COMP6011  First coursework

Brief introduction

In this document I aim to demonstrate analysis between two different machine learning algorithms. KNN (k-Nearest Neighbour) & Linear classification.

The notion of the classification problem

A classification problem is the concept that can be described as follows. Upon entering new data, we would have to classify these new data points.  Classification for these new data points are determined by a prediction. This predicted value will change due to the learning algorithm used and training data provided.   
  
Overall, the classification problem can be described as the task of determining whether an object fits into one of, with a minimum of two, different classes, outputs are discrete classes.

Explaining the IRIS dataset

The IRIS dataset was first introduced by Fisher,R.A. "The use of multiple measurements in taxonomic problems" Annual Eugenics, 7, Part II, 179-188 (1936). It is a highly reviewed database consisting of 2 datasets. Within these datasets are 3 different classes, each plant classification - Iris Setosa, -Iris Versicolour & Iris Virginica. With each of these classes having 50 instances. These instances contain the measurement of four different features:

* Sepal length in cm
* sepal width in cm
* petal length in cm
* petal width in cm

Explaining how both the KNN and linear classification as well as possible parameters

For this classification problem we are attempting two different machine learning algorithms. These are KNN and Linear Classification.

KNN

KNN is a lazy learner algorithm and works as the following:

* Assigning a value to K. The n of this number will determine how many neighbours (closest data points) you check.
* To determine the nearest neighbour, distance is calculated using a distance metric.
* N number of neighbours classes are taken and totalled. The most common class will be assumed by the new entry point. labelling it as the same class.

to summarize when a new data point is entered. It will scan over all past data, look up and compare the closest data.

Linear classification

Linear classification is a learning algorithm that labels based on a linear combination of inputs. You can visualise this but plotting data onto a graph. A linear equation may be plotted and should separate the data classes. Linear classification is generally used for problems with many variables as you can set a bias + feature vectors.

Explain how both Models can be applied to the Data set

Due to the database only containing two smaller datasets, we *should* be able to accurately predict our test data through our learning Algorithms. This would require feeding our training data to KNN / Linear classification via the fitcknn and fitcecoc functions.  
To compare the effectiveness of these learning algorithms we decided to use cross validation as our error measurement method. Cross validation is highly accurate however, it can be expensive due to the increased training time & may require high levels of computation (specifically for KNN). Fortunately, our database is small, and we should be able to avoid these issues.

my plan to calculate and compare these values are as follows:

* Randomise the order of the database to avoid overfitting. Ensuring we keep the same position for X,Y
* Split our data into a 5-fold cross validation. 4 Training folds and 1 test fold
* Train a KNN model
* Train the Linear classification Model
* Perform 5 runs when testing linear/K to get an average result.
* Use cross validation to on both learning algorithms to identify the algorithm optimal for our database

All graphs in the below section show blue hollow points and training data. Green as correct values and Red as incorrect values.

KNN graphs are: Linear Graph

fold 1 - figure 2 figure 3  
fold 2 - figure 4 figure 5

fold 3 - figure 6 figure 7  
fold4 - figure 8 figure 9  
fold5 - figure 10 figure 11

TestFull - figure 12 figure 13

KNN

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| K=1 |  |  |  |  |  |  |
| Fold Number | First30 | Second30 | Third30 | Fourth40 | Fifth30 | Train Data Average |
| 1 | 1 | 3 | 0 | 2 | 0 | 0% |
| 2 | 0 | 1 | 2 | 2 | 2 | 0% |
| 3 | 1 | 1 | 1 | 1 | 3 | 0% |
| 4 | 1 | 2 | 1 | 1 | 0 | 0% |
| 5 | 3 | 0 | 3 | 0 | 1 | 0% |
| total errors + test classification error | 6 (4%) | 7 (4.67%) | 7 (4.67%) | 6 (4%) | 7 (4.67%) | 0% |

A screenshot of a computer

Description automatically generated with medium confidence  
Example of K=1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| K=3 |  |  |  |  |  |  |
| Fold Number | First30 | Second30 | Third30 | Fourth40 | Fifth30 | Train Data Average |
| 1 | 2 | 0 | 2 | 0 | 2 | 4% |
| 2 | 1 | 1 | 0 | 1 | 1 | 4% |
| 3 | 1 | 1 | 1 | 2 | 1 | 4% |
| 4 | 1 | 1 | 2 | 2 | 1 | 4% |
| 5 | 1 | 2 | 1 | 0 | 0 | 4% |
| total errors + test classification error | 6 (4%) | 5 (3.3%) | 6 (4%) | 5 (3.3%) | 5 (3.3%) | 4% |

A screenshot of a computer

Description automatically generated with low confidence  
  
Example of K=3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| K=5 |  |  |  |  |  |  |
| Fold Number | First30 | Second30 | Third30 | Fourth40 | Fifth30 | Train Data Average |
| 1 | 1 | 1 | 1 | 2 | 0 | 3.3333%% |
| 2 | 0 | 1 | 0 | 2 | 1 | 3.3333%% |
| 3 | 2 | 2 | 3 | 1 | 0 | 3.3333%% |
| 4 | 2 | 2 | 0 | 1 | 2 | 3.3333%% |
| 5 | 1 | 0 | 1 | 0 | 1 | 3.3333%% |
| total errors + test classification error | 6 (4%) | 6 (4%) | 5 (6%) | 6 (4%) | 4 (2.6%) | 3.3333%% |

A picture containing text, computer, screenshot, indoor

Description automatically generated

example of K=5

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| K=8 |  |  |  |  |  |  |
| Fold Number | First30 | Second30 | Third30 | Fourth40 | Fifth30 | Train Data Average |
| 1 | 0 | 2 | 0 | 3 | 1 | 2% |
| 2 | 0 | 0 | 2 | 1 | 0 | 2% |
| 3 | 1 | 1 | 2 | 1 | 0 | 2% |
| 4 | 1 | 1 | 2 | 2 | 1 | 2% |
| 5 | 0 | 0 | 1 | 0 | 2 | 2% |
| total errors + test classification error | 2 (1.3%) | 4 (2.6%) | 7 (4.67%) | 7 (4.67%) | 4 (2.67%) | 2% |

A picture containing text, computer, screenshot

Description automatically generated

Example of K=8

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| K=10 |  |  |  |  |  |  |
| Fold Number | First30 | Second30 | Third30 | Fourth40 | Fifth30 | Train Data Average |
| 1 | 2 | 1 | 1 | 0 | 1 | 2% |
| 2 | 2 | 1 | 1 | 1 | 1 | 2% |
| 3 | 1 | 0 | 2 | 0 | 0 | 2% |
| 4 | 0 | 1 | 0 | 1 | 2 | 2% |
| 5 | 0 | 1 | 0 | 1 | 1 | 2% |
| total errors + test classification error | 5 (3.3%) | 4 (2%) | 4 (2%) | 3 (2%) | 5 (3.3%) | 2% |

A picture containing text, computer, screenshot, indoor

Description automatically generated

Example of K=10

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| K=13 |  |  |  |  |  |  |
| Fold Number | First30 | Second30 | Third30 | Fourth40 | Fifth30 | Train Data Average |
| 1 | 1 | 1 | 1 | 0 | 2 | 2% |
| 2 | 1 | 0 | 1 | 1 | 1 | 2% |
| 3 | 2 | 1 | 1 | 1 | 1 | 2% |
| 4 | 0 | 2 | 2 | 2 | 0 | 2% |
| 5 | 1 | 2 | 1 | 2 | 1 | 2% |
| total errors + test classification error | 5 (3.3%) | 6 (4%) | 6 (4%) | 6 (4%) | 5 (3.3%) | 2% |

A screenshot of a computer

Description automatically generated with low confidence

Example of K = 13

Linear Classification

A screenshot of a computer

Description automatically generated with low confidence

Average Errors

Linear Total Average

13 14 18 24 18 14 101    16  
  
KNN

33 27 27 24 21 28 160 26

Conclusion of results

Summary of Results

From testing the IRIS database with a 5-fold cross validation, it became clear that the higher the K value was the lower the error rate.  To ensure that our data wasn’t overfitted we randomised the data. Additionally, to draw an accurate conclusion we tested each fold 5 times, with each run being a randomised set of data.

From my findings the best value of K is 10 with its total amount of errors being 21 (14%). It is shown with k=13 that  further increasing  K may result with the error rate increasing.

Linear classification clearly showed that there is a higher accuracy compared to KNN.  Due to a lower (average) error rate of 16 (10.6%). Like KNN above, the script was run in total 25 times, with randomised data. To allow us to draw an accurate conclusion.

The difference between these error rates is 3.4% more accurate However, just because an algorithm is more accurate does not mean that it automatically is the best. Factors affecting the choice of an algorithm may include. Cost, time, and complexity. If we were to scale this operation up to 1000 data points within the dataset, linear would become more attractive. KNN would require a large increase in computational power + time to find an accurate value of K, this in turn would raise the cost of the experiment. As there are only two classes, complexity wouldn’t change. If we were to add more classes linear regression would become vastly more complicated. However, as our data is small and simple. Linear regression is the optimum classification model for this Iris Database.

*Values for results are https://drive.google.com/drive/folders/1VG\_f0ASzUDy1qjR5FEvRI4VePTAkqiA9?usp=sharing*

Code

*Sorry Matthias, the Maths spooked me, and this code is VERY inefficient.*

*%               Septal length Septal width Petal length Petal width*

*% Example of X 5.1000        3.5000   1.4000     0.2000*

*%               Species*

*% Example of Y  setosa*

*% fitcecoc + fitcknn are allowed*

*%https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761*

*clear all;*

*close all;*

*load fisheriris*

*X = meas;*

*Y = species;*

*%Randomise the ordering of Data*

*randomIndex = randperm(150);*

*X = X(randomIndex, :);*

*Y = Y(randomIndex);*

*% Setting up the training Data. teachingx/y will be fed into fitcknn to*

*% predict data, the last number is the value of K (how many nearest it*

*% checks)*

*% teachingx = X(1:100,:);*

*% teachingy = Y(1:100);*

*%nearest = fitcknn(teachingx, teachingy, 'NumNeighbors', 3);*

*%KNNguess = predict(nearest, [1, 2, 3, 4]), KNNguess does a prediction on the value (nearest, [-, -, -, -]*

*numdata = 150;*

*foldnumber = 30;*

*% Creates 5 sets of X testing Data*

*testX1 = X(1:foldnumber,:);*

*testX2 = X(foldnumber+1:2\*foldnumber,:);*

*testX3 = X(foldnumber\*2+1:3\*foldnumber,:);*

*testX4 = X(foldnumber\*3+1:4\*foldnumber,:);*

*testX5 = X(foldnumber\*4+1:5\*foldnumber,:);*

*% Creates 5 sets of Y testing data.*

*testY1 = Y(1:30,:);*

*testY2 = Y(31:60,:);*

*testY3 = Y(61:90,:);*

*testY4 = Y(91:120,:);*

*testY5 = Y(121:150,:);*

*%First section of Training Data*

*trainingX1 = X(foldnumber+1:numdata, :);*

*trainingY1 = Y(31:150);*

*%Second section of Training Data*

*trainingXa = X(61:150, :);*

*trainingXaa = X(1:30, :);*

*trainingYa = Y(61:150);*

*trainingYaa = Y(1:30);*

*trainingX2 = [trainingXa; trainingXaa];*

*trainingY2 = [trainingYa; trainingYaa];*

*%Third section of Training Data*

*trainingXb = X(91:150, :);*

*trainingXbb = X(1:60, :);*

*trainingYb = Y(91:150);*

*trainingYbb = Y(1:60);*

*trainingX3 = [trainingXb; trainingXbb];*

*trainingY3 = [trainingYb; trainingYbb];*

*%Fourth section of Training Data*

*trainingXc = X(121:150, :);*

*trainingXcc = X(1:90, :);*

*trainingYc = Y(121:150);*

*trainingYcc = Y(1:90);*

*trainingX4 = [trainingXc; trainingXcc];*

*trainingY4 = [trainingYc; trainingYcc];*

*%Fifth section of Training Data*

*trainingX5 = X(1:120, :);*

*trainingY5 = Y(1:120);*

*% Error Count*

*errorcount = 0;*

*lerrorcount = 0;*

*totalerrorcount = 0;*

*ltotalerrorcount = 0;*

*testclassificationerror = 0;*

*%K Value*

*k = 10;*

*%chosing what Graph to plot features - Septal length 1, Septal width2,*

*%Petal length 3 %Petal width 4*

*featurex = 1;*

*featurey = 2;*

*%%*

*%Plots full data set*

*figure(1);*

*gscatter(X( :, 1),X(:,2), species)*

*%%*

*% Test1KNN*

*knn = fitcknn(trainingX1, trainingY1, 'NumNeighbors', k);*

*%Plots test Data*

*for i=1:length(testX1)*

*hold on;*

*knnpredict = predict(knn,testX1(i, :));*

*knncorrect = strncmpi(knnpredict,testY1(i), 10);*

*% adds the Error count*

*if knncorrect == 0*

*errorcount = errorcount+1;*

*disp("KNN Test One Predicted Values is: " + knnpredict + ". Actual value is: " + testY1(i))*

*%disp(errorcount);*

*end*

*%plots correct values in Green, Errors in Red*

*if knncorrect == 0*

*figure(2);*

*%        title(['KNN Scatter for first block of data ' + foldnumber]);*

*%        xlabel('Sepal Lenght')*

*%        ylabel('Sepal length')*

*scatter(testX1(i,featurex), testX1(i,featurey),'red', 'filled')*

*else*

*figure(2);*

*scatter(testX1(i,featurex), testX1(i,featurey),'green', 'filled')*

*end*

*end*

*%Plots training Data*

*for i=1:150*

*scatter(X(i,featurex), X(i,featurey),'blue')*

*end*

*hold off*

*testclassificationerror = (errorcount/length(testX1))\*100;*

*totalerrorcount = totalerrorcount+errorcount;*

*disp("KNN Test 1 error count: " + errorcount + " test classification error is: " + testclassificationerror + "%")*

*errorcount = 0;*

*%%*

*%Test 1 Linear*

*linear = fitcecoc(trainingX1, trainingY1);*

*for j=1:length(testX1)*

*hold on;*

*linearpredict = predict(linear,testX1(j, :));*

*linearcorrect = strncmpi(linearpredict,testY1(j), 30);*

*if linearcorrect == 0*

*lerrorcount = lerrorcount+1;*

*disp("Linear Test One Predicted Values is: " + linearpredict + ". Actual value is: " + testY1(j))*

*end*

*if linearcorrect == 0*

*figure(3);*

*scatter(testX1(j,featurex), testX1(j,featurey),'red', 'filled')*

*else*

*figure(3);*

*scatter(testX1(j,featurex), testX1(j,featurey), 'green', 'filled')*

*end*

*end*

*for i=1:150*

*figure(3);*

*scatter(X(i,featurex), X(i,featurey),'blue')*

*end*

*ltotalerrorcount = ltotalerrorcount + lerrorcount;*

*testclassificationerror = (lerrorcount/length(testX1))\*100;*

*disp("linear Test 1 errorcount is: " + lerrorcount + " Test classification error is: " + testclassificationerror + "%")*

*lerrorcount = 0;*

*hold off;*

*%%*

*% Test2KNN*

*knnb = fitcknn(trainingX2, trainingY2, 'NumNeighbors', k);*

*%Plots test Data*

*for i=1:length(testX2)*

*hold on;*

*knnpredictb = predict(knnb,testX2(i, :));*

*knncorrectb = strncmpi(knnpredictb,testY2(i), 10);*

*% adds the Error count*

*if knncorrectb == 0*

*errorcount = errorcount+1;*

*disp("KNN Test Two Predicted Values is: " + knnpredict + ". Actual value is: " + testY2(i));*

*%disp(errorcount);*

*end*

*%plots correct values in Green, Errors in Red*

*if knncorrectb == 0*

*figure(4);*

*scatter(testX2(i,featurex), testX2(i,featurey),'red', 'filled')*

*else*

*figure(4);*

*scatter(testX2(i,featurex), testX2(i,featurey),'green', 'filled')*

*end*

*end*

*%Plots training Data*

*for i=1:150*

*scatter(X(i,featurex), X(i,featurey),'blue')*

*end*

*hold off*

*testclassificationerror = (errorcount/length(testX2))\*100;*

*totalerrorcount = totalerrorcount+errorcount;*

*disp("KNN Test 2 error count: " + errorcount + " test classification error is: " + testclassificationerror +"%")*

*errorcount = 0;*

*%%*

*%Test Two Linear*

*linear = fitcecoc(trainingX2, trainingY2);*

*for j=1:length(testX2)*

*hold on;*

*linearpredict = predict(linear,testX2(j, :));*

*linearcorrect = strncmpi(linearpredict,testY2(j), 30);*

*if linearcorrect == 0*

*lerrorcount = lerrorcount+1;*

*disp("Linear Test Two Predicted Values is: " + linearpredict + ". Actual value is: " + testY2(j))*

*end*

*if linearcorrect == 0*

*figure(5);*

*scatter(testX2(j,featurex), testX2(j,featurey),'red', 'filled')*

*else*

*figure(5);*

*scatter(testX2(j,featurex), testX2(j,featurey), 'green', 'filled')*

*end*

*end*

*for i=1:150*

*figure(5);*

*scatter(X(i,featurex), X(i,featurey),'blue')*

*end*

*ltotalerrorcount = ltotalerrorcount + lerrorcount;*

*testclassificationerror = (lerrorcount/length(testX2))\*100;*

*disp("linear Test 2 errorcount is: " + lerrorcount + " Test classification error is: " + testclassificationerror + "%")*

*lerrorcount = 0;*

*hold off;*

*%%*

*% Test3KNN*

*knnb = fitcknn(trainingX3, trainingY3, 'NumNeighbors', k);*

*%Plots test Data*

*for i=1:length(testX3)*

*hold on;*

*knnpredictb = predict(knnb,testX3(i, :));*

*knncorrectb = strncmpi(knnpredictb,testY3(i), 10);*

*% adds the Error count*

*if knncorrectb == 0*

*errorcount = errorcount+1;*

*disp("KNN Test Three Predicted Values is: " + knnpredict + ". Actual value is: " + testY3(i));*

*%disp(errorcount);*

*end*

*%plots correct values in Green, Errors in Red*

*if knncorrectb == 0*

*figure(6);*

*scatter(testX3(i,featurex), testX3(i,featurey),'red', 'filled')*

*else*

*figure(6);*

*scatter(testX3(i,featurex), testX3(i,featurey),'green', 'filled')*

*end*

*end*

*%Plots training Data*

*for i=1:150*

*scatter(X(i,featurex), X(i,featurey),'blue')*

*end*

*hold off*

*testclassificationerror = (errorcount/length(testX3))\*100;*

*totalerrorcount = totalerrorcount+errorcount;*

*disp("KNN Test 3 error count: " + errorcount + " test classification error is: " + testclassificationerror + "%")*

*errorcount = 0;*

*%%*

*%Test three Linear*

*linear = fitcecoc(trainingX3, trainingY3);*

*for j=1:length(testX3)*

*hold on;*

*linearpredict = predict(linear,testX3(j, :));*

*linearcorrect = strncmpi(linearpredict,testY3(j), 30);*

*if linearcorrect == 0*

*lerrorcount = lerrorcount+1;*

*disp("Linear Test Three Predicted Values is: " + linearpredict + ". Actual value is: " + testY3(j))*

*end*

*if linearcorrect == 0*

*figure(7);*

*scatter(testX3(j,featurex), testX3(j,featurey),'red', 'filled')*

*else*

*figure(7);*

*scatter(testX3(j,featurex), testX3(j,featurey), 'green', 'filled')*

*end*

*end*

*for i=1:150*

*figure(7);*

*scatter(X(i,featurex), X(i,featurey),'blue')*

*end*

*ltotalerrorcount = ltotalerrorcount + lerrorcount;*

*testclassificationerror = (lerrorcount/length(testX3))\*100;*

*disp("linear Test 3 errorcount is: " + lerrorcount + " Test classification error is: " + testclassificationerror + "%")*

*lerrorcount = 0;*

*hold off;*

*%%*

*% Test4KNN*

*knnb = fitcknn(trainingX4, trainingY4, 'NumNeighbors', k);*

*%Plots test Data*

*for i=1:length(testX4)*

*hold on;*

*knnpredictb = predict(knnb,testX4(i, :));*

*knncorrectb = strncmpi(knnpredictb,testY4(i), 10);*

*% adds the Error count*

*if knncorrectb == 0*

*errorcount = errorcount+1;*

*disp("KNN Test four Predicted Values is: " + knnpredict + ". Actual value is: " + testY4(i));*

*%disp(errorcount);*

*end*

*%plots correct values in Green, Errors in Red*

*if knncorrectb == 0*

*figure(8);*

*scatter(testX4(i,featurex), testX4(i,featurey),'red', 'filled')*

*else*

*figure(8);*

*scatter(testX4(i,featurex), testX4(i,featurey),'green', 'filled')*

*end*

*end*

*%Plots training Data*

*for i=1:150*

*scatter(X(i,featurex), X(i,featurey),'blue')*

*end*

*hold off*

*testclassificationerror = (errorcount/length(testX4))\*100;*

*totalerrorcount = totalerrorcount+errorcount;*

*disp("KNN Test 4 error count: " + errorcount + " test classification error is: " + testclassificationerror + "%")*

*errorcount = 0;*

*%%*

*%Test four Linear*

*linear = fitcecoc(trainingX4, trainingY4);*

*for j=1:length(testX4)*

*hold on;*

*linearpredict = predict(linear,testX4(j, :));*

*linearcorrect = strncmpi(linearpredict,testY4(j), 30);*

*if linearcorrect == 0*

*lerrorcount = lerrorcount+1;*

*disp("Linear Test Four Predicted Values is: " + linearpredict + ". Actual value is: " + testY4(j))*

*end*

*if linearcorrect == 0*

*figure(9);*

*scatter(testX3(j,featurex), testX3(j,featurey),'red', 'filled')*

*else*

*figure(9);*

*scatter(testX3(j,featurex), testX3(j,featurey), 'green', 'filled')*

*end*

*end*

*for i=1:150*

*figure(9);*

*scatter(X(i,featurex), X(i,featurey),'blue')*

*end*

*ltotalerrorcount = ltotalerrorcount + lerrorcount;*

*testclassificationerror = (lerrorcount/length(testX4))\*100;*

*disp("linear Test 4 errorcount is: " + lerrorcount + " Test classification error is: " + testclassificationerror + "%")*

*lerrorcount = 0;*

*hold off;*

*%%*

*% Test5KNN*

*knnb = fitcknn(trainingX5, trainingY5, 'NumNeighbors', k);*

*%Plots test Data*

*for i=1:length(testX5)*

*hold on;*

*knnpredictb = predict(knnb,testX5(i, :));*

*knncorrectb = strncmpi(knnpredictb,testY5(i), 10);*

*% adds the Error count*

*if knncorrectb == 0*

*errorcount = errorcount+1;*

*disp("KNN Test Five Predicted Values is: " + knnpredict + ". Actual value is: " + testY5(i));*

*%disp(errorcount);*

*end*

*%plots correct values in Green, Errors in Red*

*if knncorrectb == 0*

*figure(10);*

*scatter(testX5(i,featurex), testX5(i,featurey),'red', 'filled')*

*else*

*figure(10);*

*scatter(testX5(i,featurex), testX5(i,featurey),'green', 'filled')*

*end*

*end*

*%Plots training Data*

*for i=1:150*

*scatter(X(i,featurex), X(i,featurey),'blue')*

*end*

*hold off*

*testclassificationerror = (errorcount/length(testX5))\*100;*

*totalerrorcount = totalerrorcount+errorcount;*

*disp("KNN Test 5 error count: " + errorcount + " test classification error is: " + testclassificationerror + "%")*

*errorcount = 0;*

*%%*

*%Test five Linear*

*linear = fitcecoc(trainingX5, trainingY5);*

*for j=1:length(testX5)*

*hold on;*

*linearpredict = predict(linear,testX5(j, :));*

*linearcorrect = strncmpi(linearpredict,testY5(j), 30);*

*if linearcorrect == 0*

*lerrorcount = lerrorcount+1;*

*disp("Linear Test Five Predicted Values is: " + linearpredict + ". Actual value is: " + testY5(j))*

*end*

*if linearcorrect == 0*

*figure(11);*

*scatter(testX5(j,featurex), testX5(j,featurey),'red', 'filled')*

*else*

*figure(11);*

*scatter(testX5(j,featurex), testX5(j,featurey), 'green', 'filled')*

*end*

*end*

*% testclarficationError = (errorcount/length(testX5))\*100  //Test case Average*

*for i=1:150*

*figure(11);*

*scatter(X(i,featurex), X(i,featurey),'blue')*

*end*

*ltotalerrorcount = ltotalerrorcount + lerrorcount;*

*testclassificationerror = (lerrorcount/length(testX5))\*100;*

*disp("linear Test 5 errorcount is: " + lerrorcount + " Test classification error is: " + testclassificationerror + "%")*

*lerrorcount = 0;*

*hold off;*

*%%*

*% TrainFullyKNN*

*knnb = fitcknn(X, Y, 'NumNeighbors', k);*

*%Plots test Data*

*for i=1:length(X)*

*hold on;*

*knnpredictb = predict(knnb,X(i, :));*

*knncorrectb = strncmpi(knnpredictb,Y(i), 10);*

*% adds the Error count*

*if knncorrectb == 0*

*errorcount = errorcount+1;*

*disp("KNN train Predicted Values is: " + knnpredict + ". Actual value is: " + Y(i));*

*%disp(errorcount);*

*end*

*%plots correct values in Green, Errors in Red*

*if knncorrectb == 0*

*figure(12);*

*scatter(X(i,featurex), X(i,featurey),'red', 'filled')*

*else*

*figure(12);*

*scatter(X(i,featurex), X(i,featurey),'green', 'filled')*

*end*

*end*

*%Plots training Data*

*for i=1:150*

*scatter(X(i,featurex), X(i,featurey),'blue')*

*end*

*hold off*

*testclassificationerror = (errorcount/length(X))\*100;*

*disp("KNN Train error count: " + errorcount + " test classification error is: " + testclassificationerror + "%")*

*errorcount = 0;*

*%%*

*%Train Linear*

*linear = fitcecoc(X, Y);*

*for j=1:length(X)*

*hold on;*

*linearpredict = predict(linear,X(j, :));*

*linearcorrect = strncmpi(linearpredict,Y(j), 30);*

*if linearcorrect == 0*

*lerrorcount = lerrorcount+1;*

*disp("Linear Train Predicted Values is: " + linearpredict + ". Actual value is: " + Y(j))*

*end*

*if linearcorrect == 0*

*figure(13);*

*scatter(X(j,featurex), X(j,featurey),'red', 'filled')*

*else*

*figure(13);*

*scatter(X(j,featurex), X(j,featurey), 'green', 'filled')*

*end*

*end*

*for i=1:150*

*figure(13);*

*scatter(X(i,featurex), X(i,featurey),'blue')*

*end*

*testclassificationerror = (lerrorcount/length(X))\*100;*

*disp("linear Train errorcount is: " + lerrorcount + " Test classification error is: " + testclassificationerror + "%")*

*lerrorcount = 0;*

*hold off;*

*%%*

*knnperce = (totalerrorcount/150)\*100;*

*linearperce = (ltotalerrorcount/150)\*100;*

*disp("The total error count for KNN is: " + totalerrorcount + " error rate is: " + knnperce +"%")*

*disp("The total error count for Linear Regression is: " + ltotalerrorcount + " error rate is: " + linearperce +"%")*