REPORT

Investigation of some Machine Learning methods for Type II Diabetes Mellitus screening using Raman Spectroscopy

Le Anh Duc (G0.HN)

International School | Vietnam National University

Table of Contents

Introduction

Type 2 diabetes makes up about 90% of cases of diabetes, with the other 10% due primarily to type 1 diabetes and gestational diabetes.[1] In type 1 diabetes there is a lower total level of insulin to control blood glucose, due to an autoimmune induced loss of insulin-producing beta cells in the pancreas.[12][13] Diagnosis of diabetes is by blood tests such as fasting plasma glucose, oral glucose tolerance test, or glycated hemoglobin (A1C).[3]

Recent study

Use of Raman Spectroscopy

There are numerous study about the use of Raman spectroscopy in various area such as plastic classification, fruit classification, wine classification, etc.

Machine Learning as a tool for non-invasive diabetes screening

Dataset

The data set is given on Kaggle by the author Edgar Guevana, for the pupose of improving their former result. There are total of 20 subjects corresponding to 20 observation of Raman at range 0 to 3159 cm-1. Each subject consists of 4 distinct measurement at their ear lobe, inner arm, thumbnail and cubital vein.

Below is the given dataset in spreadsheet form:

1. Ear lobe:

```
[9]: earlobe_df
```

[9]:

	patientID	has_DM2	Var2	Var3	Var4	Var5	Var6	Var7	Var8	Var9	...	Var3152	Var3153	Var3154	Var3155	Var3156	Var3157	Var3158	Var3159	V
0	ramanShift	NaN	0.000000	1.000000	2.000000	3.000000	4.000000	5.000000	6.000000	7.000000	...	3150	3151	3152	3153	3154	3155	3156	3157	
1	DM201	1.0	181.800000	181.800000	181.800000	181.800000	181.800000	181.800000	181.800000	181.800000	...	0	0	0	0	0	0	0	0	
2	DM202	1.0	162.800000	162.800000	162.800000	162.800000	162.800000	162.800000	162.800000	162.800000	...	0	0	0	0	0	0	0	0	
3	DM203	1.0	107.400000	107.400000	107.400000	107.400000	107.400000	107.400000	107.400000	107.400000	...	0	0	0	0	0	0	0	0	
4	DM204	1.0	290.166667	290.166667	290.166667	290.166667	290.166667	290.166667	290.166667	290.166667	...	0	0	0	0	0	0	0	0	
5	DM205	1.0	50.600000	50.600000	50.600000	50.600000	50.600000	50.600000	50.600000	50.600000	...	0	0	0	0	0	0	0	0	
6	DM206	1.0	63.800000	63.800000	63.800000	63.800000	63.800000	63.800000	63.800000	63.800000	...	0	0	0	0	0	0	0	0	
7	DM207	1.0	147.800000	147.800000	147.800000	147.800000	147.800000	147.800000	147.800000	147.800000	...	0	0	0	0	0	0	0	0	
8	DM208	1.0	55.833333	55.833333	55.833333	55.833333	55.833333	55.833333	55.833333	55.833333	...	0	0	0	0	0	0	0	0	
9	DM209	1.0	136.200000	136.200000	136.200000	136.200000	136.200000	136.200000	136.200000	136.200000	...	0	0	0	0	0	0	0	0	
10	DM210	1.0	134.000000	134.000000	134.000000	134.000000	134.000000	134.000000	134.000000	134.000000	...	0	0	0	0	0	0	0	0	
11	DM211	1.0	105.800000	105.800000	105.800000	105.800000	105.800000	105.800000	105.800000	105.800000	...	0	0	0	0	0	0	0	0	
12	Ctrl01	0.0	116.000000	116.000000	116.000000	116.000000	116.000000	116.000000	116.000000	116.000000	...	0	0	0	0	0	0	0	0	
13	Ctrl02	0.0	115.600000	115.600000	115.600000	115.600000	115.600000	115.600000	115.600000	115.600000	...	0	0	0	0	0	0	0	0	
14	Ctrl03	0.0	80.600000	80.600000	80.600000	80.600000	80.600000	80.600000	80.600000	80.600000	...	0	0	0	0	0	0	0	0	
15	Ctrl04	0.0	64.000000	64.000000	64.000000	64.000000	64.000000	64.000000	64.000000	64.000000	...	0	0	0	0	0	0	0	0	
16	Ctrl05	0.0	34.833333	34.833333	34.833333	34.833333	34.833333	34.833333	34.833333	34.833333	...	0	0	0	0	0	0	0	0	
17	Ctrl06	0.0	43.000000	43.000000	43.000000	43.000000	43.000000	43.000000	43.000000	43.000000	...	0	0	0	0	0	0	0	0	
18	Ctrl07	0.0	61.750000	61.750000	61.750000	61.750000	61.750000	61.750000	61.750000	61.750000	...	0	0	0	0	0	0	0	0	
19	Ctrl08	0.0	228.666667	228.666667	228.666667	228.666667	228.666667	228.666667	228.666667	228.666667	...	0	0	0	0	0	0	0	0	
20	Ctrl09	0.0	256.600000	256.600000	256.600000	256.600000	256.600000	256.600000	256.600000	256.600000	...	0	0	0	0	0	0	0	0	

21 rows × 3162 columns

2. Inner Arm:

```
[10]: arm_df
```

[10]:

	patientID	has_DM2	Var2	Var3	Var4	Var5	Var6	Var7	Var8	Var9	...	Var3152	Var3153	Var3154	Var3155	Var3156	Var3157	Var3158	Var3159	V
0	ramanShift	NaN	0.000000	1.000000	2.000000	3.000000	4.000000	5.000000	6.000000	7.000000	...	3150	3151	3152	3153	3154	3155	3156	3157	
1	DM201	1.0	326.000000	326.000000	326.000000	326.000000	326.000000	326.000000	326.000000	326.000000	...	0	0	0	0	0	0	0	0	
2	DM202	1.0	214.800000	214.800000	214.800000	214.800000	214.800000	214.800000	214.800000	214.800000	...	0	0	0	0	0	0	0	0	
3	DM203	1.0	457.800000	457.800000	457.800000	457.800000	457.800000	457.800000	457.800000	457.800000	...	0	0	0	0	0	0	0	0	
4	DM204	1.0	181.833333	181.833333	181.833333	181.833333	181.833333	181.833333	181.833333	181.833333	...	0	0	0	0	0	0	0	0	
5	DM205	1.0	179.800000	179.800000	179.800000	179.800000	179.800000	179.800000	179.800000	179.800000	...	0	0	0	0	0	0	0	0	
6	DM206	1.0	237.400000	237.400000	237.400000	237.400000	237.400000	237.400000	237.400000	237.400000	...	0	0	0	0	0	0	0	0	
7	DM207	1.0	217.333333	217.333333	217.333333	217.333333	217.333333	217.333333	217.333333	217.333333	...	0	0	0	0	0	0	0	0	
8	DM208	1.0	89.166667	89.166667	89.166667	89.166667	89.166667	89.166667	89.166667	89.166667	...	0	0	0	0	0	0	0	0	
9	DM209	1.0	336.600000	336.600000	336.600000	336.600000	336.600000	336.600000	336.600000	336.600000	...	0	0	0	0	0	0	0	0	
10	DM210	1.0	89.600000	89.600000	89.600000	89.600000	89.600000	89.600000	89.600000	89.600000	...	0	0	0	0	0	0	0	0	
11	DM211	1.0	320.400000	320.400000	320.400000	320.400000	320.400000	320.400000	320.400000	320.400000	...	0	0	0	0	0	0	0	0	
12	Ctrl01	0.0	230.800000	230.800000	230.800000	230.800000	230.800000	230.800000	230.800000	230.800000	...	0	0	0	0	0	0	0	0	
13	Ctrl02	0.0	159.600000	159.600000	159.600000	159.600000	159.600000	159.600000	159.600000	159.600000	...	0	0	0	0	0	0	0	0	
14	Ctrl03	0.0	327.000000	327.000000	327.000000	327.000000	327.000000	327.000000	327.000000	327.000000	...	0	0	0	0	0	0	0	0	
15	Ctrl04	0.0	94.400000	94.400000	94.400000	94.400000	94.400000	94.400000	94.400000	94.400000	...	0	0	0	0	0	0	0	0	
16	Ctrl05	0.0	71.833333	71.833333	71.833333	71.833333	71.833333	71.833333	71.833333	71.833333	...	0	0	0	0	0	0	0	0	
17	Ctrl06	0.0	325.600000	325.600000	325.600000	325.600000	325.600000	325.600000	325.600000	325.600000	...	0	0	0	0	0	0	0	0	
18	Ctrl07	0.0	116.000000	116.000000	116.000000	116.000000	116.000000	116.000000	116.000000	116.000000	...	0	0	0	0	0	0	0	0	
19	Ctrl08	0.0	272.500000	272.500000	272.500000	272.500000	272.500000	272.500000	272.500000	272.500000	...	0	0	0	0	0	0	0	0	
20	Ctrl09	0.0	156.166667	156.166667	156.166667	156.166667	156.166667	156.166667	156.166667	156.166667	...	0	0	0	0	0	0	0	0	

21 rows × 3162 columns

3. Thumb Nail:

```
[11]: thumbnail_df
```

[11]:

	patientID	has_DM2	Var2	Var3	Var4	Var5	Var6	Var7	Var8	Var9	...	Var3152	Var3153	Var3154	Var3155	Var3156	Var3157	Var3158	Var3159	Var3160	Var3161
0	ramanShift	NaN	0.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0	...	3150	3151	3152	3153	3154	3155	3156	3157	3158	3159
1	DM201	1.0	170.0	170.0	170.0	170.0	170.0	170.0	170.0	170.0	...	0	0	0	0	0	0	0	0	0	0
2	DM202	1.0	116.4	116.4	116.4	116.4	116.4	116.4	116.4	116.4	...	0	0	0	0	0	0	0	0	0	0
3	DM203	1.0	104.8	104.8	104.8	104.8	104.8	104.8	104.8	104.8	...	0	0	0	0	0	0	0	0	0	0
4	DM204	1.0	72.6	72.6	72.6	72.6	72.6	72.6	72.6	72.6	...	0	0	0	0	0	0	0	0	0	0
5	DM205	1.0	90.2	90.2	90.2	90.2	90.2	90.2	90.2	90.2	...	0	0	0	0	0	0	0	0	0	0
6	DM206	1.0	34.4	34.4	34.4	34.4	34.4	34.4	34.4	34.4	...	0	0	0	0	0	0	0	0	0	0
7	DM207	1.0	72.0	72.0	72.0	72.0	72.0	72.0	72.0	72.0	...	0	0	0	0	0	0	0	0	0	0
8	DM208	1.0	49.0	49.0	49.0	49.0	49.0	49.0	49.0	49.0	...	0	0	0	0	0	0	0	0	0	0
9	DM209	1.0	73.2	73.2	73.2	73.2	73.2	73.2	73.2	73.2	...	0	0	0	0	0	0	0	0	0	0
10	DM210	1.0	114.0	114.0	114.0	114.0	114.0	114.0	114.0	114.0	...	0	0	0	0	0	0	0	0	0	0
11	DM211	1.0	111.6	111.6	111.6	111.6	111.6	111.6	111.6	111.6	...	0	0	0	0	0	0	0	0	0	0
12	Ctrl01	0.0	179.0	179.0	179.0	179.0	179.0	179.0	179.0	179.0	...	0	0	0	0	0	0	0	0	0	0
13	Ctrl02	0.0	106.4	106.4	106.4	106.4	106.4	106.4	106.4	106.4	...	0	0	0	0	0	0	0	0	0	0
14	Ctrl03	0.0	99.0	99.0	99.0	99.0	99.0	99.0	99.0	99.0	...	0	0	0	0	0	0	0	0	0	0
15	Ctrl04	0.0	73.2	73.2	73.2	73.2	73.2	73.2	73.2	73.2	...	0	0	0	0	0	0	0	0	0	0
16	Ctrl05	0.0	6.6	6.6	6.6	6.6	6.6	6.6	6.6	6.6	...	0	0	0	0	0	0	0	0	0	0
17	Ctrl06	0.0	89.4	89.4	89.4	89.4	89.4	89.4	89.4	89.4	...	0	0	0	0	0	0	0	0	0	0
18	Ctrl07	0.0	96.5	96.5	96.5	96.5	96.5	96.5	96.5	96.5	...	0	0	0	0	0	0	0	0	0	0
19	Ctrl08	0.0	145.2	145.2	145.2	145.2	145.2	145.2	145.2	145.2	...	0	0	0	0	0	0	0	0	0	0
20	Ctrl09	0.0	143.6	143.6	143.6	143.6	143.6	143.6	143.6	143.6	...	0	0	0	0	0	0	0	0	0	0

21 rows × 3162 columns

4. Cubital Vein:

```
[12]: vein_df
```

	patientID	has_DM2	Var2	Var3	Var4	Var5	Var6	Var7	Var8	Var9	...	Var3152	Var3153	Var3154	Var3155	Var3156	Var3157	Var3158	Var3159	V
0	ramanShift	NaN	0.000000	1.000000	2.000000	3.000000	4.000000	5.000000	6.000000	7.000000	...	3150	3151	3152	3153	3154	3155	3156	3157	
1	DM201	1.0	181.800000	181.800000	181.800000	181.800000	181.800000	181.800000	181.800000	181.800000	...	0	0	0	0	0	0	0	0	
2	DM202	1.0	246.200000	246.200000	246.200000	246.200000	246.200000	246.200000	246.200000	246.200000	...	0	0	0	0	0	0	0	0	
3	DM203	1.0	164.600000	164.600000	164.600000	164.600000	164.600000	164.600000	164.600000	164.600000	...	0	0	0	0	0	0	0	0	
4	DM204	1.0	293.800000	293.800000	293.800000	293.800000	293.800000	293.800000	293.800000	293.800000	...	0	0	0	0	0	0	0	0	
5	DM205	1.0	265.500000	265.500000	265.500000	265.500000	265.500000	265.500000	265.500000	265.500000	...	0	0	0	0	0	0	0	0	
6	DM206	1.0	132.400000	132.400000	132.400000	132.400000	132.400000	132.400000	132.400000	132.400000	...	0	0	0	0	0	0	0	0	
7	DM207	1.0	299.666667	299.666667	299.666667	299.666667	299.666667	299.666667	299.666667	299.666667	...	0	0	0	0	0	0	0	0	
8	DM208	1.0	136.166667	136.166667	136.166667	136.166667	136.166667	136.166667	136.166667	136.166667	...	0	0	0	0	0	0	0	0	
9	DM209	1.0	213.600000	213.600000	213.600000	213.600000	213.600000	213.600000	213.600000	213.600000	...	0	0	0	0	0	0	0	0	
10	DM210	1.0	158.800000	158.800000	158.800000	158.800000	158.800000	158.800000	158.800000	158.800000	...	0	0	0	0	0	0	0	0	
11	DM211	1.0	194.400000	194.400000	194.400000	194.400000	194.400000	194.400000	194.400000	194.400000	...	0	0	0	0	0	0	0	0	
12	Ctrl01	0.0	207.200000	207.200000	207.200000	207.200000	207.200000	207.200000	207.200000	207.200000	...	0	0	0	0	0	0	0	0	
13	Ctrl02	0.0	239.200000	239.200000	239.200000	239.200000	239.200000	239.200000	239.200000	239.200000	...	0	0	0	0	0	0	0	0	
14	Ctrl03	0.0	405.166667	405.166667	405.166667	405.166667	405.166667	405.166667	405.166667	405.166667	...	0	0	0	0	0	0	0	0	
15	Ctrl04	0.0	76.600000	76.600000	76.600000	76.600000	76.600000	76.600000	76.600000	76.600000	...	0	0	0	0	0	0	0	0	
16	Ctrl05	0.0	258.800000	258.800000	258.800000	258.800000	258.800000	258.800000	258.800000	258.800000	...	0	0	0	0	0	0	0	0	
17	Ctrl06	0.0	96.000000	96.000000	96.000000	96.000000	96.000000	96.000000	96.000000	96.000000	...	0	0	0	0	0	0	0	0	
18	Ctrl07	0.0	266.200000	266.200000	266.200000	266.200000	266.200000	266.200000	266.200000	266.200000	...	0	0	0	0	0	0	0	0	
19	Ctrl08	0.0	568.142857	568.142857	568.142857	568.142857	568.142857	568.142857	568.142857	568.142857	...	0	0	0	0	0	0	0	0	
20	Ctrl09	0.0	230.000000	230.000000	230.000000	230.000000	230.000000	230.000000	230.000000	230.000000	...	0	0	0	0	0	0	0	0	

21 rows × 3162 columns

5. AGEs:

```
[14]: ages_df
```

	AGEsID	Var802	Var803	Var804	Var805	Var806	Var807	Var808	Var809	Var810	...	Var1793	Var1794	Var1795	Var1796	Var1797	V
0	ramanShift	800.000000	801.000000	802.000000	803.000000	804.000000	805.000000	806.000000	807.000000	808.000000	...	1791.000000	1792.000000	1793.000000	1794.000000	1795.000000	1796.0
1	3-deoxyglucosone	0.001791	0.001813	0.001839	0.001868	0.001899	0.001929	0.001956	0.001980	0.002002	...	0.000492	0.000492	0.000495	0.000502	0.000509	0.0
2	glyoxal	0.000220	0.000243	0.000270	0.000299	0.000331	0.000367	0.000405	0.000445	0.000488	...	0.000213	0.000217	0.000222	0.000227	0.000233	0.0
3	GOLD	0.000302	0.000295	0.000288	0.000281	0.000275	0.000268	0.000262	0.000256	0.000251	...	0.000141	0.000142	0.000142	0.000129	0.000128	0.0
4	methylglyoxal	0.003002	0.003058	0.003095	0.003115	0.003121	0.003115	0.003098	0.003072	0.003038	...	-0.000115	-0.000130	-0.000146	-0.000161	-0.000176	-0.0
5	MG-H2	0.000236	0.000231	0.000226	0.000221	0.000215	0.000208	0.000202	0.000195	0.000187	...	0.000034	0.000049	0.000065	0.000082	0.000101	0.0
6	pentosidine	0.000803	0.000748	0.000693	0.000637	0.000582	0.000527	0.000475	0.000424	0.000377	...	0.000069	0.000067	0.000065	0.000063	0.000060	0.0

7 rows × 1002 columns

For the dataset from 1 to 4, these are the features:

- patientID
- has_DM2
- raman signal : 3160 columns

The feature patientID is not needed for data processing therefore it will be removed. Has_DM2 is the target column that has value 0 and 1 corresponding to Control and DM2 class. Hence, the method will be used for machine learning task is **Supervised learning**, the problem can be described as **Binary Classification.**

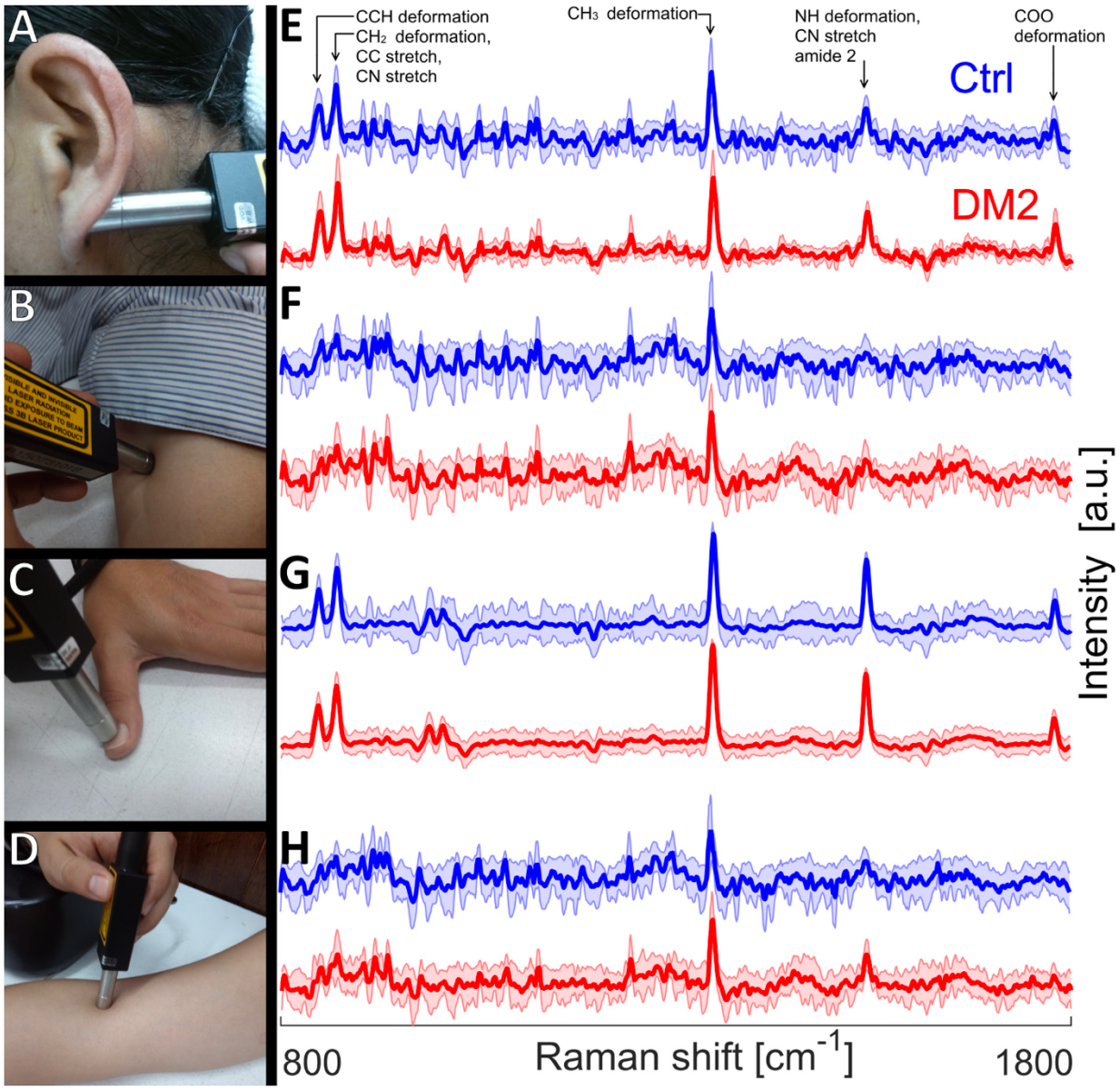
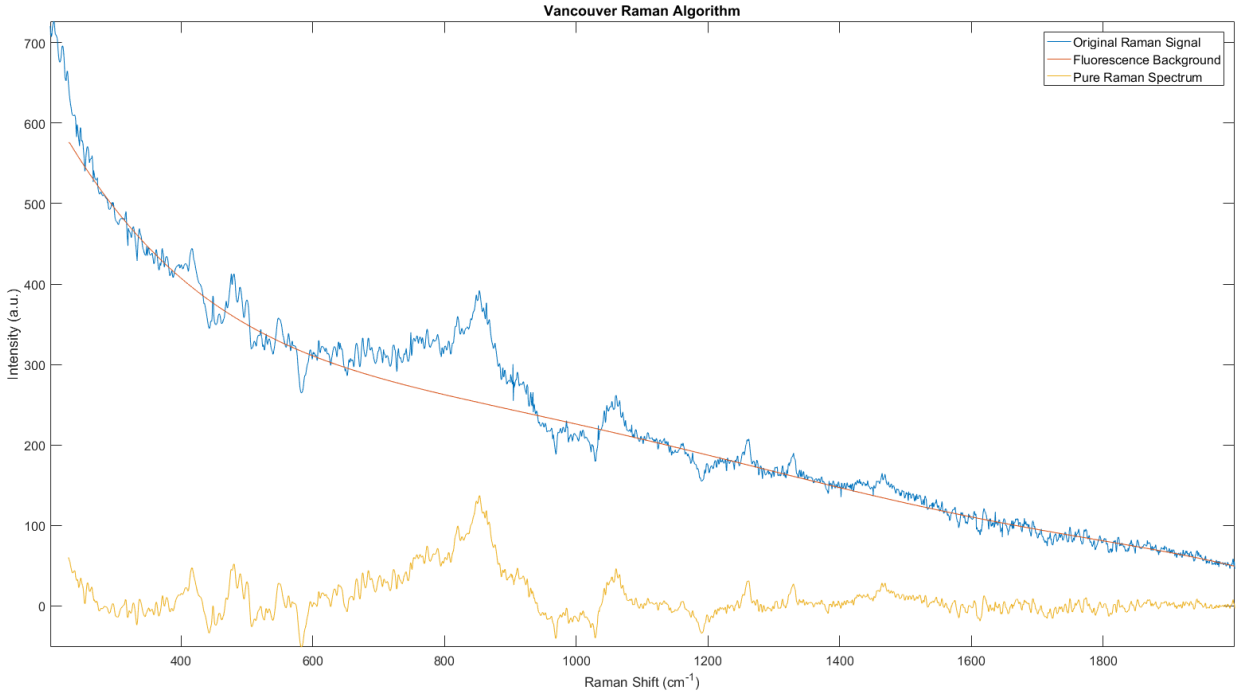


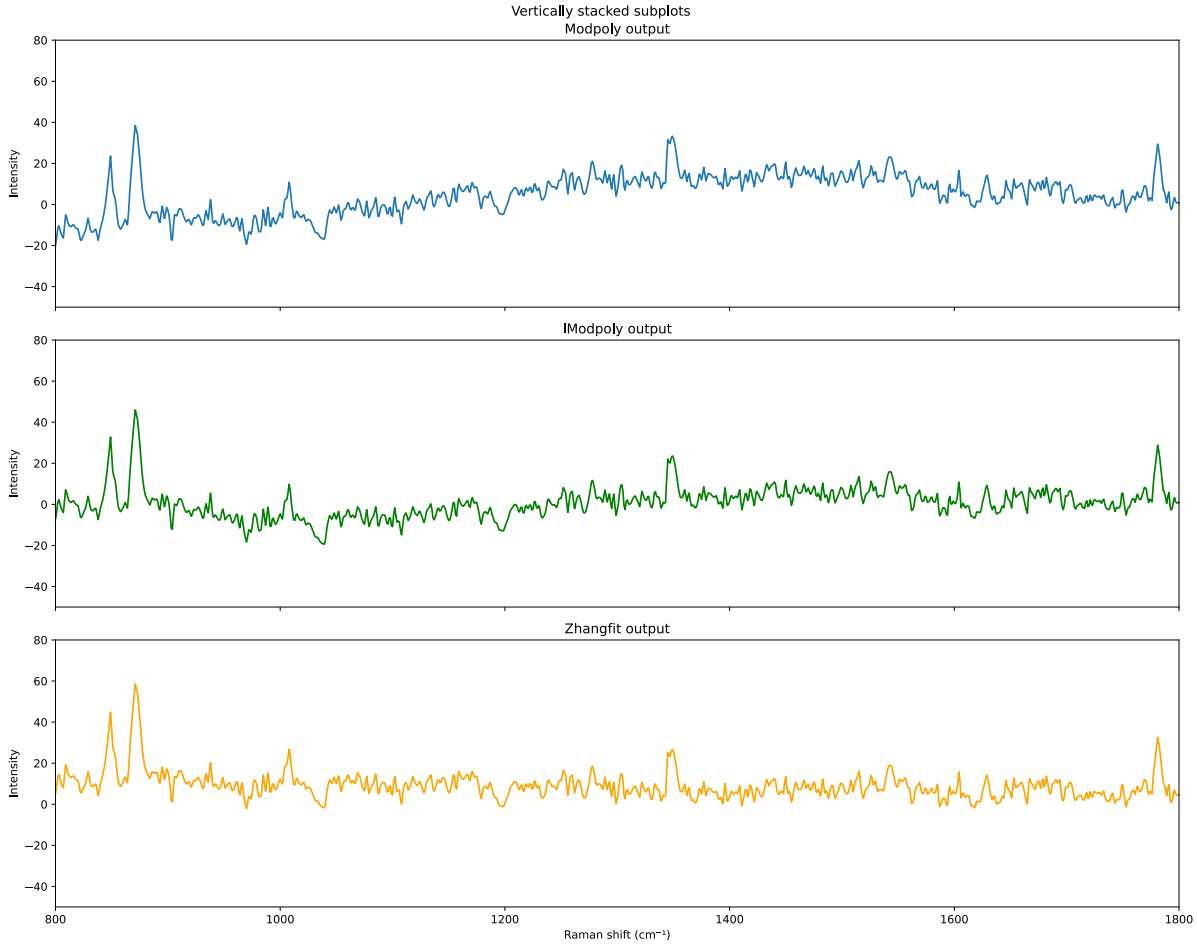
Figure 1. Images of skin sites for in vivo Raman spectra acquisition: (A) ear lobe, (B) inner arm (C) thumb nail (D) median cubital vein. Also shown at the right side are the corresponding Raman measurements (mean ± standard deviation) acquired at an excitation wavelength of 785nm (E-H).

Data Preprocessing

Several polynomial fitting algorithm were investigated with the purpose of removing background fluorescence in the original signal. However, to preserve the clarity of data, IModPoly method proposed by [] was used for the experiment. The Raman spectrum without fluorescence is computed by subtracting the adjusted polynomial from the original Raman

spectrum


Vancouver Raman Algorithm
Original Raman Signal
Fluorescence Background
Pure Raman Spectrum
Intensity (a.u.)
Raman Shift (cm-1)


Vertically stacked subplots
Modpoly output
IModpoly output
Zhangfit output
Intensity
Raman shift (cm⁻¹)

After receiving pure Raman signal, the data is then cropped to range 800-1800 cm-1

```
[23]: print(X)
      print('\n')
      print('The shape of ear lobe dataset is ', X.shape)
```

```
[[345.8314642  350.6934332  351.9772339  ... 107.9693807  113.2883358
  111.4543031 ]
 [232.1680176  235.3096008  240.2931427  ...  76.33356629  72.39400024
   72.14946137]
 [482.391803   487.6876221  490.1527954  ... 160.3856354  164.080011
  159.1892487 ]
 ...
 [133.321551   137.2859294  139.4813639  ...  40.94071897  46.81751378
   39.37948608]
 [301.1747081  306.9342702  309.3126323  ... 111.7359607  107.1366704
  105.9139811 ]
 [170.328982   174.9665553  179.3695526  ...  57.2306633   56.78333728
   54.74552091]]


The shape of ear lobe dataset is  (20, 1000)
```

```
print(X)
print('\n')
print('The shape of inner arm dataset is ', X.shape)
```

```
[[345.8314642  350.6934332  351.9772339  ... 107.9693807  113.2883358
  111.4543031 ]
 [232.1680176  235.3096008  240.2931427  ...  76.33356629  72.39400024
   72.14946137]
 [482.391803   487.6876221  490.1527954  ... 160.3856354  164.080011
  159.1892487 ]
 ...
 [133.321551   137.2859294  139.4813639  ...  40.94071897  46.81751378
   39.37948608]
 [301.1747081  306.9342702  309.3126323  ... 111.7359607  107.1366704
  105.9139811 ]
 [170.328982   174.9665553  179.3695526  ...  57.2306633   56.78333728
   54.74552091]]


The shape of inner arm dataset is  (20, 1000)
```

```
[24]: print(X)
      print('\n')
      print('The shape of thumb nail dataset is ', X.shape)
```

```
[[345.8314642  350.6934332  351.9772339  ... 107.9693807  113.2883358
  111.4543031 ]
 [232.1680176  235.3096008  240.2931427  ...  76.33356629  72.39400024
   72.14946137]
 [482.391803   487.6876221  490.1527954  ... 160.3856354  164.080011
  159.1892487 ]
 ...
 [133.321551   137.2859294  139.4813639  ...  40.94071897  46.81751378
   39.37948608]
 [301.1747081  306.9342702  309.3126323  ... 111.7359607  107.1366704
  105.9139811 ]
 [170.328982   174.9665553  179.3695526  ...  57.2306633   56.78333728
   54.74552091]]


The shape of thumb nail dataset is  (20, 1000)
```

```
[26]: print(X)
      print('\n')
      print('The shape of cubital vein dataset is ', X.shape)
```

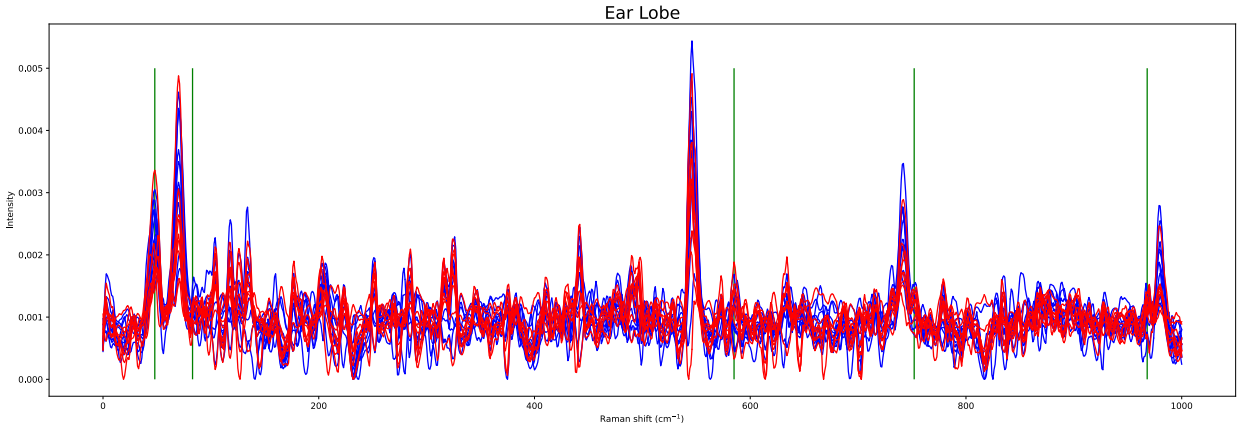
```
[[191.2475708  195.5560272  198.0248413  ...  59.15124054  63.82999878
   65.05268784]
 [262.1858643  266.9431122  271.2298065  ...  91.87278899  90.74000244
   88.29462433]
 [180.2089203  181.0167572  182.0608642  ...  60.97762298  63.40801468
   55.58280182]
 ...
 [282.5561218  284.6205933  286.8745239  ... 101.6480621  108.5540176
   98.52796326]
 [585.4418204  590.9556318  594.1446272  ... 225.6394087  222.9635729
  221.4788796 ]
 [240.7701238  245.1085027  249.0548528  ...  85.07067298  85.62666956
   83.99641737]]


The shape of cubital vein dataset is  (20, 1000)
```

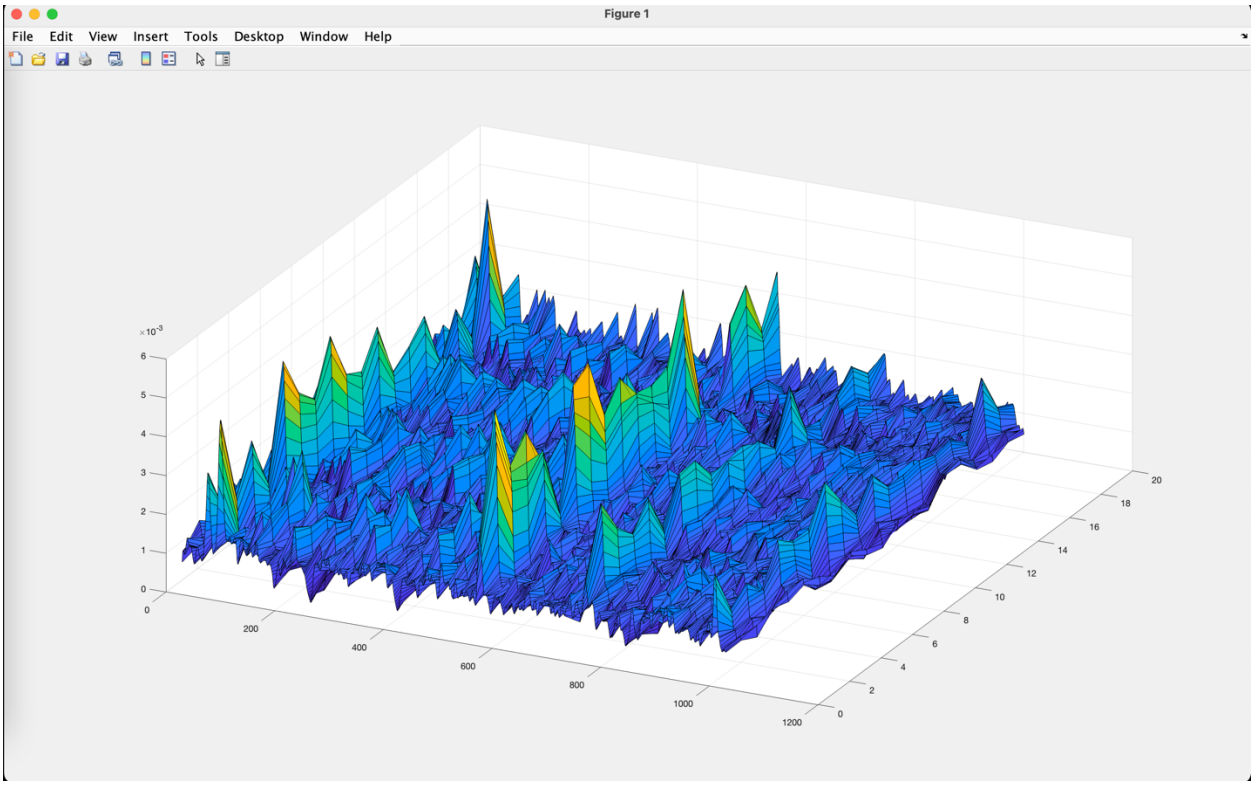
After cropping the signal, the data is then normalized with AUC = 1 in Matlab (this function is defined by Edgar Guevara)

```m
function [ y_normalized ] = norm_auc( x, y)
% SYNTAX
% [ y_normalized ] = norma_auc( x, y)
%
% INPUTS
% x             abcisse vector (e.g. wave Number)
% y             oordinate vector (e.g. Raman Signal)
%
% OUTPUT
% y_normalized  normalized abcisse vector (i.e. auc=1)
%
%_____________________________________________________________________________
% Copyright (C) 2016 Edgar Guevara, PhD
% CONACYT-Universidad Autónoma de San Luis Potosí
% Coordinación para la Innovación y Aplicación de la Ciencia y la Tecnología
%_____________________________________________________________________________
Area=trapz(x,y);
y_normalized = y./Area;
% trapz(x, y_normalized);
end
```

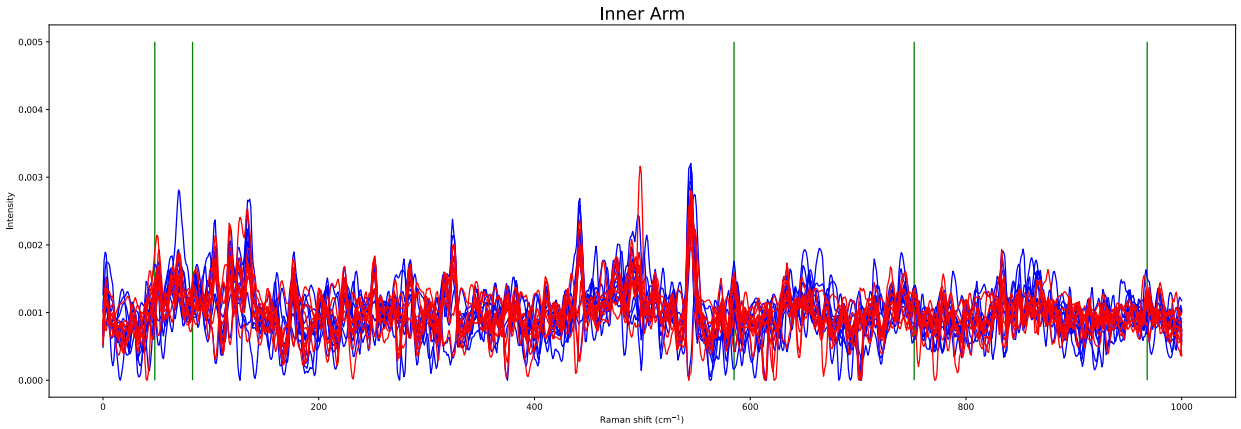
Then after fitting the signal to normalize function, we get normalized data for 4 dataset:

1. Ear Lobe:





2. Inner Arm


Inner Arm
Intensity
Raman shift (cm⁻¹)
0.005
0.004
0.003
0.002
0.001
0.000
0
200
400
600
800
1000

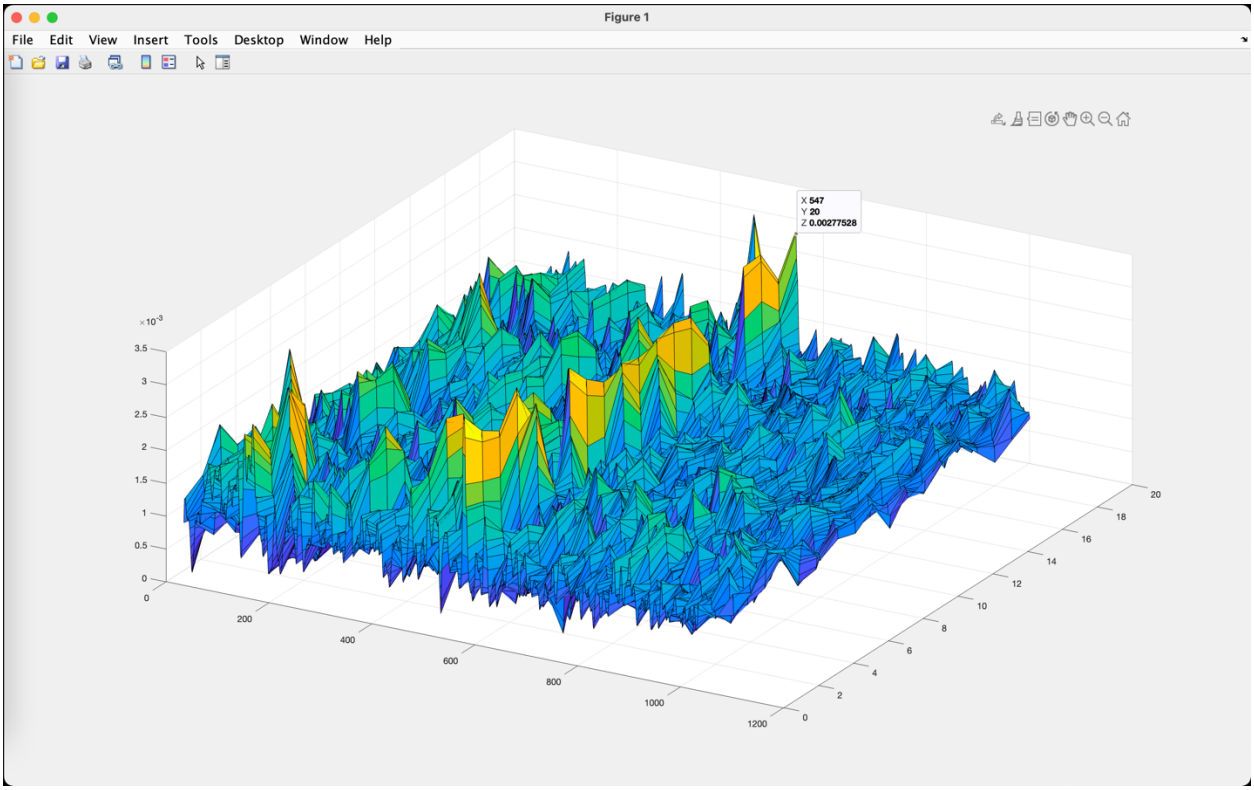
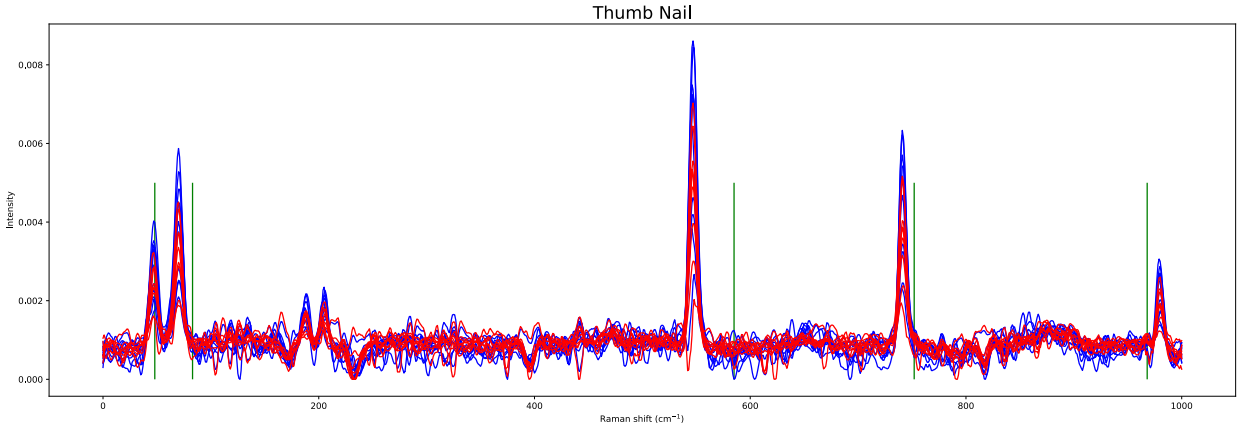
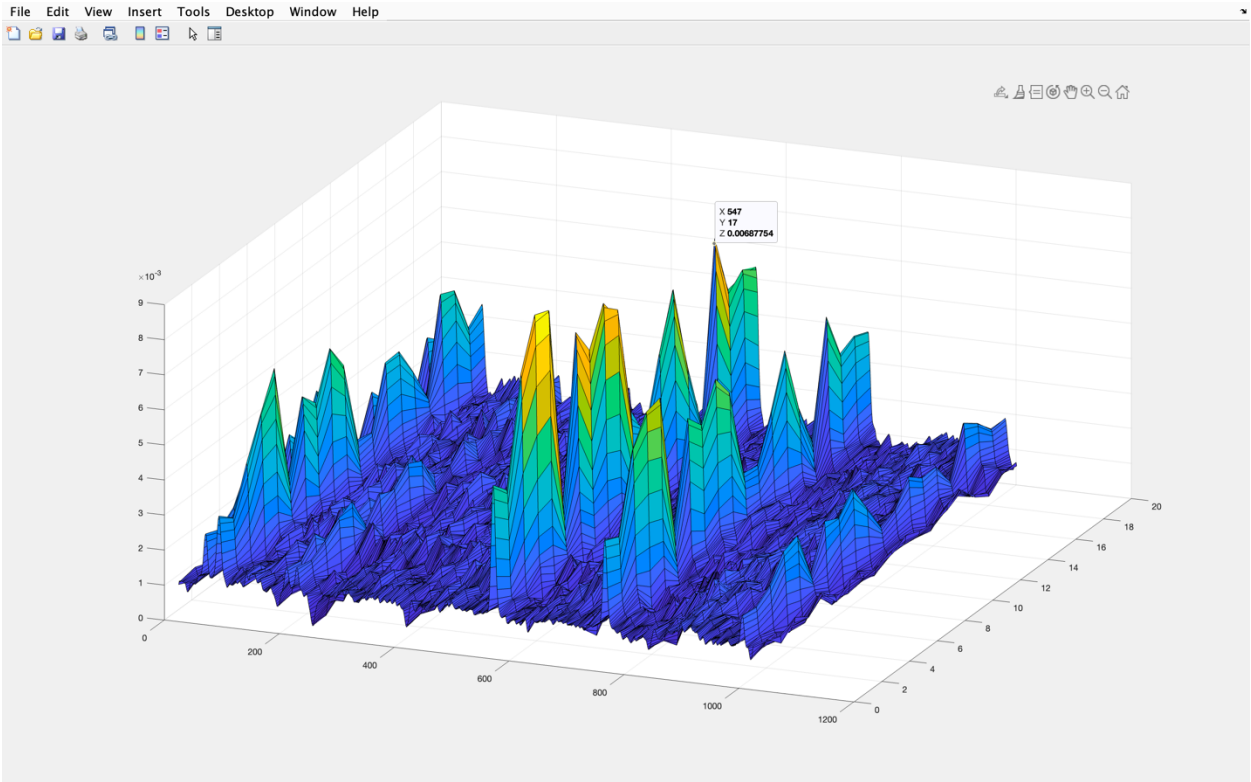


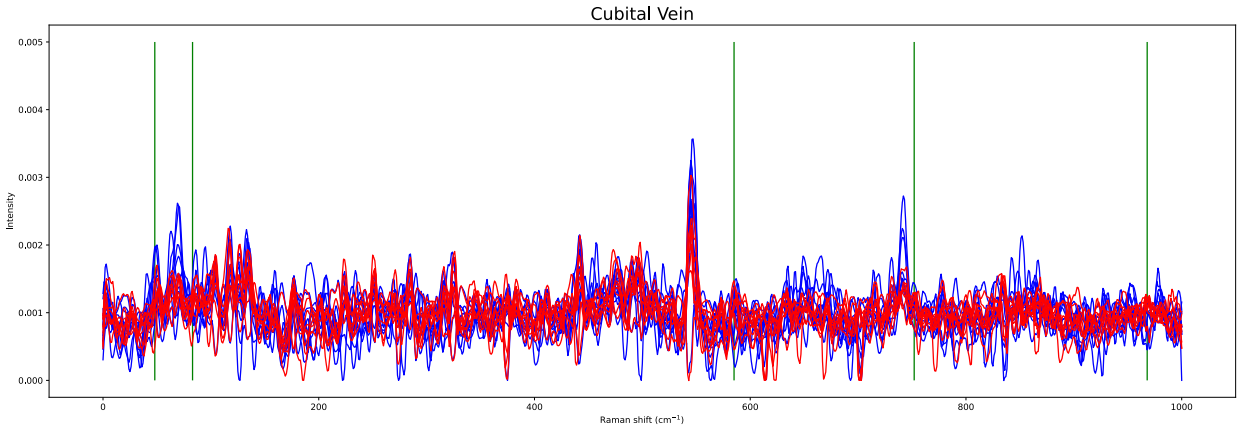
Figure 1
File Edit View Insert Tools Desktop Window Help
X 547
Y 20
Z 0.00277528
×10⁻³
3.5
3
2.5
2
1.5
1
0.5
0
0
200
400
600
800
1000
1200
0
2
4
6
8
10
12
14
16
18
20

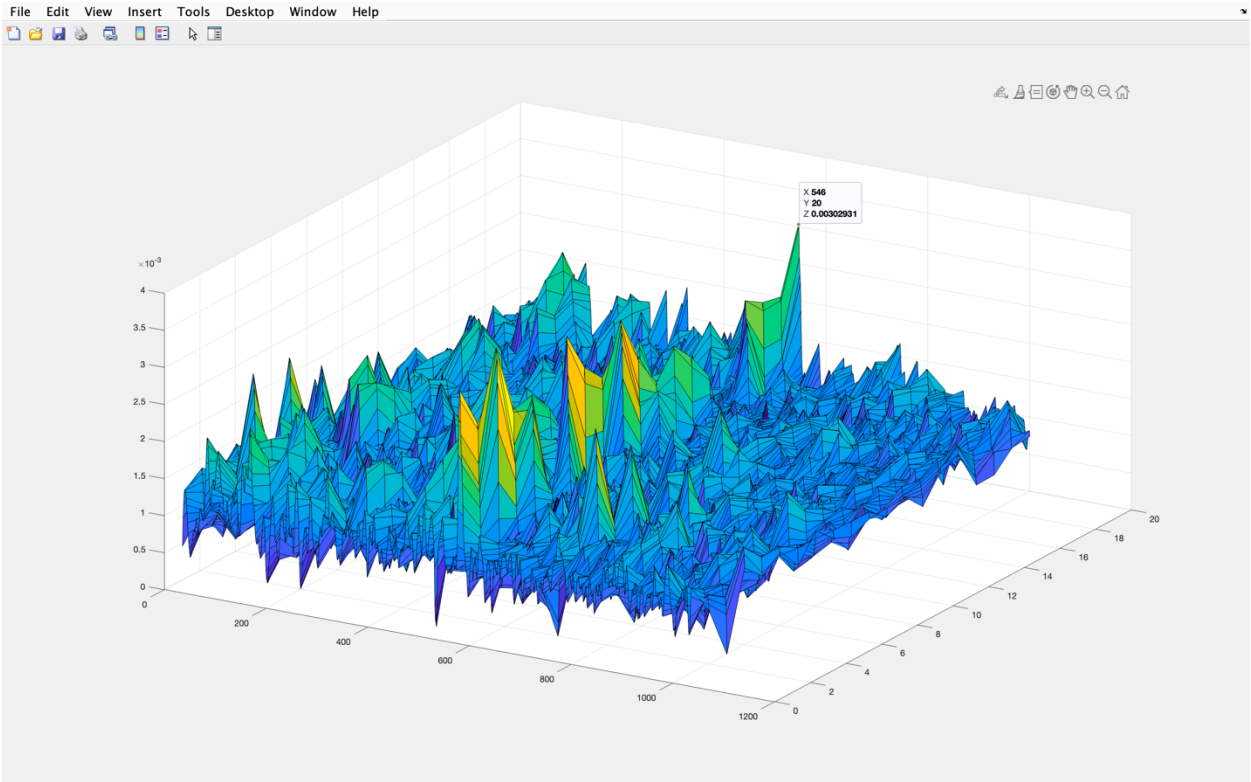
3. Thumb Nail


Thumb Nail
Intensity
0.008
0.006
0.004
0.002
0.000
0
200
400
600
800
1000
Raman shift (cm⁻¹)


File Edit View Insert Tools Desktop Window Help
X 547
Y 17
Z 0.00687754
×10⁻³
9
8
7
6
5
4
3
2
1
0
0
200
400
600
800
1000
1200
0
2
4
6
8
10
12
14
16
18
20

4. Cubital Vein



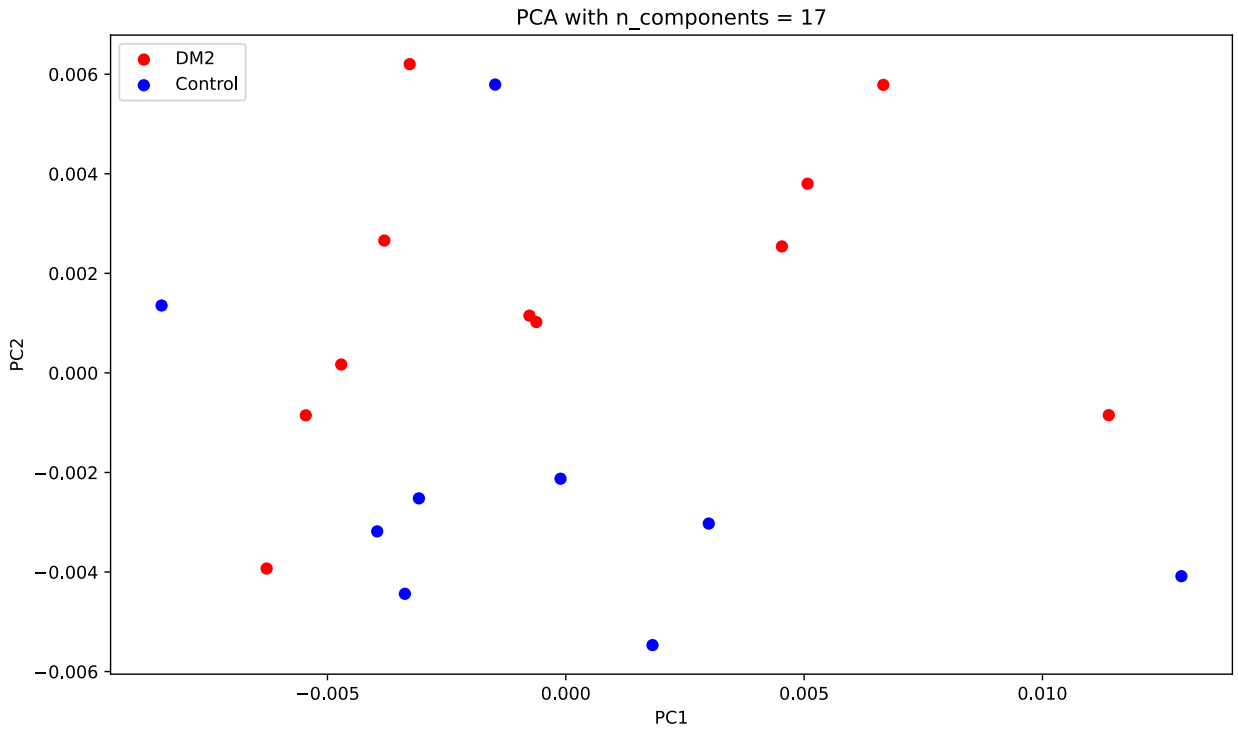


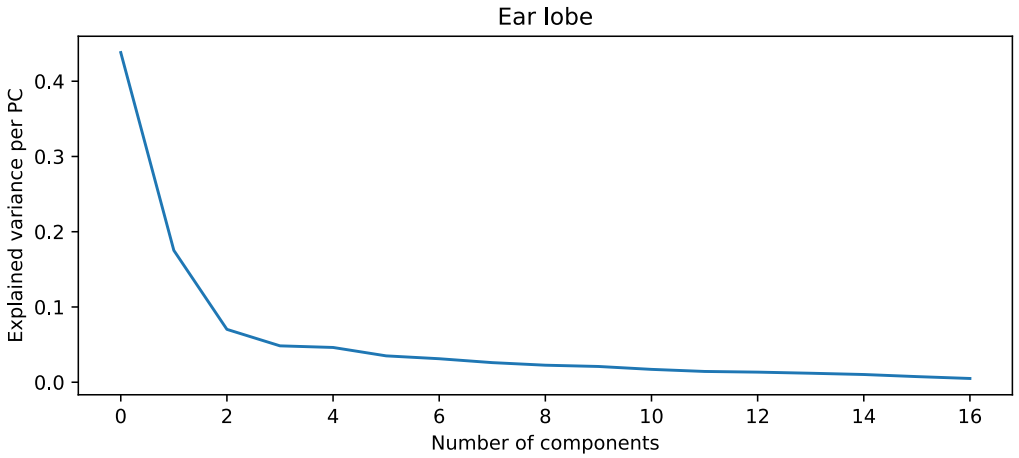
There is no exact seperation between 2 classes.

Dimensionality Reduction

Given 1000 dimension for each observation, it is hard to fit into an estimator with large number of features. The main idea of principal component analysis (PCA) is to reduce the dimensionality of a data set consisting of many variables correlated with each other, either heavily or lightly, while retaining the variation present in the dataset, up to the maximum extent. The same is done by transforming the variables to a new set of variables, which are known as the principal components (PCs) and are orthogonal, ordered such that the retention of variation present in the original variables decreases as we move down in the order. So, in this way, the 1st principal component retains maximum variation that was present in the original components. The principal components are the eigenvectors of a covariance matrix, and hence they are orthogonal.

As described in [], a PCA step is necessary to solve problems with high dimension data. [] shows that PCA-SVM analysis added to Raman spectroscopy significantly increased the accuracy of this diagnostic tool.



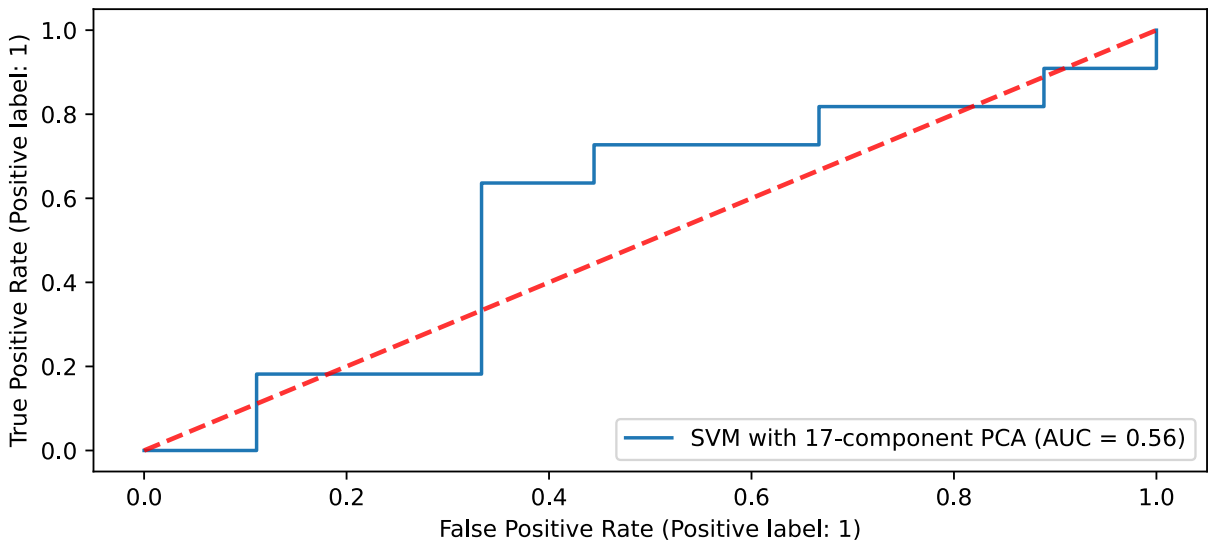


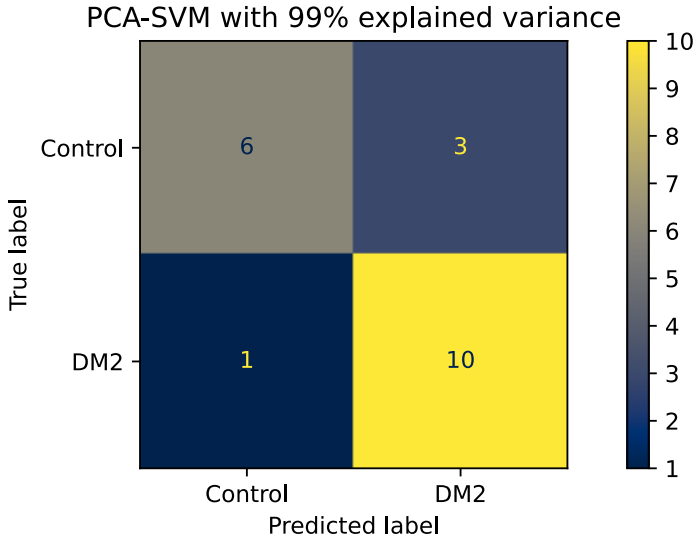
The initial number of components is chosen to match 99% explained variance ratio

Support Vector Machine

After getting PCA result, Stratified k-fold was applied for cross-validation with k = 10. The following parameter was chosen

[59]:
```python
# Run classifier with cross-validation and plot ROC curves
cv = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)
classifier = SVC(kernel='rbf', probability=True, random_state=42)

tprs = []
aucs = []
y_test = []
y_pred_proba = []
mean_fpr = np.linspace(0, 1, 100)

X = X_pca
y = target

fig, ax = plt.subplots(figsize=(15,10))
for i, (train, test) in enumerate(cv.split(X, y)):

    print(f'FOLD {i+1} Test set')
    # print(y[test])

    classifier.fit(X[train], y[train])

    y_test.append(y[test].to_list()[0])
    y_test.append(y[test].to_list()[1])
    y_pred_proba.append(classifier.predict_proba(X[test]))

    RocCurveDisplay.from_estimator(classifier, X[test], y[test])

    print(f'y_test : {classifier.predict(X[test])} \n')
    print('--------------------')
```





...	precision	recall	f1-score	support
0.0	0.86	0.67	0.75	9
1.0	0.77	0.91	0.83	11
accuracy			0.80	20
macro avg	0.81	0.79	0.79	20
weighted avg	0.81	0.80	0.80	20

Code Github: https://github.com/ladcva/raman_spectroscopy

Todos

- Run linear kernel SVM with each 2-component in 17 dimension space after PCA
- Run grid search to find best parameter for SVM model
- Investigate in One-shot Learning and Few-shot Learning technique
- Construct LSTM Neural Network for classification task