, {4, 0, False}, {6, 3, False}, {6, 8, False}]
Feature weight: 0.03895062068538383
[{9, 6, True}, {8, 2, False}, {2, 1, False}, {7, 8, True}]
Feature weight: 0.22757259685591613
[{5, 9, False}, {5, 9, False}, {7, 6, False}, {0, 3, True}]
Feature weight: -0.0719142514794614
[{6, 0, False}, {2, 4, True}, {4, 4, False}, {3, 3, False}]
Feature weight: -0.13794169741876552
[{1, 3, True}, {6, 7, False}, {0, 9, False}, {3, 0, True}]
Feature weight: -0.1736998710215403
[{1, 9, False}, {9, 8, False}, {0, 5, True}, {6, 4, True}]
Feature weight: -0.09643371013058351
[{8, 8, False}, {2, 3, True}, {9, 5, False}, {3, 4, True}]
Feature weight: -0.10178270909408559
[{1, 5, False}, {4, 7, False}, {9, 6, True}, {9, 3, True}]
Feature weight: -0.14009975982614778
[{8, 5, True}, {8, 2, False}, {6, 0, False}, {9, 5, True}]
Feature weight: -0.06619803025122716
[{5, 6, True}, {7, 8, False}, {2, 9, True}, {0, 8, False}]
Feature weight: 0.3002123021405422
[{1, 5, False}, {6, 6, True}, {3, 8, True}, {7, 2, False}]
Feature weight: -0.1704175948907234
[{4, 1, False}, {9, 8, True}, {3, 3, False}, {0, 4, True}]
Feature weight: 0.08690758113167318
[{0, 8, True}, {6, 1, False}, {6, 5, True}, {0, 6, False}]
Feature weight: 0.022387545632253945
[{4, 6, False}, {9, 8, True}, {7, 8, True}, {2, 6, True}]
Feature weight: -0.09623578340540602
[{9, 8, False}, {4, 1, True}, {9, 5, False}, {4, 6, False}]
Feature weight: -0.12047040633307123
[{8, 7, False}, {9, 3, True}, {0, 1, False}, {0, 5, True}]
Feature weight: 0.3983432253521067
[{6, 3, True}, {1, 8, True}, {5, 9, False}, {8, 1, True}]
Feature weight: -0.34700600585230507
[{6, 1, True}, {9, 3, True}, {4, 0, True}, {4, 7, False}]
Feature weight: 0.11509367017438495
[{4, 3, True}, {4, 9, False}, {3, 8, True}, {6, 3, False}]
Feature weight: -0.008435091141972906
[{8, 1, False}, {3, 9, False}, {9, 9, False}, {7, 3, True}]
Feature weight: 0.026859203195334098
[{6, 2, False}, {4, 5, False}, {3, 8, False}, {9, 6, True}]
Feature weight: -0.026573986750375495
[{3, 7, False}, {0, 9, True}, {5, 7, True}, {4, 5, False}

Feature weight: -0.07704717924099823
[{2, 2, False}, {5, 5, False}, {9, 0, True}, {5, 2, False}]
Feature weight: -0.04422258091578986
[{7, 1, True}, {6, 3, True}, {9, 6, False}, {8, 7, False}]
Feature weight: -0.1081161487559964
[{4, 7, False}, {9, 0, True}, {4, 3, False}, {6, 3, True}]
Feature weight: -0.02981148099927556
[{1, 3, True}, {4, 8, False}, {7, 0, False}, {0, 3, False}]
Feature weight: 0.05789422138806305
[{1, 9, True}, {8, 3, True}, {0, 4, True}, {4, 2, False}]
Feature weight: -0.1231766538780412
[{4, 4, False}, {5, 5, False}, {3, 4, True}, {3, 3, False}]
Feature weight: -0.37982837719776547
[{9, 7, False}, {6, 6, True}, {0, 0, True}, {7, 7, False}]
Feature weight: 0.11580919718846368
[{1, 4, False}, {0, 2, False}, {4, 6, False}, {0, 5, True}]
Feature weight: 0.02665120061026401
[{6, 1, False}, {0, 3, False}, {3, 1, True}, {4, 4, True}]
Feature weight: -0.2156811525985204
[{8, 8, False}, {4, 8, False}, {3, 2, False}, {5, 9, False}]
Feature weight: -0.08181039574849015
[{5, 2, True}, {2, 7, False}, {1, 0, False}, {8, 5, True}]
Feature weight: 0.04331613121619206
[{7, 6, True}, {9, 9, False}, {7, 1, False}, {0, 6, True}]
Feature weight: 0.4573457688083813
[{4, 3, False}, {9, 9, True}, {9, 3, True}, {0, 3, False}]
Feature weight: 0.26770035134435083
[{2, 3, False}, {1, 9, True}, {0, 1, True}, {9, 6, False}]
Feature weight: 0.10424869858106935
[{8, 2, False}, {9, 6, False}, {9, 2, False}, {4, 6, True}]
Feature weight: 0.30874966447487634
[{1, 8, True}, {7, 8, False}, {5, 3, False}, {1, 4, False}]
Feature weight: 0.05787955051161985
[{1, 0, True}, {8, 0, False}, {7, 3, False}, {3, 6, True}]
Feature weight: -0.3108586580011158
[{8, 7, True}, {5, 4, False}, {1, 5, True}, {3, 4, False}]
Feature weight: 0.11132765419808295
[{3, 5, False}, {8, 3, True}, {5, 0, False}, {4, 6, True}]
Feature weight: -0.18329980320376404
[{7, 5, True}, {0, 9, False}, {9, 8, True}, {7, 2, False}]
Feature weight: 0.13029585718968784
[{3, 9, True}, {2, 1, False}, {0, 0, False}, {8, 7, False}]
Feature weight: -0.14456081570817192
[{8, 7, True}, {6, 4, True}, {6, 3, True}, {7, 5, False}]
Feature weight: -0.1944926940980264
[{4, 1, False}, {9, 8, False}, {4, 7, True}, {7, 8, False}]
Feature weight: 0.042917262523476246
[{6, 7, False}, {0, 3, True}, {3, 9, True}, {1, 6, False}]
Feature weight: -0.07496961094268938
[{6, 5, False}, {7, 5, False}, {6, 6, True}, {9, 0, False}]

2. Explain why evaluating the perceptron's performance on the training data is not a good measure of its effectiveness. You may wish to create additional data to get a better measure. If you do, report on the perceptron's performance on this additional data.

It is hard to fully evaluate the effectiveness of the perceptron based on the training data because we can not guarantee the perceptron doesn't have overfitting. Because the perceptron has been trained using this data file, the weights are tailored to fit this data set while another data set could potentially need completely different weights. Our perceptron is trained to get 100% accuracy on this data set alone and as such, the weights may not be generalised enough for classifying other data successfully.