ECE 1518

Seminar in Identity, Privacy and Security

Assignment 2: Hand Geometry based Person Identification

by

Madathingal Shreya Kishore
Student no: 1005620151

Date Due: 21st February 2019
Date Handed in: 20th February 2019

1. Introduction

Hand geometry is an excellent biometric for identification because it is less intrusive than others like fingerprints or iris and also perceived as non-threatening.[1] Such systems are easy to use with relatively high accuracy. The main drawback is that this characteristic is not very unique. Another limitation that the user psychology may prevent them from using devices used my others.[1]

The main aim of this assignment was to make a classifier which can identify different classes based on their hand geometry co-ordinates provided. Matlab is the platform which is used for this purpose.

2. Feature Extraction

Feature selection is a very important aspect of machine learning. The correct features should be chosen based on the application. For example, to differentiate between flowers and leaves, color can be one of the main varying factors followed by shape, length, width, etc. It is best to identify features which give a good sense of proportionality. With this point in mind, the features taken into consideration were the lengths and widths of the five fingers along with the palm width. The length was calculated by finding the mean between the points at the valleys between the fingers and then computing its distance from the top of the finger. Figure 1 is a good illustration of the features used for this assignment.

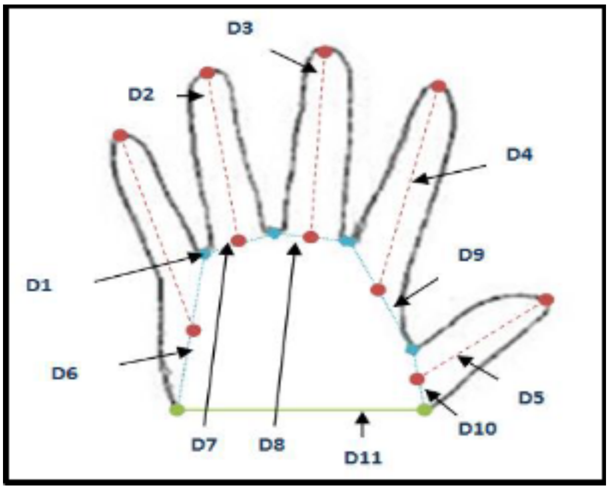

Figure 1. Distances between co-ordinates [2]

Euclidean distance formula was used for finding out all the lengths and widths. It is the optimal choice because it is fast and easy to accurate. Based on the fact that x and y coordinates of various points corresponding to a palm print were provided, this method is best suited for such data.

3. Choice of Classifier

Classification can be defined as grouping things based on their shared features. There are different classifiers which can be used ranging from simple to more complex neural networks.

The first thing done was to determine which models were unsuitable for the given data set in order to choose an appropriate classifier. Regression was eliminated because it was used for real numbers rather than classes. Decision trees give precedence to one feature over the other and forms a hierarchy based on this. Since this couldn't be done in this case, it was out of the picture. Support vector machine separates classes by distinctive planes. I thought it was more suitable for cases where there are fewer classes. The next two to be considered were KNN and LDA as I thought them to be the best option.

Table 1. Characteristics of classifiers [6]

Classifier	Multiclass support	Prediction Speed	Memory usage	Interpretability	Types of predictors
Decision tree	Yes	Fast	Small	Easy	Numeric, categorical and a mix of both
Support Vector Machine	No	Mostly slow	Medium for linear and large for binary.	Easy for linear SVM and hard for others.	All types
Naïve Bayes	Yes	Medium for simple distributions otherwise slow.	Medium for high-dimensional data	Easy	All types
KNN	Yes	Medium	Medium	Hard	Numeric-Euclidean distance only Categorical - Hamming distance
Discriminant analysis	Yes	Fast	Small for linear	Easy	Only numeric
Ensemble	Yes	Fast to medium based on choice of algorithm.	Low to medium.	Hard	All

i. KNN (nearest neighbour method) : In this algorithm, Euclidean distances are used to intuitively form polygon called Voronoi cell around true points. This classifier is based on feature similarity. KNN makes no generalization and hence is very quick.[5] The output is classes which are decided on the basis of the majority vote of the neighbours. Though it is fast, it uses a lot of memory because it stores the whole training set.

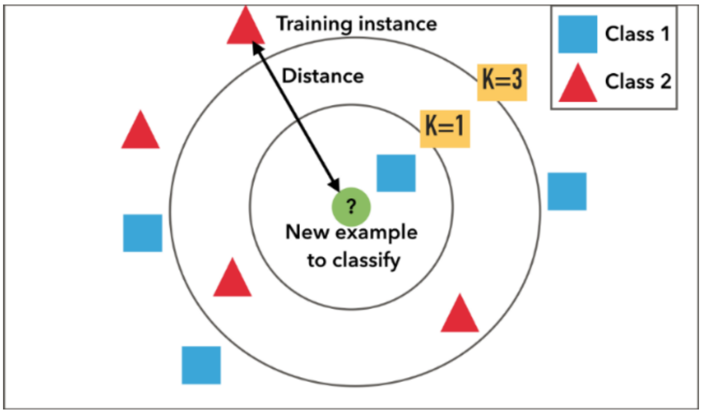

Figure 2. Example of KNN classification

ii. LDA (linear discriminant analysis) : This is best in case of multiple classes. The means and covariance of every predictor is calculated to get a Gaussian representation. The model uses Bayes theorem to predict the probability that a new set of inputs belongs to each class.[7] LDA can only be used in case that the classes are categorical, which is true in our case.

iii. Ensemble classifiers: They are hybrid kind of classifiers. The subspace ensemble classifiers use KNN or LDA and are best suited for multiple classes. They use less memory and can also handle missing data. Subspace type of ensemble works well for multiple class outputs.

Based on all the characteristics of classifiers, the one that is best suited for the function of identifying classes based on hand geometry is an ensemble classifier using linear discriminant analysis. This is due to the fact that it has best accuracy for a large number of classes, uses less memory and is easy to interpret. It also works best for numerical predictors and categorical outputs.

4. Preliminary Analysis

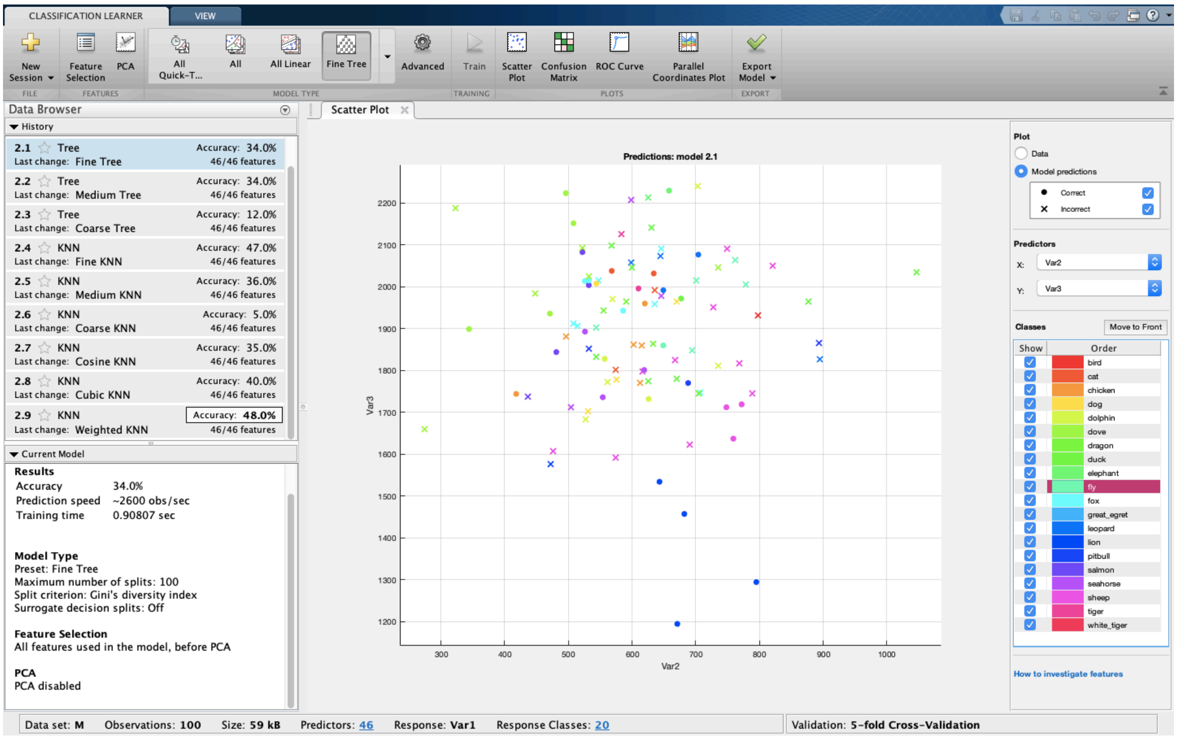

Figure 3. Scatterplot 1

For a basic understanding, I used the classification learner app in Matlab. The data was converted into a .csv file and imported into Matlab. On a quick analysis, it is shown that Weighted KNN has the best accuracy in the least time as shown in Figure 3.

Then, the feature extraction was done. I tried different combinations of features and classifiers with varying results. But, these eleven features were found to be the the most suitable. Figure 4 shows the results of another attempt to identify the features and classifier. In this case, Fine KNN is found to be more suitable as it gives an accuracy of 88% with the training time of 0.56 sec.

Table 2 shows the best classifiers in other test-runs where the number and type of features were varied. In some cases, PCA (Principal Component Analysis) was enabled to observe how it affects the behaviour of classifiers. But, it was finally decided to not use PCA because though it improved the accuracy in some cases, it was difficult to pinpoint which components it used to the same. It was also noted that PCA worked best when the number of components to be considered was pre-determined. More variations were done by using different sub-types of KNN and LDA. The result was that the accuracy increases with the growing number of features but they have to be carefully selected. But after one point, it saturates and starts to decrease.

Table 2. Iteration results

Classifier	No. of features	Accuracy (in %)	PCA enabled
Fine KNN	5	72	No
Fine KNN	55	77	Yes
Weighted KNN	15	83	Yes
Weighted KNN	46	48	No
Weighted KNN	6	73	Yes with 15 components specified
Subspace KNN	6	74	No

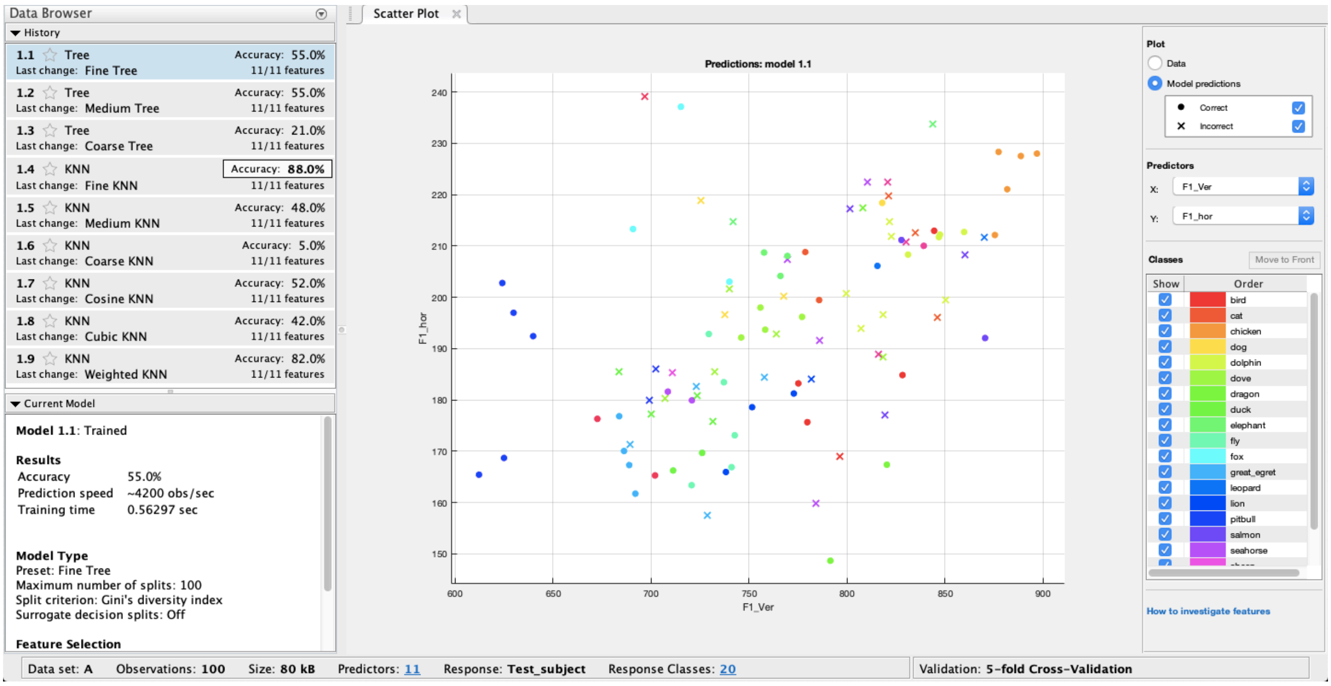

Figure 4. Scatterplot 2

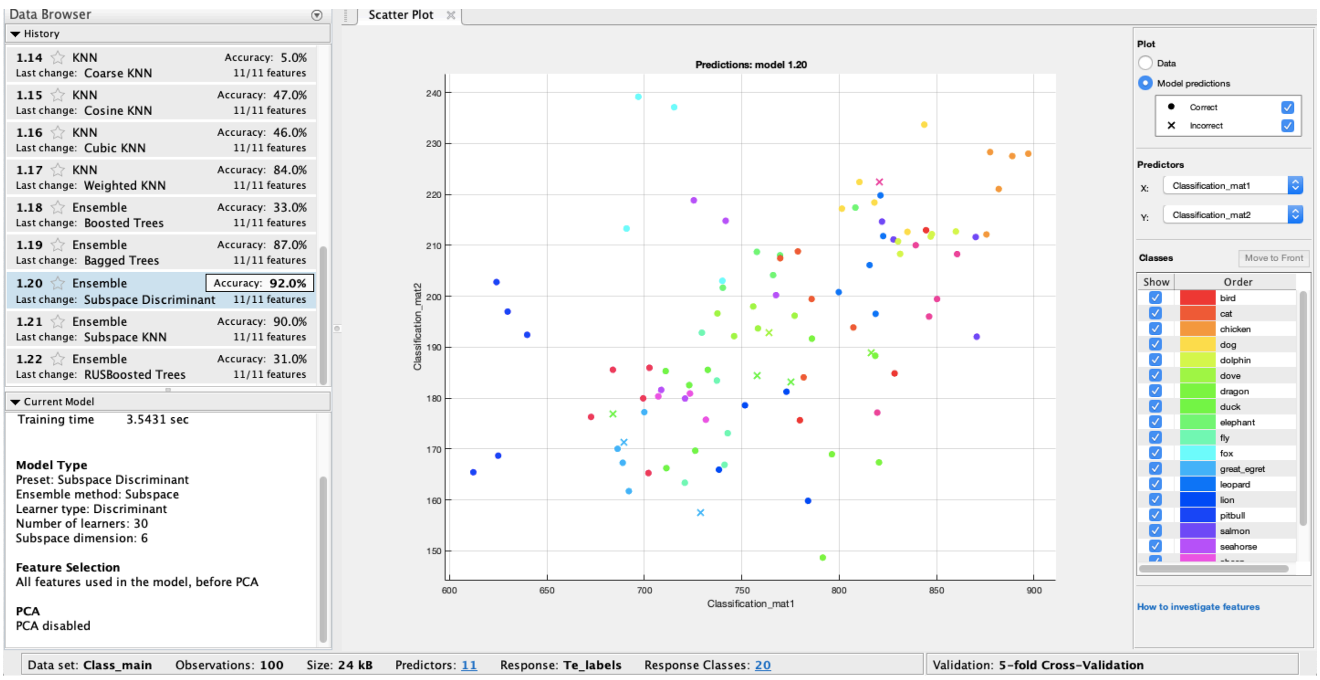

Figure 5. Scatterplot 3

Just by observing the three scatterplots, it can be inferred that the number of incorrect points decreases as accuracy increases with scatterplot 3 having the least. Also, the incorrect points belong to four or five classes. In other cases, there are wrong predictions in more classes
During another iteration, I did a test run with the chosen features using every classifier. Figure 6 shows the results of it. It proves that Ensemble classifier which uses subspace discriminant analysis has the highest accuracy of 92%. However, it is a trade-off between accuracy and time because this takes more time(3.54 sec) than Fine KNN and Subspace KNN classifiers.

Figure 6. All classifiers test-run result

The preliminary analysis just goes on to fortify the statement that KNN and LDA are the most suitable classifiers for this application. Out of them, Ensemble classifier which is basically Subspace discriminant should be the final choice of classifier because there are no time restraints in our case. We can afford to have better accuracy at the cost of time. Another advantage is that it uses less memory since it is more computationally intensive.

5. Methodology

The majority of the assignment was done using Matlab with Excel used in the initial stages. The exact process followed is described in this section.

The preliminary analysis was done using tables created in MS Excel and imported into Matlab. The first thing was to import the data from given text files and form required tables using Matlab.

Once the classifier was finalized, the trained model was exported into my Matlab script. A function was written for feature extraction. It used the Euclidean distance formula to find the distances in a 2-D plane between different parts of the hand. The required eleven features are computed using this function. It gives us the length and width of little finger, ring finger, middle finger, index finger and thumb. The last feature used is the palm width.

Then another matrix is created which contains the class names and features. This is given as input to the classifier. The classifier is trained on the training data provided using Classification learner application in Matlab. The model is then exported and can be used to predict the classes of new data.

The classes are predicted and stored in a cell array. There is yet another table created which consists of the new class array and original co-ordinates. This is given as text output. The same program is run twice by switching inputs and outputs from test to train. The text files are stored as the final files.

Once the basic program was up and running, I worked to optimize it. A new function to do the feature extraction and another function which is the ensemble subspace discriminant classifier were included in the main code.

Editor - /Users/shreya/Documents/UofT/Winter'19/Seminar/Assignment2/feature_extraction.m

att4.m | classifier.m | feature_extraction.m | ensemble.m | untitled6

```
function [lf_Ver,lf_hor,rf_Ver,rf_hor,mf_Ver,mf_hor,if_Ver,if_hor,thumb_Ver,thumb_hor,palm_width] = feature_extraction(Test_numbers)
% Feature extraction is done by using Euclidean distances between
% co-ordinates
little_finger_x = (Test_numbers(1:100,1) +Test_numbers(1:100,11))/2;     % get the midpoint of x-coordinate of little finger
little_finger_y = (Test_numbers(1:100,2) +Test_numbers(1:100,12))/2;     % get the midpoint of y-coordinate of little finger
lf_Ver= sqrt(((little_finger_x-Test_numbers(1:100,7)).^2) +((little_finger_y-Test_numbers(1:100,8)).^2));   %length
lf_hor= sqrt(((Test_numbers(1:100,5)-Test_numbers(1:100,9)).^2) +((Test_numbers(1:100,6)-Test_numbers(1:100,10)).^2)); %width

ring_finger_x = (Test_numbers(1:100,11) +Test_numbers(1:100,19))/2;  % get the midpoint of x-coordinate of ring finger
ring_finger_y = (Test_numbers(1:100,12) +Test_numbers(1:100,20))/2;  % get the midpoint of y-coordinate of ring finger
rf_Ver= sqrt(((ring_finger_x-Test_numbers(1:100,15)).^2) +((ring_finger_y-Test_numbers(1:100,16)).^2));
rf_hor= sqrt(((Test_numbers(1:100,13)-Test_numbers(1:100,17)).^2) +((Test_numbers(1:100,14)-Test_numbers(1:100,18)).^2));

middle_finger_x = (Test_numbers(1:100,19) +Test_numbers(1:100,27))/2;   % get the midpoint of x-coordinate of middle finger
middle_finger_y = (Test_numbers(1:100,20) +Test_numbers(1:100,28))/2;   % get the midpoint of y-coordinate of middle finger
mf_Ver= sqrt(((middle_finger_x-Test_numbers(1:100,23)).^2) +((middle_finger_y-Test_numbers(1:100,24)).^2));
mf_hor= sqrt(((Test_numbers(1:100,21)-Test_numbers(1:100,25)).^2) +((Test_numbers(1:100,22)-Test_numbers(1:100,26)).^2));

index_finger_x = (Test_numbers(1:100,27) +Test_numbers(1:100,35))/2;    % get the midpoint of x-coordinate of index finger
index_finger_y = (Test_numbers(1:100,28) +Test_numbers(1:100,36))/2;    % get the midpoint of y-coordinate of index finger
if_Ver= sqrt(((index_finger_x-Test_numbers(1:100,31)).^2) +((index_finger_y-Test_numbers(1:100,32)).^2));
if_hor= sqrt(((Test_numbers(1:100,29)-Test_numbers(1:100,33)).^2) +((Test_numbers(1:100,30)-Test_numbers(1:100,34)).^2));

thumb_x = (Test_numbers(1:100,37) +Test_numbers(1:100,45))/2;           % get the midpoint of x-coordinate of thumb
thumb_y = (Test_numbers(1:100,38) +Test_numbers(1:100,46))/2;           % get the midpoint of y-coordinate of thumb
thumb_Ver= sqrt(((thumb_x-Test_numbers(1:100,41)).^2) +((thumb_y-Test_numbers(1:100,42)).^2));
thumb_hor= sqrt(((Test_numbers(1:100,39)-Test_numbers(1:100,43)).^2) +((Test_numbers(1:100,40)-Test_numbers(1:100,44)).^2));

palm_width= sqrt(((Test_numbers(1:100,1)-Test_numbers(1:100,35)).^2) +((Test_numbers(1:100,2)-Test_numbers(1:100,36)).^2));

end
```

Figure 7. Function for feature extraction

Then, the classifier is trained using the given dataset and predicts the classes from the test data set in the same programme. This is made possible due to the ensemble function.

The confusion matrices and ROC give a good idea about the variation in between true class and predicted class. The scatterplots, confusion matrices and ROC curves were obtained through the Classification learner application. All these are saved along with relevant information about different iterations. The results of the best model are considered as final. In this case, it's the Subspace Discriminate ensemble classifier.

6. Results

i. Confusion Matrix

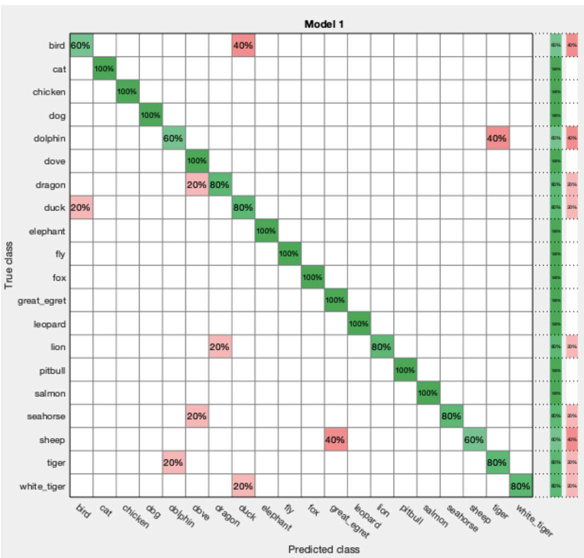

Figure 8. Fine KNN confusion matrix.

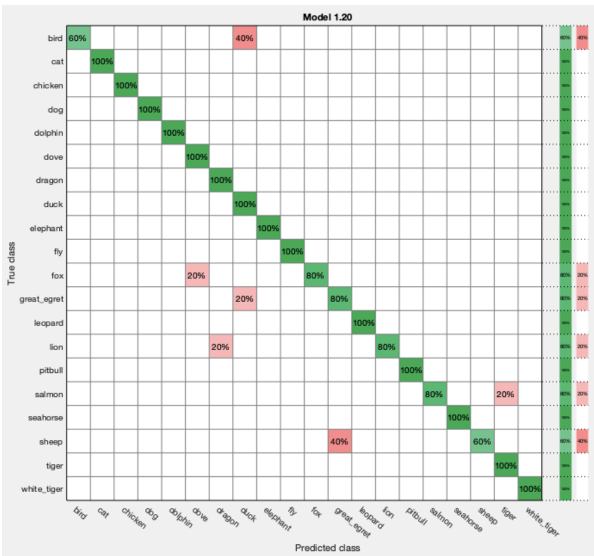

Figure 9. Ensemble confusion matrix

Based on these two figures, Ensemble classifier has fewer False Positives. Overall, it can be seen that only five classes are affected in case of ensemble classifier. On the other hand, seven classes have inaccurate predictions in the case of KNN classifier.

ii. ROC Curves

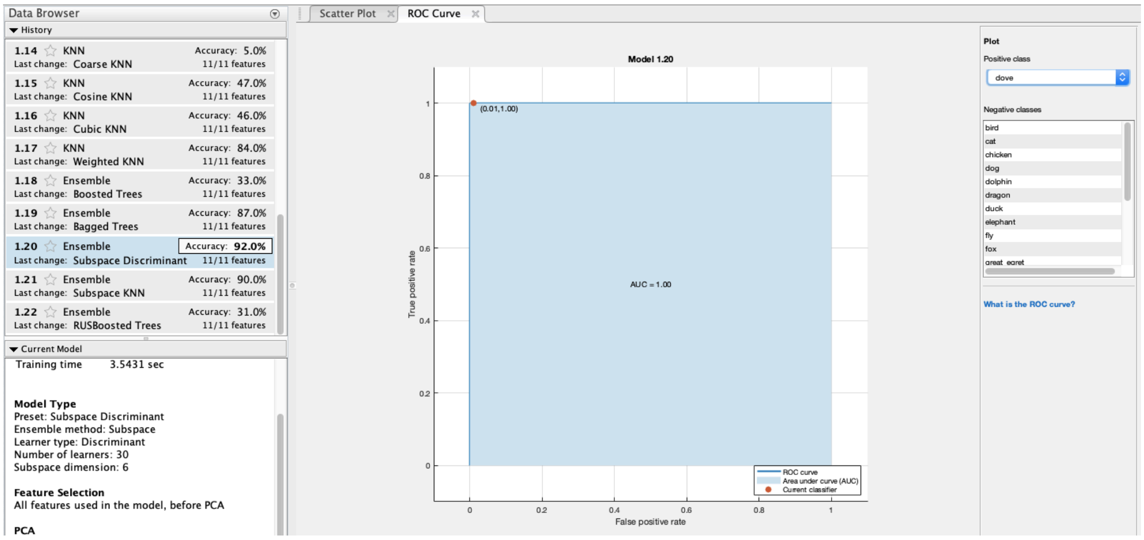

Figure 10. ROC curve for class dove

With Ensemble Subspace Discriminate classifier, most of the classes have similar ROC curves as shown above in figure 9. While there are some exceptional cases like great egret, sheep, lion, tiger, etc. The ROC curves for such cases can be illustrated by a couple of examples shown below in Figure 10. The curve in figure 9 is a perfect square with zero false positives. The exceptional curves have a few abnormal lines.

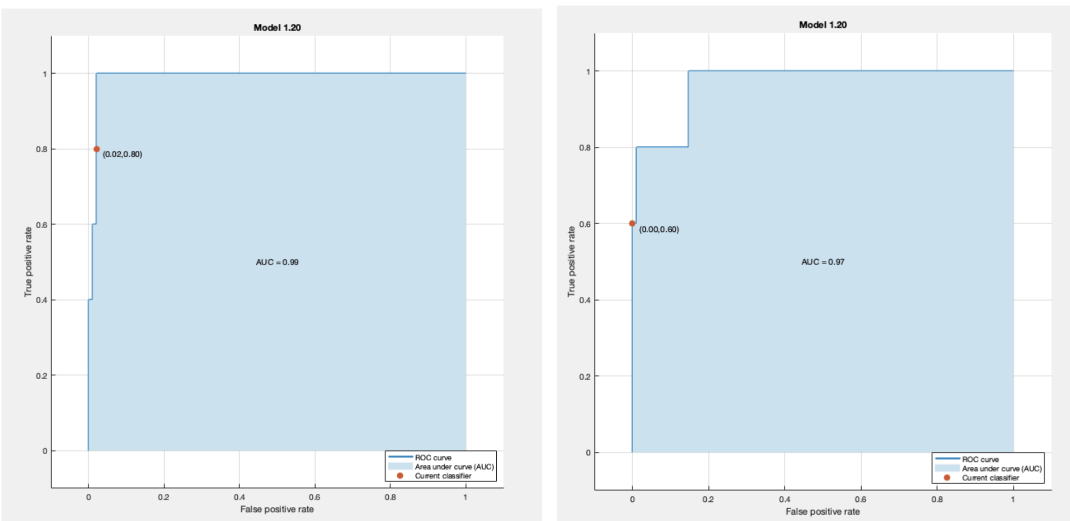

Figure 11. Exceptional ROC curves

The curve at the right side shows one of the worst results for this classifier. In this case, the True positive rate is only 60%.

iii. Classification results

The final code uses a single classifier function for the whole process of identifying the handprints of different individuals. This satisfies the expected outcome of the assignment. The screenshot of the final code is shown in figure 12.

```
/ ▸ Users ▸ shreya ▸ Documents ▸ UofT ▸ Winter'19 ▸ Seminar ▸ Assignment2 ▸ Final
Editor – /Users/shreya/Documents/UofT/Winter'19/Seminar/Assignment2/Final/Classifier.m
Classifier.m

% Imports both the text files as input by giving file pathway
Test_data = importdata('Assignment2-handtrainfile.txt');
Test_data2 = importdata('Assignment2-handtestfile.txt');

%Prediction of classes
[out,out2] = Classifier_main(Test_data,Test_data2); % Calls classifier function

% Write the outputs into two different text files
writetable(out,'Trainfile_output.txt','WriteVariableName',false,'Delimiter',' ')% Print the answer table in a text file
writetable(out2,'Testfile_output.txt','WriteVariableName',false,'Delimiter',' ')


function [out,out2] = Classifier_main(Test_data,Test_data2)...
```

Figure 12. Final Matlab code

This code gives two text files as output. It also stores the .csv files that it creates to do pre-processing. Though the classifier is slower than others, the accuracy is higher. The assumption is that it correctly predicts the classes of the test set. This method is more effective. After multiple runs, the final predicted classes were cross-examined. The result is the same while using this classifier.

```
Command Window
New to MATLAB? See resources for Getting Started.

trainedClassifier =

  struct with fields:

                predictFcn: @(x)ensemblePredictFcn(predictorExtractionFcn(x))
         RequiredVariables: {1×11 cell}
    ClassificationEnsemble: [1×1 classreg.learning.classif.ClassificationEnsemble]
                     About: 'This struct is a trained model exported from Classification Learner R2018a.'
               HowToPredict: 'To make predictions on a new table, T, use: ↵  yfit = c.predictFcn(T) ↵replacing 'c' with the name of the variable that is this struct, e.g. 'traine


validationAccuracy =

    0.9300

fx >> |
```

Figure 13. Validation accuracy result

The above figure is a screenshot of the command window of Matlab after running the final code. The validation accuracy is above 90%.

```
fly 730 1713 727 1606 639 1366 580 915 806 1292 917 1518 902 1151 953 558 1089 1107
1122 1373 1132 1011 1221 429 1350 1021 1355 1429 1451 1105 1674 624 1654 1130 1568
1454 1550 1885 1781 1655 2080 1444 1953 1771 1760 2085
fox 619 2037 634 1861 599 1647 626 1175 805 1600 875 1771 912 1419 1073 865 1145 1384
1170 1600 1235 1329 1520 798 1498 1344 1470 1669 1590 1436 1927 1078 1801 1555 1679
1818 1674 2223 1994 2183 2367 2111 2106 2407 1806 2583
great_egret 837 1801 800 1678 697 1486 602 1107 827 1418 998 1613 964 1220 1001 687
1162 1182 1223 1438 1247 1063 1316 543 1462 1076 1483 1435 1603 1141 1790 680 1814
1196 1732 1497 1794 1910 2019 1705 2306 1582 2190 1842 2043 2074
pitbull 527 1683 547 1546 531 1342 547 978 723 1298 759 1518 835 1154 979 654 1039
1146 1015 1410 1135 1074 1384 610 1368 1122 1280 1482 1500 1210 1808 894 1704 1342
1532 1602 1508 2011 1808 1927 2132 1903 1924 2143 1700 2279
leopard 735 1975 687 1825 537 1643 359 1186 708 1527 909 1736 796 1312 725 687 1015
1230 1165 1561 1189 1134 1322 506 1442 1141 1462 1538 1606 1213 1866 690 1852 1281
1756 1613 1794 1961 2033 1736 2347 1459 2272 1842 2122 2142
elephant 599 2207 551 2063 439 1887 303 1414 607 1771 795 1935 767 1562 779 962 987
1466 1059 1755 1067 1374 1127 782 1328 1330 1360 1711 1440 1362 1616 878 1692 1406
1656 1707 1736 2147 1988 1959 2356 1807 2204 2127 2028 2359
chicken 274 1660 346 1480 382 1180 574 706 610 1168 604 1486 790 1084 1078 508 1024
1156 952 1462 1198 1114 1642 652 1456 1258 1210 1678 1516 1420 1966 1048 1726 1600
1432 1900 1288 2320 1570 2260 2002 2170 1744 2488 1396 2770
great_egret 649 1862 615 1722 506 1541 410 1162 653 1456 834 1640 789 1254 813 742
984 1203 1063 1497 1087 1134 1169 608 1305 1145 1322 1514 1503 1227 1753 803 1695
1312 1568 1596 1599 2026 1907 1947 2279 1913 2074 2111 1876 2272
tiger 646 1978 562 1756 352 1492 226 1120 538 1420 832 1618 718 1240 658 664 946 1150
1084 1468 1114 1024 1174 460 1336 1000 1396 1450 1594 1102 1846 634 1816 1186 1654
1486 1654 1888 1954 1768 2368 1630 2146 1954 1780 2326
fly 730 1958 730 1850 649 1604 613 1135 823 1541 907 1736 910 1363 1003 783 1112 1333
1125 1589 1153 1237 1277 664 1376 1247 1358 1655 1474 1328 1730 872 1690 1394 1583
1723 1568 2100 1793 1872 2092 1672 1973 1994 1763 2316
```

Figure 14. Screenshot of Test output

The predicted class along with the given co-ordinates are stored in two text files, one with training data and one with the testing data. The screenshots shown in figure 14 and figure 15 show the partial sections from the two outputs.

```
bird 821 2050 809 1856 652 1523 573 1142 858 1469 1076 1747 1124 1366 1221 863 1330
1384 1366 1681 1493 1360 1742 906 1675 1457 1578 1826 1833 1590 2226 1293 1996 1717
1808 1977 1681 2347 1972 2244 2341 2268 2002 2492 1645 2704
bird 1047 2035 947 1951 743 1827 403 1490 819 1659 1043 1751 871 1474 527 946 955
1290 1099 1430 971 1182 723 638 1143 1078 1332 1358 1243 1026 1135 494 1432 982 1552
1250 1776 1498 1800 1218 1928 854 2000 1150 2036 1406
bird 616 1798 610 1606 538 1342 472 886 718 1300 886 1534 1000 1192 1186 694 1192
1210 1168 1504 1366 1162 1654 766 1528 1294 1366 1666 1666 1432 2062 1162 1774 1624
1552 1876 1432 2176 1840 2212 2122 2362 1726 2446 1408 2566
bird 895 1827 811 1719 623 1582 311 1242 731 1434 1011 1526 907 1202 743 658 1091
1106 1215 1338 1191 1022 1167 418 1404 990 1480 1358 1596 1026 1804 538 1804 1046
1752 1370 1820 1735 2064 1546 2396 1366 2252 1679 2084 1895
bird 472 1576 556 1390 586 1114 754 730 760 1138 802 1426 1036 1156 1444 760 1240
1186 1084 1486 1354 1252 1798 952 1492 1384 1216 1678 1570 1552 2044 1438 1660 1756
1318 1948 1132 2170 1450 2140 1762 2242 1420 2404 1090 2500
cat 634 2032 604 1864 508 1588 460 1144 700 1534 826 1732 814 1366 868 736 1018 1276
1072 1576 1120 1168 1192 574 1360 1168 1384 1594 1510 1270 1720 748 1762 1270 1660
1666 1720 2128 1972 1912 2356 1798 2158 2158 1912 2494
cat 555 1943 523 1799 447 1598 367 1126 643 1530 759 1735 723 1366 759 730 935 1302
1015 1578 1031 1218 1095 562 1288 1218 1304 1590 1464 1266 1744 770 1724 1310 1596
1651 1704 2167 1920 1951 2276 1807 2112 2159 1920 2371
cat 532 2026 514 1810 436 1540 394 1126 610 1480 736 1714 748 1312 808 748 958 1264
1018 1606 1072 1174 1192 616 1324 1222 1306 1636 1486 1282 1726 826 1714 1378 1582
1756 1636 2236 1900 2008 2266 1936 2026 2314 1780 2578
cat 568 2038 538 1828 436 1552 382 1120 616 1480 778 1708 772 1318 868 718 964 1276
1030 1576 1096 1132 1198 574 1342 1132 1330 1618 1492 1276 1750 796 1726 1414 1618
1696 1618 2146 1912 1948 2302 1876 1984 2278 1714 2614
cat 635 1959 607 1819 511 1582 443 1138 711 1522 859 1751 843 1422 883 750 1055 1366
1119 1614 1143 1266 1215 614 1400 1274 1420 1655 1556 1342 1812 830 1792 1394 1684
```

Figure 15. Screenshot of Train output

The train output is exactly the same as the original data used for training. This shows that the classifier is functioning with sufficient accuracy.

7. References

[1] *Biometric Technology Application Manual*, Volume 1. National Biometric Security Project, 2008.

[2] D. Al-Fiky and Z. Ageed, "A New Features Extracted for Recognizing a Hand Geometry Using BPNN", *International Journal of Scientific & Engineering Research*, vol. 5, no. 8, pp. 232-236, 2014. Available: https://www.ijser.org/paper/A-New-Features-Extracted-for-Recognizing-a-Hand-Geometry-Using-BPNN.html. [Accessed 20 February 2019].

[3] M. Mayo, "The Practical Importance of Feature Selection", *Kdnuggets.com*, 2017. [Online]. Available: https://www.kdnuggets.com/2017/06/practical-importance-feature-selection.html. [Accessed: 20- Feb- 2019].

[4] C. Ravinath, P. Agrawal and P. Venkata, "Personal Authentication using hand geometry", *Www4.comp.polyu.edu.hk*. [Online]. Available: http://www4.comp.polyu.edu.hk/~csajaykr/myhome/teaching/biometrics/hg.pdf. [Accessed: 20- Feb- 2019].

[5] A. Bronshtein, "A Quick Introduction to K-Nearest Neighbors Algorithm", *Medium*, 2017. [Online]. Available: https://medium.com/@adi.bronshtein/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7. [Accessed: 20- Feb- 2019].

[6]"Supervised Learning Workflow and Algorithms- MATLAB & Simulink", *Mathworks.com*, 2018. [Online]. Available: https://www.mathworks.com/help/stats/supervised-learning-machine-learning-workflow-and-algorithms.html#bswluh9. [Accessed: 20- Feb- 2019].

[7]J. Brownlee, "Linear Discriminant Analysis for Machine Learning", *Machine Learning Mastery*, 2016. [Online]. Available: https://machinelearningmastery.com/linear-discriminant-analysis-for-machine-learning/. [Accessed: 20- Feb- 2019].