Mezbah Uddin
ID: 500793378
Machine Learning, Project 1

K Nearest Neighborhood (KNN) for Contact lens dataset

For the purpose of the cross-validation the lens data set which has 24 observations, have been splitted into training set of 15 observations and testing set of 9 observations. The dataset contains categorical data of 4 independent variables and one dependent variable. In general, it is expected that decision tree and multilayer perceptions are expected to work better on contact lens data. Because, it has been seen from diverse studies that, those algorithms work best on categorical data.

The algorithm finds the nearest neighborhood using knn function on training data and then predictions on the testing data. The confusion matrix is given below for different k values –

K=1			K=3			K=5			K=7			K=9		
label	Pred	Match	label	Pred	Match	label	Pred	Match	label	Pred	match	Label	Pred	match
2	2	Match	2	2	Match	1	2	No Match	3	2	No Match	3	2	No Match
3	3	Match	3	3	Match	3	3	Match	3	3	Match	3	3	Match
3	3	Match	3	3	Match	3	3	Match	3	3	Match	3	3	Match
2	2	Match	2	2	Match	3	2	No Match	3	2	No Match	3	2	No Match
1	3	No Match	1	3	No Match	1	3	No Match	3	3	Match	1	3	No Match
3	3	Match	3	3	Match	3	3	Match	3	3	Match	3	3	Match
3	3	Match	3	3	Match	3	3	Match	3	3	Match	3	3	Match
3	3	Match	3	3	Match	3	3	Match	3	3	Match	3	3	Match
3	3	Match	2	3	No Match	3	3	Match	3	3	Match	3	3	Match

Table 1: Confusion Matrix knn(Euclidian) on Contact lens data

From the confusion matrix we see that for k=1 there is one classification error, for k=3 there are two, and for k=5,7,9 there are three errors. It is seen that **k=1 gives the best result,** because its categorical data with only 24 data points, and data is not very sparsely distanced. Thus, its best classifies is its nearest neighbor.

Next, I tried the *Manhattan distance function* to estimate knn, using knnVCN function. Important thing to note here is that, given the small training data set of 15 points which only represent small number of classes, the function only allows to choose up to k=3.

Below table gives the confusion matrix.

K=1			K=3		
label	*Pred*	*Match*	*label*	*Pred*	*Match*
3	2	No Match	1	2	No Match
3	3	Match	3	3	Match
3	3	Match	3	3	Match
2	2	Match	2	2	Match
1	3	No Match	1	3	No Match
3	3	Match	3	3	Match
3	3	Match	3	3	Match
3	3	Match	3	3	Match
1	3	No Match	1	3	No Match

Table 2: Confusion Matrix knnVCN(Manhattan) on Contact lens data

From the confusion matrix, we see that in both cases of k=1 & k=3, there are three classification errors out of 9 test labels, resulting an error rate of 33%.

Next, step is trying *Minkowski distance function* to estimate knn, using knnVCN function. Important thing to note here is that, again given the small training data set of 15 points which only represent small number of classes, the function only allows to choose up to k=3.

Below is the confusion matrix

K=1			K=3		
label	*Pred*	*Match*	*label*	*Pred*	*Match*
3	2	No Match	1	2	No Match
3	3	Match	3	3	Match
3	3	Match	3	3	Match
2	2	Match	2	2	Match
1	3	No Match	1	3	No Match
3	3	Match	3	3	Match
3	3	Match	3	3	Match
3	3	Match	3	3	Match
1	3	No Match	1	3	No Match

Table 3: Confusion Matrix knnVCN(Minkowski) on Contact lens data

From the confusion matrix, we see that in both cases of k=1 & k=3, there are three classification errors out of 9 test labels.

Next, step is trying *Binary distance function* to estimate knn, using knnVCN function. Important thing to note here is that, again given the small training data set of 15 points which only represent small number of classes, the function only allows to choose up to k=3.

Below is the confusion matrix

K=1			K=3		
label	*Pred*	*Match*	*label*	*Pred*	*Match*
3	2	No Match	3	2	No Match
3	3	Match	3	3	Match
3	3	Match	3	3	Match
3	2	No Match	3	2	No Match
3	3	Match	3	3	Match
3	3	Match	3	3	Match
3	3	Match	3	3	Match
3	3	Match	3	3	Match
3	3	Match	3	3	Match

Table 4: Confusion Matrix knnVCN(binary) on Contact lens data

From the confusion matrix, we see that in both cases of k=1 & k=3, there are two classification errors out of 9 test labels. Below table gives the summary of errors

Overall, it can be seen that **binary distance function** works best among all the distance function taken into consideration. This is because of the categorical nature of the data.

	Knn (Euclidian)					Knn(Manhattan)		Knn(Minkowski)		Knn(Binary)	
	K=1	**K=3**	**K=5**	**K=7**	**K=9**	**K=1**	**K=3**	**K=1**	**K=3**	**K=1**	**K=3**
Error(%)	11%	22%	33%	22%	33%	33%	33%	33%	33%	22%	22%

Table: Accuracy of the Knn's

K Nearest Neighborhood (KNN) for IRIS dataset

For the purpose of the cross-validation the lens data set which has 150 observations(numerical), have been splitted into training set of 94 observations and testing set of 56 observations. The dataset contains numerical(continuous) data of 4 independent variables and one dependent variable(classification variable).
In general, it is expected that knn and multilayer perception are expected to work best on this dataset, given the numerical nature of the data.
The algorithm finds the nearest neighborhood using knn function on training data and then predictions on the testing data. The confusion matrix is given below for different k values –

	K=1			K=3			K=5			K=7			K=9		
	sent	vers	virgn	sent	vers	virgn	sent	vers	virgn	sent	vers	virgn	sent	vers	virgn
sent	21	0	0	21	0	0	21	0	0	21	0	0	21	0	0
Vers	0	12	1	0	12	1	0	12	1	0	12	1	0	12	1
virgn	0	1	21	0	1	21	0	1	21	0	1	21	0	1	21

Table 5: Confusion Matrix knn(Euclidian) on Iris data

	Knn (Euclidian)				
	K=1	**K=3**	**K=5**	**K=7**	**K=9**
Error(%)	.03%	.03%	.03%	.03%	.03%

Table: Summary of classification error of knn(Euclidian)

From the confusion matrix, it can be seen that all cases of k, there are always two classification error out of 56 test labels.

	K=1			K=3			K=5			K=7			K=9		
	sent	vers	virgn	sent	vers	virgn	sent	vers	virgn	sent	vers	virgn	sent	vers	virgn
Virgn	0	1	21	0	1	21	0	1	21	0	1	20	0	1	21
sent	21	0	0	21	0	0	21	0	0	21	0	0	21	0	0
vers	0	12	1	0	12	1	0	12	1	0	12	2	0	12	1

Table 6: Confusion Matrix knn(Manhattan) on Iris data

Knn (Manhattan)					
	K=1	**K=3**	**K=5**	**K=7**	**K=9**
Error(%)	.03%	.03%	.03%	.05%	.03%

Table: Summary of classification error of knn(Manhattan)

Next, I tried the *Manhattan distance function* to estimate knn, using knnVCN function. Important thing to note here is that, since , the iris data set, has enough data points with fair distribution of classes(Species), k could be chosen up to 9. From the above table, we see that, for all k, there were always two classification errors out of 56 test labels. So, the error is very insignificant for knn using manhattan.

	K=1			**K=3**			**K=5**			**K=7**			**K=9**		
	sent	vers	virgn	sent	vers	virgn	sent	vers	virgn	sent	vers	virgn	sent	vers	virgn
Virgn	0	1	21	0	1	21	0	1	21	0	1	20	0	1	21
sent	21	0	0	21	0	0	21	0	0	21	0	0	21	0	0
vers	0	12	1	0	12	1	0	12	1	0	12	2	0	12	1

Table 7: Confusion Matrix knn(Minkowski) on Iris data

Knn (Minkowski)					
	K=1	**K=3**	**K=5**	**K=7**	**K=9**
Error(%)	.03%	.03%	.03%	.05%	.03%

Also for the minkowski distance function, there were maximum two classification error out of 56 testing set. So, overall Minkowski does best on iris data.

Decision Trees on Contact lens dataset

ID3 Tree:

For the ID3 implementation, I used the RWEKA package.

For the root node, tear should be chosen as it gives higher information gain (IG) based on the power it has on predicting the class variable.

For the algorithm implementation, I used the cross of training set of 19 observations and testing set of 5 observations. Below is the summary of ID3 implementation using training data –

Correctly Classified Instances	19	100 %
Incorrectly Classified Instances	0	0 %
Kappa statistic	1	
Mean absolute error	0	
Root mean squared error	0	
Relative absolute error	0 %	
Root relative squared error	0 %	
Total Number of Instances	19	

=== Confusion Matrix ===

```
 a  b  c   <-- classified as
 4  0  0 |  a = 1
 0  4  0 |  b = 2
 0  0 11 |  c = 3
```

for the ID3 there are no training errors.

Now, if the ID3 is implemented on testing data, the following result is produced –

ID3 Pred	Testing Labels
3	3
3	3
3	3
3	2

1	3

The table indicates **two** classification error.

J48 Tree

Next, I implemented the J48 trees to compare the performance with ID3 on same data split. Below gives the training summary –

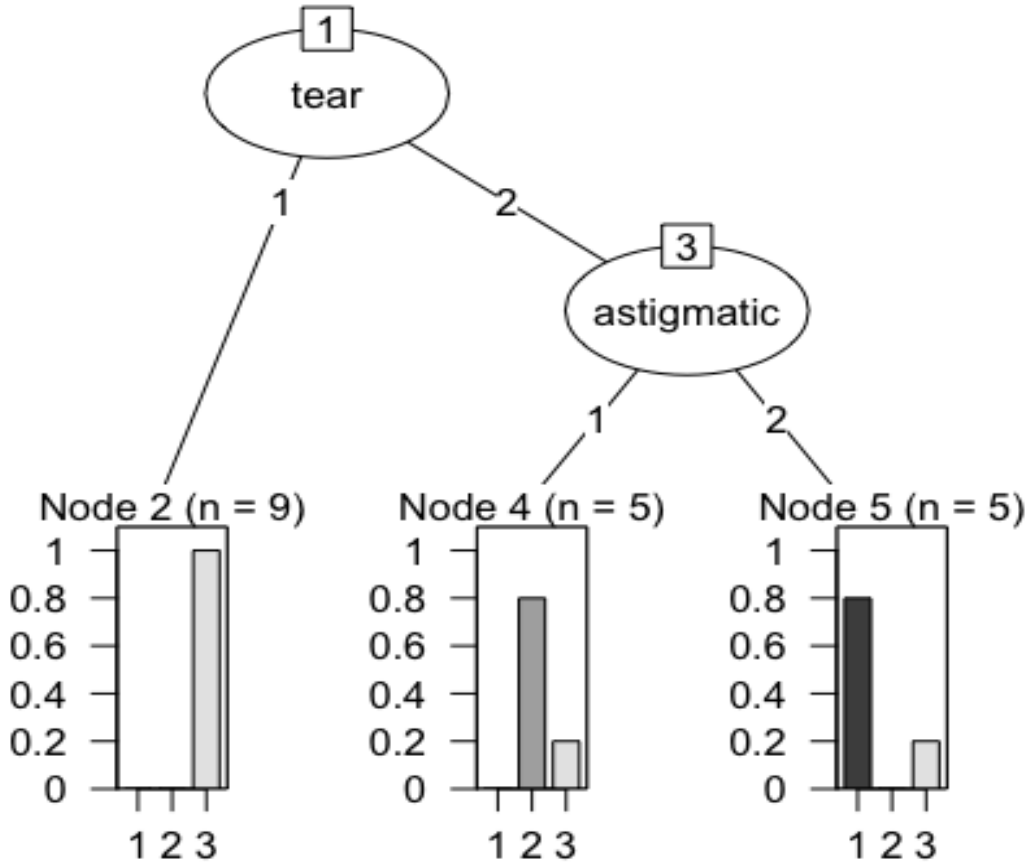
=== Summary ===

Correctly Classified Instances	17	89.4737 %
Incorrectly Classified Instances	2	10.5263 %
Kappa statistic	0.8288	
Mean absolute error	0.1123	
Root mean squared error	0.2369	
Relative absolute error	28.6179 %	
Root relative squared error	53.9867 %	
Total Number of Instances	19	

=== Confusion Matrix ===

```
 a b c   <-- classified as
 4 0 0 | a = 1
 0 4 0 | b = 2
 1 1 9 | c = 3
```

From above we see that there are two training errors. Also we get the following decision tree -



From the tree it can be confirmed that, indeed, tear, is chosen as the root node, as J48 is implemented based on IG.

Below gives the result of J48 on test labels –

J48 Pred	**Testing Labels**
3	3
3	3
3	3
2	2
1	3

Table indicates **one** classification error. This is understandable, as j48 is a refined version of ID3 and constructs decision tree, maximizing information gain, its expected to give better results. This has been true in lens data set case as well, by reducing error. Therefore, J48 gives a better performance than ID3.

Bagging Cart Tree

Bootstrapped Aggregation (Bagging) is an ensemble method that creates multiple models of the same type from different sub-samples of the same dataset. The predictions from each separate model are combined together to provide a superior result. This approach has shown participially effective for high-variance methods such as decision trees. Below table shows **one** classification error.

Bagging Cart Pred	**Testing Labels**
3	3
3	3
3	3
2	2
1	3

Prunning and overfitting

Pruning is a technique in machine learning that reduces the size of decision trees by removing sections of the tree that provide little power to classify instances. Pruning reduces the complexity of the final classifier, and hence improves predictive accuracy by the reduction of overfitting.

Pruning has been done contact lens data in J48, Id3 and Bagging Cart based on maximizing IG. And overall the effect has been less under fitting.

Finally, since contact lense data is categorical data, decision tree was expected to perform best on this data set and it has done reasonably well.

Decision Trees on Contact IRIS dataset

ID3

ID3 implementation is designed to be applied on categorical data. Since, IRIS data set has all numerical(continuous) as explanatory variable, ID3 can't be implemented.

J48 on Iris

Below gives the training summary of iris dataset using J48 implementation on the training data set of 93 obs.

Below is the training summary –

=== Summary ===

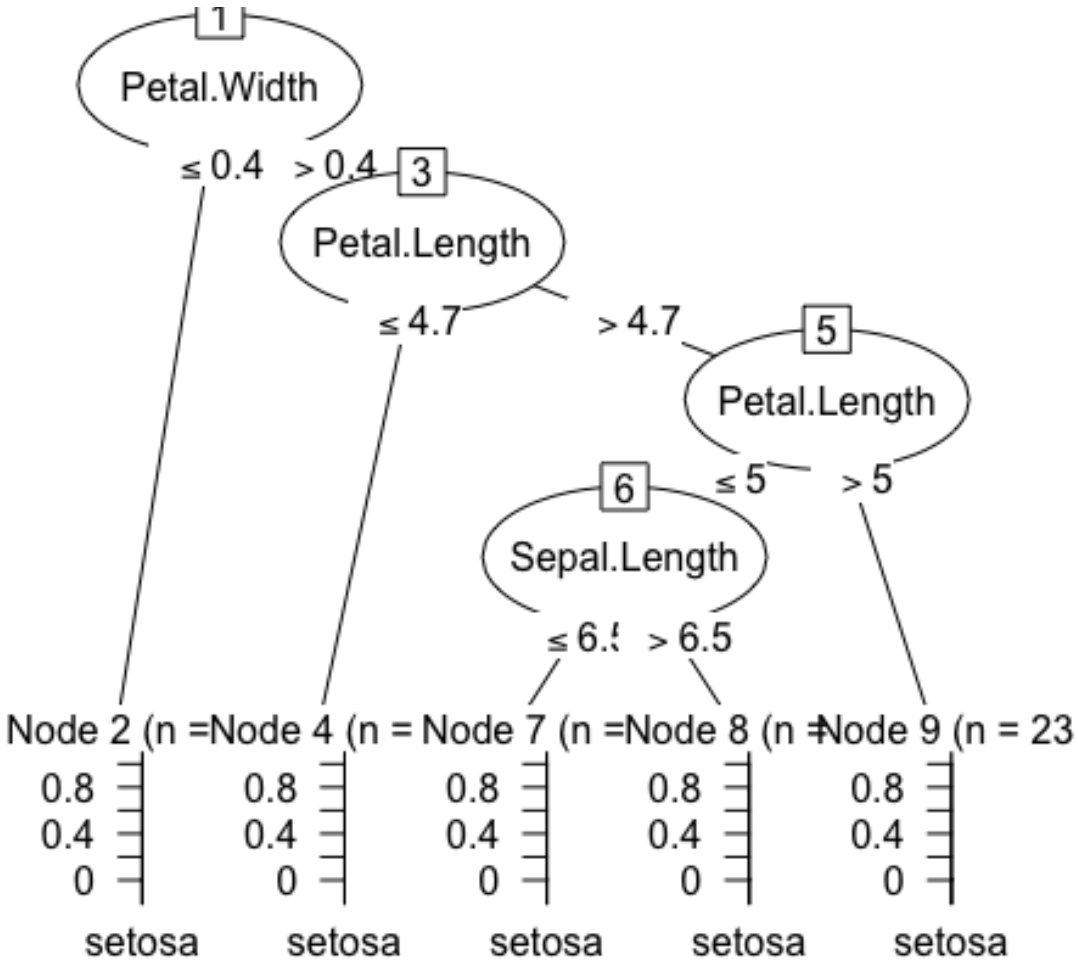
Correctly Classified Instances	92	98.9247 %
Incorrectly Classified Instances	1	1.0753 %
Kappa statistic	0.9839	
Mean absolute error	0.0139	
Root mean squared error	0.0833	
Relative absolute error	3.1228 %	
Root relative squared error	17.6715 %	
Total Number of Instances	93	

=== Confusion Matrix ===

```
  a  b  c   <-- classified as
 31  0  0 |  a = setosa
  0 30  0 |  b = versicolor
  0  1 31 |  c = virginica
```

Which shows there are **one training error of versicolor, which is about 1%.**

Also we get the following decision tree –



Below gives the result of J48 on test labels of iris–

	Setosa	Versicolor	Virginica
Setosa	19	0	0
Versicolor	0	19	4
Virginica	0	1	14

There are **five** classification errors.

Bagging Cart Tree

Below gives the confusion matrix after applying bagging cart on iris

	Setosa	Versicolor	Virginica
Setosa	19	0	0
Versicolor	0	19	3
Virginica	0	1	15

There are **four** classification errors.

Prunning and overfitting for IRIS

As we know from the theory that, pruning leads to less overfitting, in the case of iris, we also see that by decreasing classification error from 5 to 4 by changing from j48 to bagging cart.

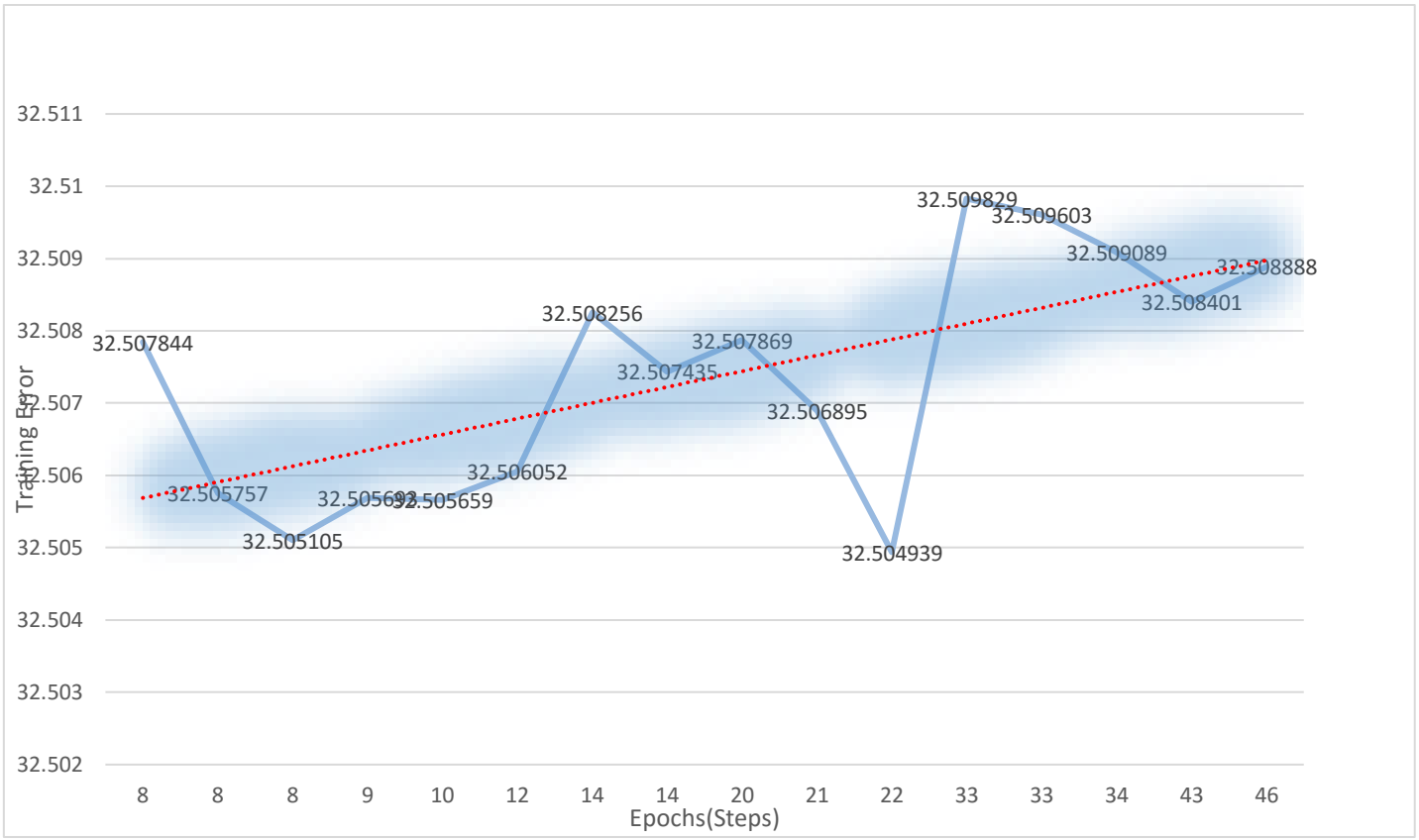
Though the decision tree works better with categorical data, even in the case of iris, it has performed reasonably well. I believe this has been mainly caused by a diverse data set, that has the ability to predict classes accurately.

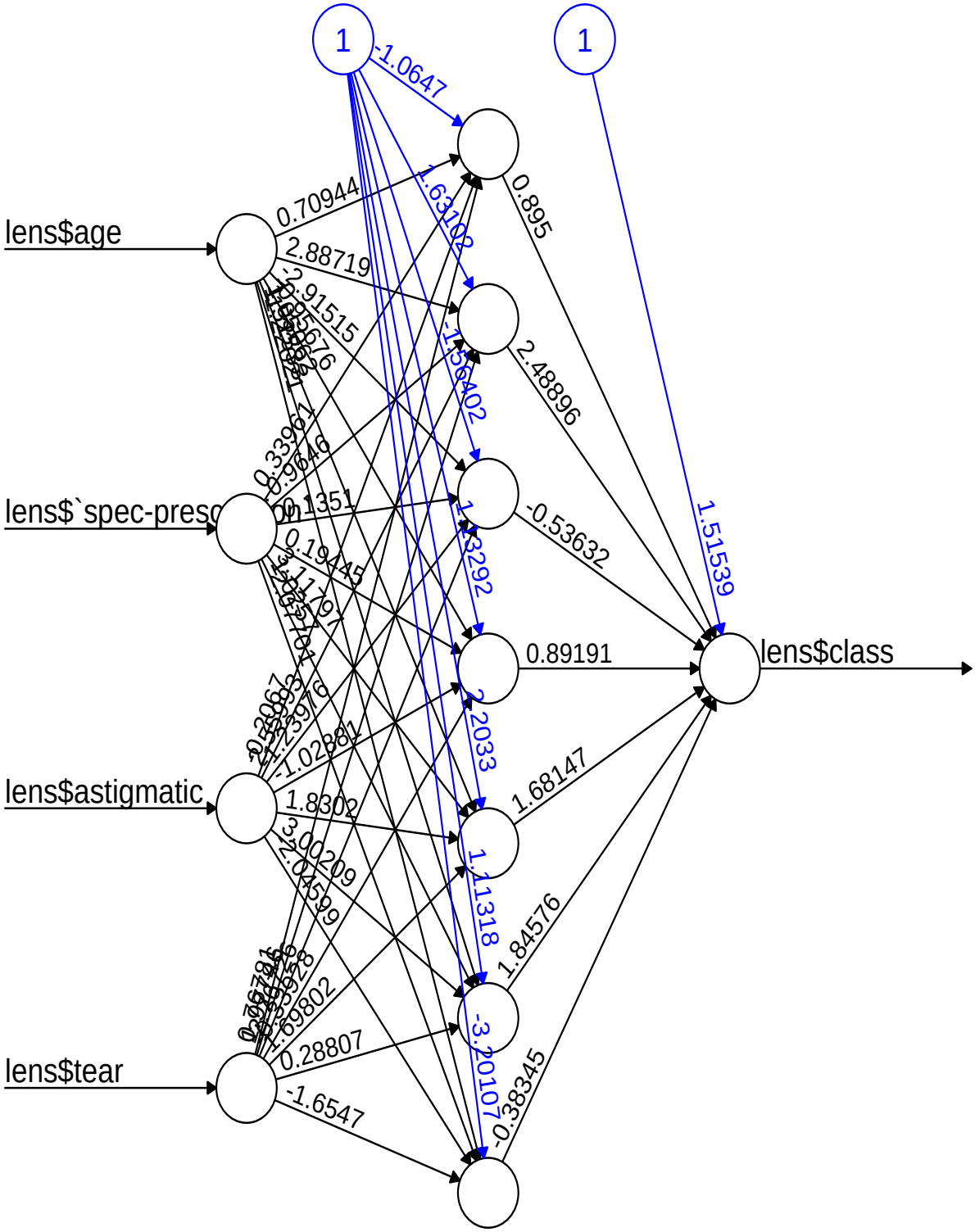
Multilayer Perceptions of Contact Lens Data

For this part of the project, neural let package has been applied on contact lens data. Different network structure and different hidden layers will be tried out to get a relationship between epochs and training errors.
Below table and graph gives an indication of the different trials –

Errors	Epochs
32.507844	8
32.505757	8
32.505105	8
32.505693	9
32.505659	10
32.506052	12
32.508256	14
32.507435	14
32.507869	20
32.506895	21
32.504939	22
32.509829	33
32.509603	33
32.509089	34
32.508401	43
32.5	46
32.507844	8

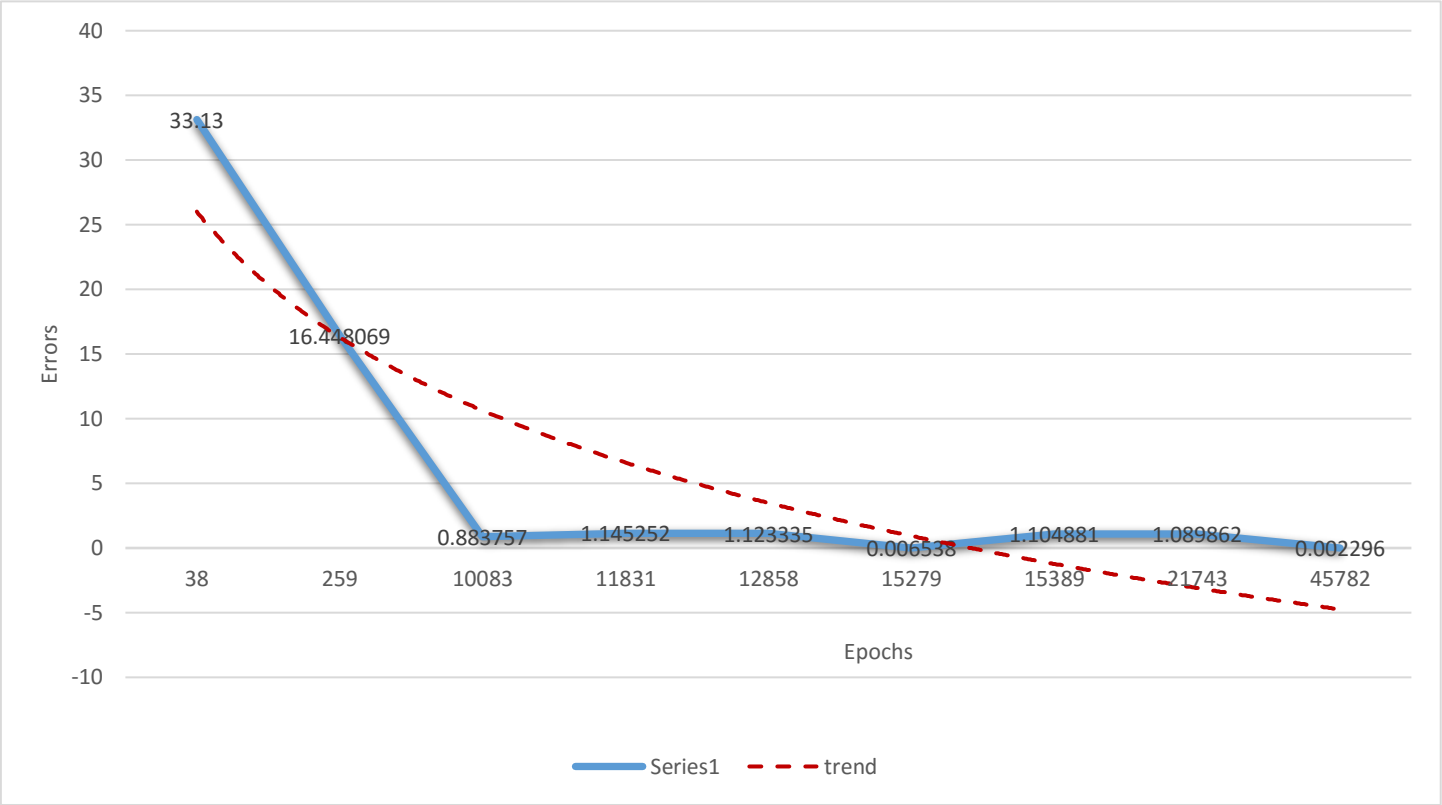
From the table and graph, it can be seen that training error goes down sharply after increasing steps until 8. However it starts increasing after that & falls again sharply after 20 steps and reaches global minimum at **22 steps.**


32.511
32.51
32.509
32.508
32.507
32.506
32.505
32.504
32.503
32.502
Training Error
32.507844
32.505757
32.505105
32.505693
32.505659
32.506052
32.508256
32.507435
32.507869
32.506895
32.504939
32.509829
32.509603
32.509089
32.508401
32.508888
8
8
8
9
10
12
14
14
20
21
22
33
33
34
43
46
Epochs(Steps)



Error: 32.508284 Steps: 16

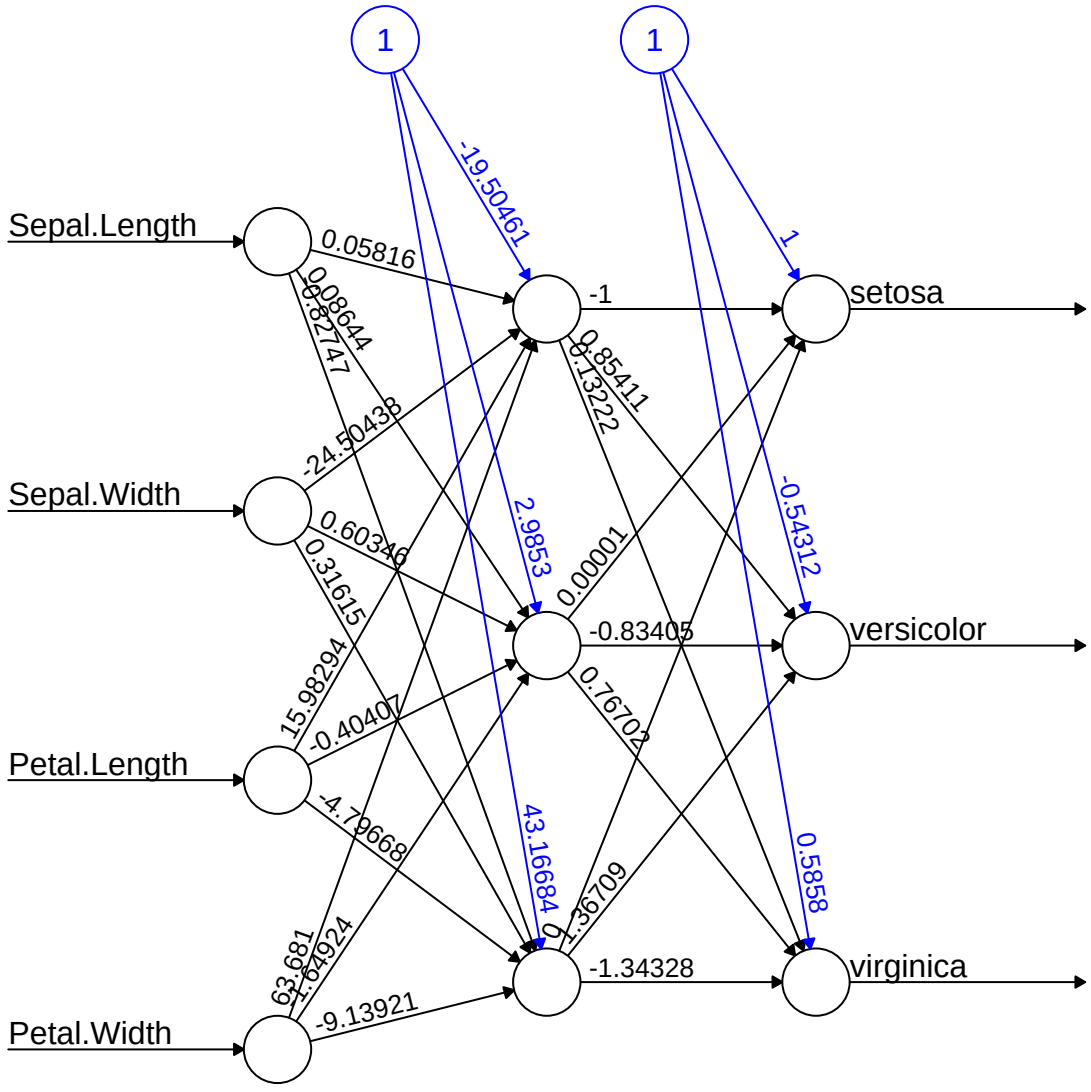
Multilayer Perceptions of Contact Iris Data



Errors	Epochs
33.13	38
16.448069	259
0.883757	10083
1.145252	11831
1.123335	12858
0.006538	15279
1.104881	15389
1.089862	21743
0.002296	45782
33.13	38
16.448069	259
0.883757	10083
1.145252	11831
1.123335	12858
0.006538	15279

1.104881	15389
1.089862	21743

We can see pretty good results for the iris dataset, where errors sharply decrease for every step increase and reaches **global lowest of 0 at 10000 steps.**



Error: 0.456121 Steps: 12733

General

Decision Tree has done best with contact lense data.Knn and multilayer perception worked best with IRIS data.

Overall, most algorithms performed per my expectations, except neural net on contact lens data. This could because of small datasets with lower predictability of its classes. There is a general rule in statistics that, data should be at least 30 to accurately train the model. Since the data set only had 24 observations, which on its own its low, and after cross validation it has reduced to even smaller number.

Overall, IRIS data set has best performance in general across all algorithms. This happened because of large dataset with evenly spread data, strong predictability of its class & testing/training set containing enough information gain to correctly classify the unknown instances.