Nicholas Tran, 300296259

COMP307 - Assignment 1:

Basic Machine Learning Algorithms

Part 1: k-Nearest Neighbour Method

Comparing Ks: k = 1 vs. k = 5:

Predicted class labels predicted by the k-nearest neighbours method where k = 1:

#	Predicted Class:	Actual Class:	#	Predicted Class:	Actual Class:	#	Predicted Class:	Actual Class:
0	3	3	***30***	2	2	***60***	3	3
1	3	3	***31***	1	1	***61***	1	2
2	3	3	***32***	2	2	***62***	2	2
3	1	1	***33***	1	1	***63***	3	3
4	1	1	***34***	2	2	***64***	2	2
5	1	1	***35***	2	2	***65***	3	3
6	1	2	***36***	2	2	***66***	3	3
7	2	2	***37***	2	2	***67***	1	1
8	1	1	***38***	2	2	***68***	1	1
9	2	2	***39***	1	1	***69***	2	2
10	2	2	***40***	2	2	***70***	1	2
11	3	2	***41***	2	2	***71***	3	3
12	3	3	***42***	3	3	***72***	2	2
13	3	3	***43***	1	1	***73***	2	2
14	1	1	***44***	2	2	***74***	1	1
15	2	2	***45***	1	1	***75***	1	1
16	3	3	***46***	3	3	***76***	1	1
17	3	3	***47***	2	2	***77***	3	3
18	1	1	***48***	2	2	***78***	1	1
19	1	1	***49***	1	1	***79***	1	1
20	3	3	***50***	3	3	***80***	2	2
21	2	2	***51***	1	1	***81***	2	2

Nicholas Tran, 300296259

22	2	2	**52**	1	1	**82**	3	3
23	3	3	**53**	3	3	**83**	1	1
24	2	2	**54**	3	3	**84**	2	2
25	3	2	**55**	1	1	**85**	1	1
26	2	2	**56**	1	1	**86**	1	1
27	3	3	**57**	3	3	**87**	2	2
28	2	2	**58**	1	1	**88**	1	1
29	1	1	**59**	3	3			

Testing Prediction Accuracy when k = 1: 84/89 = 94.38%

Testing Prediction Accuracy when k = 3: 85/89 = 95.51%

The difference between **k = 1** and **k = 3** is really marginal. It only differs in one more correct prediction when k = 3 (or a prediction accuracy increase of 1.13%). When k = 1 the model follows the training data too precisely and doesn't generalise well. The higher the k the less chance for the model to overfit the data. Increasing the k from 1 to 3 increased the test accuracy and theoretically as we increase k the higher the test accuracy but also at the cost of training accuracy (there's a k that exists where there's a middle ground between training accuracy and testing accuracy).

The advantages of the k-Nearest Neighbours Method:

- It's a lazy learner which means there's no training for the model. It just stores data and the algorithm uses it as a glorified look-up table.
- As a non-parametric model it makes zero assumptions about the distribution of the underlying data.
- It's easily implemented and easily understood.
- Adaptable to new training instances. As new data gets added, the algorithm can respond and learn quickly from the changes.

The disadvantages of the k-Nearest Neighbours Method:

- It can be computationally expensive. It's time complexity is O(n*m) where n is the test instances and m is the training instances.
- It can also be space consuming since it has to store all the training data first. As the dataset grows it can be space computationally expensive.
- Can be quite sensitive to noisy and messy data. Garbage in, garbage out etc.

Nicholas Tran, 300296259

k-Fold Cross Validation:

For a k = 5 k-fold cross validation you:

1. Split the data into 5 equal subsets
2. For each subset: treat the subset as the test set then the rest of the 4 subsets are treated as the training sets.
3. Train the model on the test set using the training sets, repeat 5 times rotating each of the 5 subsets as the test set so that each of the subsets is used as the test set at least once.
4. The results of the model with each of the subsets (folds) can then be averaged or combined.

k-Fold Cross Validation is useful for comparing two algorithms with one another and also useful when your data is small.

What happens when there's no class labels?

This is called unsupervised learning. Compared to supervised learning like the knn algorithm which is used to identify the classes of the instances, unsupervised learning is more about learning about the patterns, relationships and similarities between the instances.
K-means clustering is one technique used to cluster instances together based on the means or centroids of the instances. How the algorithm works is that:

- It initiates a k number of means
- Then it assigns the instances to respective clusters based on the closest means
- Reiterate and recalculate the mean
- Reassign the instances to the new means
- Repeat until convergence (when the clusters no longer change)

k-Means Clustering when k = 3:

Cluster 1:
[6, 7, 9, 15, 21, 22, 24, 26, 28, 30, 32, 34, 35, 37, 38, 40, 41, 47, 62, 64, 69, 70, 72, 73, 80, 81, 84, 87]
Number of instances in cluster 1: 28

Cluster 2:
[0, 1, 2, 10, 11, 12, 13, 16, 17, 20, 23, 25, 27, 36, 42, 44, 46, 50, 53, 54, 57, 59, 60, 63, 65, 66, 71, 77, 82]
Number of instances in cluster 2: 29

Cluster 3:
[3, 4, 5, 8, 14, 18, 19, 29, 31, 33, 39, 43, 45, 48, 49, 51, 52, 55, 56, 58, 61, 67, 68, 74, 75, 76, 78, 79, 83, 85, 86, 88]
Number of instances in cluster 3: 32

1st	knn	Actual	2nd	knn	Actual	3rd	knn	Actual

Nicholas Tran, 300296259

Cluster	Predicted Class:	Class:	Cluster	Predicted Class:	Class:	Cluster	Predicted Class:	Class:
0	3	3	***3***	1	1	***6***	1	2
1	3	3	***4***	1	1	***7***	2	2
2	3	3	***5***	1	1	***9***	2	2
10	2	2	***8***	1	1	***15***	2	2
11	3	2	***14***	1	1	***21***	2	2
12	3	3	***18***	1	1	***22***	2	2
13	3	3	***19***	1	1	***24***	2	2
16	3	3	***29***	1	1	***26***	2	2
17	3	3	***31***	1	1	***28***	2	2
20	3	3	***33***	1	1	***30***	2	2
23	3	3	***39***	1	1	***32***	2	2
25	3	2	***43***	1	1	***34***	2	2
27	3	3	***45***	1	1	***35***	2	2
36	2	2	***48***	2	2	***37***	2	2
42	3	3	***49***	1	1	***38***	2	2
44	2	2	***51***	1	1	***40***	2	2
46	3	3	***52***	1	1	***41***	2	2
50	3	3	***55***	1	1	***47***	2	2
53	3	3	***56***	1	1	***62***	2	2
54	3	3	***58***	1	1	***64***	2	2
57	3	3	***61***	1	2	***69***	2	2
59	3	3	***67***	1	1	***70***	1	2
60	3	3	***68***	1	1	***72***	2	2
63	3	3	***74***	1	1	***73***	2	2
65	3	3	***75***	1	1	***80***	2	2
66	3	3	***76***	1	1	***81***	2	2
71	3	3	***78***	1	1	***84***	2	2
77	3	3	***79***	1	1	***87***	2	2
82	3	3	***83***	1	1			
			85	1	1			
			86	1	1			
			88	1	1			

Nicholas Tran, 300296259

The k-means clustering is fairly accurate in grouping the same classes together. The first cluster clusters class 3 instances, the second cluster clusters the class 1 instances and the final cluster clusters the class 2 instances. Since we have labels for the instances we can compare the actual class labels to see how accurately the k-means algorithm clusters the instances, the k-means accurately classifies 82/89 instances compared to 85/89 from our knn model (k-means: 92.14%, knn: 95.51%).

k-Means Clustering when k = 5:

Cluster 1:
[0, 1, 2, 11, 12, 13, 16, 17, 20, 23, 25, 27, 42, 46, 50, 53, 54, 59, 60, 63, 65, 66, 71, 77, 82]
Number of instances in cluster 1: 25

Cluster 2:
[9, 10, 21, 34, 36, 37, 44, 47, 57, 72, 73]
Number of instances in cluster 2: 11

Cluster 3:
[7, 48, 80]
Number of instances in cluster 3: 3

Cluster 4:
[6, 15, 22, 24, 26, 28, 30, 32, 35, 38, 40, 41, 62, 64, 69, 70, 81, 84, 87]
Number of instances in cluster 4: 19

Cluster 5:
[3, 4, 5, 8, 14, 18, 19, 29, 31, 33, 39, 43, 45, 49, 51, 52, 55, 56, 58, 61, 67, 68, 74, 75, 76, 78, 79, 83, 85, 86, 88]
Number of instances in cluster 5: 31

Nicholas Tran, 300296259

1st Cluster	knn Predicted Class:	Actual Class:	2nd Cluster	knn Predicted Class:	Actual Class:	3rd Cluster	knn Predicted Class:	Actual Class:
0	3	3	***9***	2	2	***7***	1	2
1	3	3	***10***	3	2	***48***	2	2
2	3	3	***21***	2	2	***80***	2	2
11	3	2	***34***	2	2			
12	3	3	***36***	2	2			
13	3	3	***37***	2	2			
16	3	3	***44***	2	2			
17	3	3	***47***	2	2			
20	3	3	***57***	3	3			
23	3	3	***72***	2	2			
25	3	2	***73***	2	2			
27	3	3						
42	3	3						
46	3	3						
50	3	3						
53	3	3						
54	3	3						
59	3	3						
60	3	3						
63	3	3						
65	3	3						
66	3	3						
71	3	3						
77	3	3						
82	3	3						

Nicholas Tran, 300296259

4th Cluster	knn Predicted Class:	Actual Class:	5th Cluster	knn Predicted Class:	Actual Class:
6	2	2	***3***	1	1
15	2	2	***4***	1	1
22	2	2	***5***	1	1
24	2	2	***8***	1	1
26	2	2	***14***	1	1
28	2	2	***18***	1	1
30	2	2	***19***	1	1
32	2	2	***29***	1	1
35	2	2	***31***	1	1
38	2	2	***33***	1	1
40	2	2	***39***	1	1
41	2	2	***43***	1	1
62	2	2	***45***	1	1
64	2	2	***49***	1	1
69	2	2	***51***	1	1
70	2	2	***52***	1	1
81	2	2	***55***	1	1
84	2	2	***56***	1	1
87	2	2	***58***	1	1
			61	1	2
			67	1	1
			68	1	1
			74	1	1
			75	1	1
			76	1	1
			78	1	1
			79	1	1
			83	1	1
			85	1	1
			86	1	1
			88	1	1

When running the k-means algorithm with a k of 5, a lot of the boundary cases become easier to separate but the resulting clusters have class 1 and 3 in their respective clusters but class 2 becomes separated into 3 different groups.

Part 2: Decision Tree Learning Method

Classification of the dataset using the decision tree learning algorithm:

Baseline accuracy: 91/112 = 81.25%
Training accuracy: 112/112 = 100.0%
Test Accuracy: 20/25 = 80.0%

The algorithm performs quite well compared to the baseline classifier, the training evaluation is 100% accurate and the testing accuracy is only slightly lower than the baseline.

Decision Tree classification using 10-fold cross validation:

Fold: 0
Baseline Accuracy: 86/107 = 80.37%,
Training Accuracy: 107/107 = 100.0%,
Testing Accuracy: 23/30 = 77.0%

Fold: 1
Baseline Accuracy: 87/107 = 81.31%,
Training Accuracy: 107/107 = 100.0%,
Testing Accuracy: 24/30 = 80.0%

Fold: 2
Baseline Accuracy: 87/107 = 81.31%,
Training Accuracy: 107/107 = 100.0%,
Testing Accuracy: 24/30 = 80.0%

Fold: 3
Baseline Accuracy: 88/107 = 82.24%,
Training Accuracy: 107/107 = 100.0%,
Testing Accuracy: 25/30 = 83.0%

Fold: 4
Baseline Accuracy: 86/107 = 80.37%,
Training Accuracy: 107/107 = 100.0%,
Testing Accuracy: 25/30 = 83.0%

Fold: 5
Baseline Accuracy: 88/107 = 82.24%,
Training Accuracy: 107/107 = 100.0%,
Testing Accuracy: 23/30 = 77.0%

Fold: 6
Baseline Accuracy: 85/107 = 79.44%,
Training Accuracy: 107/107 = 100.0%,
Testing Accuracy: 25/30 = 83.0%

Fold: 7
Baseline Accuracy: 88/107 = 82.24%,
Training Accuracy: 107/107 = 100.0%,
Testing Accuracy: 26/30 = 87.0%

Fold: 8
Baseline Accuracy: 94/107 = 87.85%,
Training Accuracy: 107/107 = 100.0%,
Testing Accuracy: 17/30 = 56.99%

Fold: 9
Baseline...
Accuracy: 89/107 = 83.18%,
Training Accuracy: 107/107 = 100.0%,
Testing Accuracy: 22/30 = 73.0%

Average training accuracy: 100.0 %
Average testing accuracy: 78.0 %

The algorithm is fairly robust to differing combinations of the data. There's one fold which is an outlier, fold 8 which has the lowest test accuracy of only 56.99%. In all folds the training accuracy is 100% and besides from fold 8 the testing accuracy is around 73-87%. The average testing accuracy of the ten folds is 78%.

What is pruning?

Decision Tree Learning algorithms are highly susceptible to overfitting especially in cases where there's a lot of attributes - the deeper the tree the more specific thus it will be highly stylised to the training data which results in poor generalisation. To avoid overfitting there's a process called pruning which removes nodes from the tree that are redundant or poor estimators of the instance classes. There's two types of pruning: pre-pruning - which stops building the tree before it has completely classified the training set, and post-pruning - which removes nodes and trims the tree after the tree has already been built. In post-pruning each node is evaluated and if it doesn't meet a certain threshold then the node doesn't get split and is replaced with a leaf node instead. In pre-pruning a stopping criteria is implemented such as a maximum depth that the tree doesn't extend beyond. The trade-off with pruning is that the training accuracy diminishes due to less parameters being learned which can result in a model that isn't complex enough to explain the data, however it can also result in better testing accuracy due to it being better at generalising to unseen data.

The Impurity measure on 3 or more classes:

The impurity measure does not scale well the more classes there are. This is due to the fact that the more classes, the smaller the probabilities get and with smaller probabilities it becomes harder to distinguish between each of the cases.

Part 3: Perceptron

Accuracy of the Perceptron:

The Perceptron's training accuracy fluctuates between 85% and 90%. It appears that the ordering of the instances can influence the perceptron. The algorithm that was implemented shuffles the data due to the fact that splitting the dataset into train/test sets the final 20% or so of the data belongs to the same class, it became apparent while running the algorithm that the 100% accuracy on the test set vs the 90% on the training set was an anomaly. The algorithm itself doesn't converge, it never finds a set of weights that perfectly predicts the training instances. The stopping criteria of the algorithm is that it stops once an optimal solution is found or if the best accuracy doesn't improve for a number of epochs (in this case 100 but setting it higher at 1000 or 10000 does not improve the accuracy by much)

Training Only vs. Training and Test Split:

Training the Perceptron then testing it on the training set will cause overfitting where the model is extremely good at predicting those instances it has already seen but it doesn't generalise well so it results in high training accuracy but low or middling accuracy on data it has not seen. Training and testing on the same data is pointless because it can only explain the variation within that one dataset. It can introduce underlying biases that are unintentionally found within that particular set of data.

Training and testing on the same dataset:

Training perceptron with learning rate 1.0
Accuracy: 87.75%

Nicholas Tran, 300296259

Training and Testing on an 80:20 train/test split:

Training perceptron with learning rate 1.0
Training Accuracy: 90.71%

Testing perceptron with learning rate 1.0
Test Accuracy: 88.73%