hw6

July 16, 2018

1 homework6

1.1 Name: Yiming Yan

1.2 USCID: 9932750243

1.2.1 (a) Download the Anuran Calls (MFCCs) Data Set. Choose 70% of the data randomly as the training set.

```
In [1]: import pandas as pd
        import random
        import numpy as np
        import math
```

```
In [2]: def split_data(o_data,train_per=0.7):
            data=o_data.copy()
            columns=data.columns
            training_data=pd.DataFrame(columns=columns)
            test_data=pd.DataFrame(columns=columns)
            length=len(data.values)
            l=round(length*train_per)
            ran_num=range(length)
            i=0
            while i<1:
                k=random.choice(ran_num)
                index=list(training_data.index)
                train_length=len(training_data.values)
                if train_length==l:
                    break
                else:
                  if k in index:
                     continue
                  else:
                    training_data=training_data.append(data.loc[k])
                    data.drop([k],inplace=True)
            test_data=data
            training_data.index=range(len(training_data))
            test_data.index=range(len(test_data))
            return training_data,test_data
```

```
In [3]: origin_set=pd.DataFrame(pd.read_csv('G:/EE559/hw/hw6/Frogs_MFCCs.csv'))
        training_set,test_set=split_data(origin_set)

In [30]: training_set
```

Out[30]:

	MFCCs_ 1	MFCCs_ 2	MFCCs_ 3	MFCCs_ 4	MFCCs_ 5	MFCCs_ 6	MFCCs_ 7	\
0	1.000000	0.331730	0.018946	0.398650	0.400487	0.104183	-0.227688	
1	1.000000	0.155049	0.125537	0.576056	0.232781	-0.047139	-0.089704	
2	1.000000	0.308728	0.234934	0.527024	0.178472	0.009294	-0.140521	
3	1.000000	0.447066	0.398681	0.512818	0.064754	0.001957	-0.033550	
4	1.000000	0.118416	0.336452	0.287914	0.163894	0.220670	0.062985	
5	1.000000	0.235832	0.353584	0.262535	0.126709	0.186260	0.072340	
6	1.000000	0.394704	0.420662	0.352724	0.046914	0.105684	0.095945	
7	1.000000	0.240077	0.219120	0.574663	0.235800	0.003970	-0.101245	
8	1.000000	0.397913	0.275841	0.455123	0.092774	0.024469	-0.061071	
9	1.000000	0.260914	0.195402	0.497887	0.168214	0.036571	-0.090312	
10	1.000000	0.449248	0.319378	0.596528	0.226209	-0.012801	-0.179358	
11	1.000000	0.123143	0.275968	0.367521	0.221370	0.218392	0.076591	
12	1.000000	0.494415	0.716297	0.719341	-0.566960	0.284962	0.935137	
13	1.000000	0.348169	0.361122	0.563090	0.122009	0.023332	-0.055133	
14	1.000000	0.309286	-0.076083	-0.036950	-0.017738	0.370032	0.562586	
15	1.000000	0.713326	0.386898	0.478786	0.126294	0.001281	-0.019061	
16	1.000000	0.509845	0.653281	0.540366	-0.242250	-0.156556	0.409252	
17	1.000000	0.289052	0.782472	0.337615	-0.336413	0.332130	0.346450	
18	1.000000	0.140045	0.332487	0.323340	0.160920	0.218701	0.101077	
19	1.000000	0.230407	0.448832	0.599867	0.076521	-0.023593	-0.135755	
20	1.000000	0.231222	0.060272	0.474235	0.214972	-0.016644	-0.175745	
21	1.000000	0.338028	0.354497	0.530717	0.109942	0.014237	-0.111019	
22	1.000000	0.546294	0.508594	0.430515	-0.161305	-0.031026	0.339025	
23	1.000000	0.187939	0.129238	0.513947	0.173136	-0.105928	-0.160451	
24	1.000000	0.360613	0.276497	0.508723	0.158514	0.068847	-0.039742	
25	1.000000	0.136023	0.314181	0.342524	0.188271	0.221157	0.139841	
26	1.000000	0.126523	-0.040482	0.341129	0.381446	0.261154	-0.017049	
27	1.000000	0.170717	0.144414	0.549158	0.211581	-0.024628	-0.114756	
28	1.000000	0.117212	0.342653	0.396098	0.196066	0.222107	0.115975	
29	1.000000	0.185114	0.189778	0.544999	0.257440	0.137223	-0.056784	
...	...	...	...	...	...	...	...	
5006	1.000000	0.202650	0.143126	0.514337	0.210715	0.051945	-0.129845	
5007	0.353828	0.738706	1.000000	0.058063	-0.007726	-0.017785	0.251147	
5008	1.000000	0.200545	0.040068	0.554907	0.204934	0.032694	-0.216340	
5009	1.000000	0.551762	0.718145	0.512648	-0.214054	-0.033060	0.444853	
5010	1.000000	0.298836	0.248292	0.543452	0.168648	0.074394	-0.063312	
5011	1.000000	0.518024	0.666760	0.556458	-0.219730	-0.103481	0.359989	
5012	1.000000	0.536769	0.567899	0.527404	-0.145212	-0.070823	0.395728	
5013	1.000000	0.544686	0.439680	0.218671	0.023406	0.179262	0.057246	
5014	1.000000	0.561337	0.468984	0.365545	0.106437	0.184007	0.016814	
5015	1.000000	0.756413	0.732319	0.385807	-0.034821	0.073881	-0.058027	
5016	1.000000	0.228557	0.133194	0.474253	0.138945	0.024468	-0.112011	

5017	1.000000	0.282017	0.175524	0.553760	0.142184	0.075520	0.010301
5018	1.000000	0.657327	0.276159	0.466472	0.164187	0.002949	-0.084537
5019	1.000000	0.378115	0.303957	0.594208	0.182674	0.039678	-0.087378
5020	1.000000	0.369604	0.244297	0.367811	0.125979	0.017008	0.068207
5021	1.000000	0.400885	-0.046711	0.016626	0.249026	0.215358	0.048200
5022	1.000000	0.091751	0.331106	0.398033	0.218120	0.206887	0.121458
5023	1.000000	0.219705	0.531895	0.239888	-0.166340	0.335300	0.268786
5024	1.000000	0.373753	0.246475	0.420087	0.078226	0.051154	0.098371
5025	1.000000	0.251343	0.626495	0.238225	-0.223535	0.412602	0.326805
5026	1.000000	0.376366	0.186578	0.704748	0.354129	0.037594	-0.258755
5027	0.391752	0.060603	1.000000	0.530521	-0.231525	0.202215	-0.100826
5028	1.000000	0.054280	0.550081	0.358632	0.075983	0.250352	0.124023
5029	1.000000	0.672225	0.245210	0.520395	0.163773	0.007445	-0.146772
5030	1.000000	-0.064242	0.750258	0.606765	-0.013647	-0.071581	-0.066679
5031	1.000000	0.373900	0.418835	0.703118	0.137848	0.032452	-0.048695
5032	1.000000	0.497009	0.605041	0.520536	-0.168562	-0.085801	0.406189
5033	1.000000	0.616759	0.723097	0.423993	0.081408	0.150938	0.002346
5034	1.000000	-0.064172	0.489457	0.416768	0.117747	0.208769	0.080956
5035	1.000000	0.174428	0.231712	0.577202	0.181156	0.041322	-0.133516

	MFCCs_ 8	MFCCs_ 9	MFCCs_10	...	MFCCs_17	MFCCs_18	MFCCs_19	\
0	-0.090622	0.164318	0.255465	...	-0.184352	0.069889	0.165210	
1	0.156122	0.200915	-0.064862	...	0.088486	0.006328	-0.062312	
2	0.110709	0.326415	-0.014896	...	0.161011	0.055237	-0.063321	
3	0.078006	0.232600	-0.001685	...	0.196299	-0.043328	-0.045204	
4	-0.162259	-0.030521	0.093829	...	0.089309	0.002693	0.000863	
5	-0.100870	-0.004227	0.069698	...	0.068307	-0.017618	-0.008540	
6	-0.040442	0.043874	0.173450	...	0.022360	-0.006503	0.017768	
7	0.079260	0.292515	0.118585	...	0.232252	0.036825	-0.046847	
8	0.092801	0.289208	-0.006060	...	0.182672	-0.002742	-0.148895	
9	0.064087	0.289242	0.018740	...	0.197074	0.109075	-0.053968	
10	0.034145	0.337430	0.023163	...	0.189402	0.014251	-0.074459	
11	-0.181694	-0.054109	0.103885	...	0.115976	0.045906	-0.012504	
12	-0.036480	-0.520712	0.159460	...	0.023378	0.062315	0.073659	
13	0.114721	0.287866	0.001201	...	0.167501	-0.041628	-0.061758	
14	0.191680	-0.345951	-0.343723	...	-0.230722	-0.035997	-0.015390	
15	0.081164	0.230161	0.059517	...	0.162731	0.039506	-0.017147	
16	0.086875	-0.296439	0.130367	...	0.012871	0.181440	0.013746	
17	-0.263158	-0.032527	0.228686	...	0.081724	0.026146	-0.073252	
18	-0.149016	-0.064823	0.120945	...	0.117539	0.051347	-0.005590	
19	0.144281	0.325694	-0.031479	...	0.222756	0.012389	-0.079069	
20	0.050639	0.255028	0.076444	...	0.168302	-0.014224	-0.145391	
21	0.037581	0.260263	-0.043006	...	0.207760	-0.035013	-0.102734	
22	0.042329	-0.217117	0.177639	...	0.027343	0.055294	-0.106551	
23	0.140682	0.282613	-0.003009	...	0.089049	-0.089553	-0.126195	
24	0.073935	0.306667	0.077962	...	0.216225	0.041289	-0.024708	
25	-0.168543	-0.077071	0.096265	...	0.093931	0.030588	0.004550	
26	-0.294064	-0.222278	0.282338	...	-0.071544	-0.060630	-0.067230	

```
27    0.123335  0.253060 -0.055609    ...     0.134105  0.011404 -0.058421
28   -0.159555 -0.046428  0.132759    ...     0.095513  0.048738  0.017728
29    0.006148  0.260712  0.029221    ...     0.222170 -0.074660 -0.119579
...        ...       ...       ...    ...          ...       ...       ...
5006 -0.021634  0.206101  0.019852    ...     0.218563  0.096637 -0.128033
5007 -0.009766 -0.045430 -0.073709    ...     0.034332  0.037496 -0.055489
5008  0.002779  0.262262  0.061340    ...     0.220403  0.006996 -0.143599
5009  0.081867 -0.312846  0.196871    ...     0.076298  0.112130 -0.174369
5010  0.075835  0.320798  0.011334    ...     0.177445 -0.033864 -0.121435
5011  0.000542 -0.339366  0.131569    ...    -0.000547  0.144797 -0.118098
5012 -0.003123 -0.289558  0.164059    ...     0.081056  0.087179 -0.100578
5013 -0.090659  0.079299  0.089402    ...    -0.021332 -0.034789  0.004906
5014  0.056236  0.222406 -0.052234    ...    -0.105888  0.031633  0.120533
5015  0.183775  0.219419 -0.445112    ...    -0.182019  0.006296 -0.005868
5016  0.075913  0.323418 -0.028529    ...     0.192672  0.013936 -0.133675
5017  0.116700  0.192931  0.084691    ...     0.248258  0.132922 -0.085529
5018 -0.012962  0.240007  0.002402    ...     0.181692  0.068267 -0.003826
5019  0.043905  0.305400  0.018757    ...     0.237367  0.039808 -0.135040
5020 -0.010902 -0.088036  0.055345    ...    -0.093531  0.099070  0.058483
5021 -0.027456  0.111209  0.178018    ...    -0.065566 -0.153994 -0.075444
5022 -0.130602 -0.057117  0.095442    ...     0.115139  0.060286  0.039668
5023 -0.169379  0.051430  0.115948    ...     0.037371 -0.008438  0.014969
5024 -0.038033 -0.217526 -0.026579    ...    -0.142855  0.055312 -0.023529
5025 -0.201316  0.086767  0.101075    ...     0.045672 -0.081784  0.009511
5026  0.035887  0.317806  0.154039    ...     0.095530  0.056634 -0.005075
5027  0.188864  0.161072 -0.402714    ...     0.394586  0.095154 -0.185053
5028 -0.165860 -0.110445  0.050673    ...     0.151060  0.023675  0.005238
5029  0.085799  0.329690  0.063946    ...     0.217742 -0.030439 -0.091215
5030  0.188915  0.194248 -0.153385    ...     0.070819  0.007139 -0.062212
5031  0.093771  0.281103 -0.040590    ...     0.179023 -0.082026 -0.122191
5032  0.062033 -0.320859  0.102065    ...     0.033155  0.142101 -0.065467
5033  0.138806  0.211282 -0.211649    ...    -0.149854 -0.064224 -0.025762
5034 -0.103185 -0.056210  0.057235    ...     0.125545  0.042975 -0.031189
5035 -0.012741  0.193100 -0.045363    ...     0.185206 -0.135896 -0.195989

      MFCCs_20  MFCCs_21  MFCCs_22          Family          Genus  \
0     0.011003 -0.069136 -0.069716   Dendrobatidae       Ameerega
1    -0.074370  0.078301  0.128258  Leptodactylidae      Adenomera
2    -0.115348  0.097101  0.229945  Leptodactylidae      Adenomera
3    -0.019360  0.087126  0.160549  Leptodactylidae      Adenomera
4    -0.042927 -0.107604  0.003791          Hylidae      Hypsiboas
5    -0.005693 -0.022963  0.032874          Hylidae      Hypsiboas
6     0.012500 -0.029048 -0.026501          Hylidae      Hypsiboas
7    -0.126784 -0.086751  0.181513  Leptodactylidae      Adenomera
8    -0.170784  0.100100  0.166811  Leptodactylidae      Adenomera
9    -0.164694  0.038401  0.202065  Leptodactylidae      Adenomera
10   -0.121436  0.022453  0.198188  Leptodactylidae      Adenomera
11    0.019083 -0.066537 -0.073459          Hylidae      Hypsiboas
```

12	0.190577	0.158191	0.050206	Bufonidae	Rhinella
13	-0.089028	0.007104	0.142397	Leptodactylidae	Adenomera
14	-0.008444	0.152844	0.131876	Dendrobatidae	Ameerega
15	-0.045201	0.061611	0.189305	Leptodactylidae	Adenomera
16	-0.076975	0.036776	0.072065	Leptodactylidae	Adenomera
17	0.002443	0.062916	-0.044216	Leptodactylidae	Leptodactylus
18	-0.036005	-0.045596	0.018125	Hylidae	Hypsiboas
19	-0.100352	0.029063	0.120259	Leptodactylidae	Adenomera
20	-0.104062	0.115421	0.209586	Leptodactylidae	Adenomera
21	-0.063321	0.123559	0.188292	Leptodactylidae	Adenomera
22	-0.031339	0.104696	0.019413	Leptodactylidae	Adenomera
23	-0.039565	0.114205	0.055593	Leptodactylidae	Adenomera
24	-0.109553	-0.041444	0.146975	Leptodactylidae	Adenomera
25	-0.023813	-0.066641	-0.012616	Hylidae	Hypsiboas
26	-0.038196	0.070127	0.048440	Leptodactylidae	Adenomera
27	-0.061842	0.097548	0.120786	Leptodactylidae	Adenomera
28	0.001252	-0.069369	-0.029865	Hylidae	Hypsiboas
29	-0.038229	0.171278	0.201467	Leptodactylidae	Adenomera
...	...	...	...	...	...
5006	-0.206245	0.016843	0.218825	Leptodactylidae	Adenomera
5007	0.025635	0.026718	0.014445	Hylidae	Hypsiboas
5008	-0.176058	0.033527	0.229847	Leptodactylidae	Adenomera
5009	-0.104454	0.018764	-0.028911	Leptodactylidae	Adenomera
5010	-0.126118	0.069974	0.234257	Leptodactylidae	Adenomera
5011	-0.077529	0.078134	-0.016718	Leptodactylidae	Adenomera
5012	-0.026648	0.067888	0.003876	Leptodactylidae	Adenomera
5013	-0.025547	-0.018980	0.039341	Hylidae	Hypsiboas
5014	0.068179	0.047785	-0.114410	Hylidae	Hypsiboas
5015	0.029439	0.004783	-0.198014	Hylidae	Hypsiboas
5016	-0.212496	0.051246	0.224806	Leptodactylidae	Adenomera
5017	-0.176131	0.052579	0.246340	Leptodactylidae	Adenomera
5018	-0.099102	-0.008368	0.160634	Leptodactylidae	Adenomera
5019	-0.166045	0.063398	0.206537	Leptodactylidae	Adenomera
5020	-0.056739	-0.038908	0.008616	Dendrobatidae	Ameerega
5021	0.090291	0.099940	0.023965	Leptodactylidae	Adenomera
5022	0.008710	-0.064771	-0.024435	Hylidae	Hypsiboas
5023	0.116018	0.051397	-0.072896	Leptodactylidae	Leptodactylus
5024	-0.102049	-0.018633	0.026146	Dendrobatidae	Ameerega
5025	0.122360	0.011990	-0.121023	Leptodactylidae	Leptodactylus
5026	-0.120230	0.001882	0.169984	Leptodactylidae	Adenomera
5027	0.060182	-0.056968	-0.025700	Hylidae	Scinax
5028	0.005753	-0.063041	-0.008374	Hylidae	Hypsiboas
5029	-0.184621	-0.072720	0.189081	Leptodactylidae	Adenomera
5030	0.026286	0.039517	0.039943	Hylidae	Scinax
5031	-0.100669	0.142209	0.243593	Leptodactylidae	Adenomera
5032	-0.058481	0.058066	0.000468	Leptodactylidae	Adenomera
5033	0.073420	0.049090	-0.157737	Hylidae	Hypsiboas
5034	-0.018791	-0.081901	-0.019989	Hylidae	Hypsiboas

```
5035 -0.095111  0.148506  0.201124  Leptodactylidae      Adenomera

                     Species  RecordID
0         Ameeregatrivittata        11
1     AdenomeraHylaedactylus        18
2     AdenomeraHylaedactylus        23
3     AdenomeraHylaedactylus        19
4          HypsiboasCordobae        41
5          HypsiboasCordobae        40
6          HypsiboasCordobae        41
7     AdenomeraHylaedactylus        22
8     AdenomeraHylaedactylus        14
9     AdenomeraHylaedactylus        23
10    AdenomeraHylaedactylus        19
11         HypsiboasCordobae        42
12         Rhinellagranulosa        53
13    AdenomeraHylaedactylus        19
14        Ameeregatrivittata        12
15    AdenomeraHylaedactylus        22
16            AdenomeraAndre         8
17       LeptodactylusFuscus        47
18         HypsiboasCordobae        43
19    AdenomeraHylaedactylus        19
20    AdenomeraHylaedactylus        18
21    AdenomeraHylaedactylus        19
22            AdenomeraAndre         8
23    AdenomeraHylaedactylus        18
24    AdenomeraHylaedactylus        22
25         HypsiboasCordobae        40
26            AdenomeraAndre         1
27    AdenomeraHylaedactylus        18
28         HypsiboasCordobae        40
29    AdenomeraHylaedactylus        24
...                     ...       ...
5006  AdenomeraHylaedactylus        16
5007    HypsiboasCinerascens        36
5008  AdenomeraHylaedactylus        15
5009          AdenomeraAndre         8
5010  AdenomeraHylaedactylus        20
5011          AdenomeraAndre         8
5012          AdenomeraAndre         8
5013       HypsiboasCordobae        42
5014    HypsiboasCinerascens        38
5015    HypsiboasCinerascens        36
5016  AdenomeraHylaedactylus        23
5017  AdenomeraHylaedactylus        16
5018  AdenomeraHylaedactylus        22
5019  AdenomeraHylaedactylus        19
```

```
5020      Ameeregatrivittata        13
5021          AdenomeraAndre         4
5022       HypsiboasCordobae        41
5023     LeptodactylusFuscus        47
5024      Ameeregatrivittata        13
5025     LeptodactylusFuscus        47
5026  AdenomeraHylaedactylus        20
5027             ScinaxRuber        56
5028       HypsiboasCordobae        41
5029  AdenomeraHylaedactylus        22
5030             ScinaxRuber        56
5031  AdenomeraHylaedactylus        19
5032          AdenomeraAndre         8
5033   HypsiboasCinerascens        36
5034       HypsiboasCordobae        41
5035  AdenomeraHylaedactylus        24

[5036 rows x 26 columns]
```

1.2.2 (b)

i. Research exact match and hamming score/ loss methods for evaluating multilabel classi cation and use them in evaluating the classifiers in this problem. Accuracy score/ Exact match metric: This function calculates subset accuracy meaning the predicted set of labels should exactly match with the true set of labels.Exact match is the most strict metric, indicating the percentage of samples that have all their labels classified correctly.

Hamming loss is the fraction of the wrong labels to the total number of labels.

```
In [4]: def exact_match(y_true,y_predict):
            length,width=y_true.shape
            count=0
            for true,predict in zip(y_true,y_predict):
                true=list(true)
                predict=list(predict)
                if true==predict:
                    count+=1
            fraction=count/length
            return fraction
```

ii. Train a SVM for each of the labels, using Gaussian kernels and one versus all classifiers. Determine the weight of the SVM penalty and the width of the Gaussian Kernel using 10 fold cross validation. You are welcome to try to solve the problem with both normalized and raw attributes and report the results.

```
In [5]: from sklearn.model_selection  import GridSearchCV
        from sklearn.svm import SVC
        from sklearn.metrics import hamming_loss
```

```
In [6]: def extrate_data(data):
            features=np.array(data.drop(['Family', 'Genus', 'Species', 'RecordID'], axis=1))
            label1=np.array(data['Family'])
            label2=np.array(data['Genus'])
            label3=np.array(data['Species'])
            return features,label1,label2,label3

In [7]: # Gaussian Kernel width and Parameter C
        sigma=[i/10 for i in range(1,21,1)]
        gamma=[1/(2*sig*sig) for sig in sigma]
        param_grid={'kernel':['rbf'],'C':[0.01,0.1,1,10,100],'gamma':gamma}

In [8]: print(param_grid['gamma'])

[49.99999999999999, 12.499999999999998, 5.555555555555555, 3.1249999999999996, 2.0, 1.38888888

In [9]: training_data,training_family,training_genus,training_species=extrate_data(training_se
        test_data,test_family,test_genus,test_species=extrate_data(test_set)

In [10]: svc1 = SVC()
         svc2 = SVC()
         svc3 = SVC()
         family_svc = GridSearchCV(svc1,param_grid,cv = 10,refit=True, n_jobs=-1)
         genus_svc = GridSearchCV(svc2,param_grid,cv = 10,refit=True,n_jobs=-1)
         species_svc = GridSearchCV(svc3,param_grid,cv = 10,refit=True,n_jobs=-1)

In [11]: family_model=family_svc.fit(training_data,training_family)
         family_pre = family_model.predict(test_data)
         wFamily = family_model.best_params_
         hammingloss1=hamming_loss(test_family,family_pre)

In [12]: genus_model=genus_svc.fit(training_data,training_genus)
         genus_pre = genus_model.predict(test_data)
         wGenus = genus_model.best_params_
         hammingloss2=hamming_loss(test_genus,genus_pre)

In [13]: species_model=species_svc.fit(training_data,training_species)
         species_pre =species_model.predict(test_data)
         wSpecies = species_model.best_params_

         label_pre = np.array([family_pre, genus_pre, species_pre]).T
         label_true= np.array([test_family, test_genus,test_species]).T

         hammingloss3=hamming_loss(test_species,species_pre)
         hammingloss=(hammingloss1+hammingloss2+hammingloss3)/3

         exactmatch=exact_match(label_true,label_pre)
```

```
In [14]: print(" penalty and width for Family: ",wFamily['C']," The gamma is",wFamily['gamma']
               "The sigma is",math.sqrt(1/(2*wFamily['gamma'])))
         print("penalty and width for Genus: ",wGenus['C']," The gamma is",wGenus['gamma'],
               "The sigma is",math.sqrt(1/(2*wGenus['gamma'])))
         print("penalty and width for Species: ",wSpecies['C']," The gamma is",wSpecies['gamma
               "The sigma is",math.sqrt(1/(2*wSpecies['gamma'])))
         print("exact match", exactmatch)
         print("hamming loss",hammingloss)
```

```
 penalty and width for Family:  10  The gamma is 3.1249999999999996 The sigma is 0.4
penalty and width for Genus:  100  The gamma is 1.3888888888888888 The sigma is 0.6
penalty and width for Species:  10  The gamma is 2.0 The sigma is 0.5
exact match 0.9911996294580825
hamming loss 0.0057125212289640265
```

iii. Repeat 1(b)ii with L1-penalized SVMs. Remember to normalize the attributes. According to the description "Mel-frequency cepstral coefficients (MFCCs) are coefficients that collectively make up an mel-frequency cepstrum (MFC). Due to each syllable has different length, every row (i) was normalized acording to MFCCs_i/(max(abs(MFCCs_i)))." , the whole data has been normalized. So, I don't need to normalize the data.

```
In [15]: from sklearn.svm import LinearSVC

In [16]: params = {'C':[0.001,0.01,0.1,1,10,100,1000]}

In [17]: lsvc1 = LinearSVC(penalty='l1', dual=False)
         lsvc2 = LinearSVC(penalty='l1', dual=False)
         lsvc3 = LinearSVC(penalty='l1', dual=False)
         family_svcl=GridSearchCV(lsvc1,params,cv = 10,refit=True,n_jobs=-1)
         genus_svcl= GridSearchCV(lsvc2,params,cv = 10,refit=True,n_jobs=-1)
         species_svcl= GridSearchCV(lsvc3,params,cv = 10,refit=True,n_jobs=-1)

In [18]: family_modell = family_svcl.fit(training_data,training_family)
         family_prel = family_modell.predict(test_data)
         wFamilyl = family_modell.best_params_
         hammingloss11=hamming_loss(test_family,family_prel)

In [19]: genus_modell=genus_svcl.fit(training_data,training_genus)
         genus_prel = genus_modell.predict(test_data)
         wGenusl=genus_modell.best_params_
         hammingloss12=hamming_loss(test_genus,genus_prel)

In [20]: species_modell = species_svcl.fit(training_data,training_species)
         species_prel =species_modell.predict(test_data)
         wSpeciesl = species_modell.best_params_

         label_prel = np.array([family_prel, genus_prel, species_prel]).T
         label_true= np.array([test_family, test_genus,test_species]).T
```

```
hamminglossl3=hamming_loss(test_species,species_prel)
hamminglossl=(hamminglossl1+hamminglossl2+hamminglossl3)/3

exactmatch=exact_match(label_true,label_prel)
```

```
In [21]: print(" penalty and width for Family: ",wFamilyl['C'])
         print("penalty and width for Genus: ",wGenusl['C'])
         print("penalty and width for Species: ",wSpeciesl['C'])
         print("exact match", exactmatch)
         print("hamming loss",hamminglossl)
```

```
 penalty and width for Family:  100
penalty and width for Genus:  100
penalty and width for Species:  100
exact match 0.9212598425196851
hamming loss 0.04616334722865525
```

iv. Repeat 1(b)iii by using SMOTE or any other method you know to remedy class imbalance. Report your conclusions about the classifiers you trained.

```
In [22]: from imblearn.over_sampling import SMOTE
```

```
In [23]: params = {'C':[0.001,0.01, 0.1,1,10,100,1000]}
```

```
In [24]: sm=SMOTE()
         family_data_res,family_label_res = sm.fit_sample(training_data, training_family)
         genus_data_res,genus_label_res = sm.fit_sample(training_data, training_genus)
         species_data_res,species_label_res = sm.fit_sample(training_data, training_species)
```

```
In [25]: lsvc1 = LinearSVC(penalty='l1', dual=False)
         lsvc2 = LinearSVC(penalty='l1', dual=False)
         lsvc3 = LinearSVC(penalty='l1', dual=False)
         family_svcl=GridSearchCV(lsvc1,params,cv = 10,refit=True,n_jobs=-1)
         genus_svcl= GridSearchCV(lsvc2,params,cv = 10,refit=True,n_jobs=-1)
         species_svcl= GridSearchCV(lsvc3,params,cv = 10,refit=True,n_jobs=-1)
```

```
In [26]: family_modell = family_svcl.fit(family_data_res,family_label_res)
         family_prel = family_modell.predict(test_data)
         wFamilyl = family_modell.best_params_
         hamminglossl1=hamming_loss(test_family,family_prel)
```

```
In [27]: genus_modell=genus_svcl.fit(genus_data_res,genus_label_res)
         genus_prel = genus_modell.predict(test_data)
         wGenusl=genus_modell.best_params_
         hamminglossl2=hamming_loss(test_genus,genus_prel)
```

```
In [28]: species_modell = species_svcl.fit(species_data_res,species_label_res)
         species_prel =species_modell.predict(test_data)
         wSpeciesl = species_modell.best_params_

         label_prel = np.array([family_prel, genus_prel, species_prel]).T
         label_true= np.array([test_family, test_genus,test_species]).T

         hamminglossl3=hamming_loss(test_species,species_prel)
         hamminglossl=(hamminglossl1+hamminglossl2+hamminglossl3)/3

         exactmatch=exact_match(label_true,label_prel)

In [29]: print(" penalty and width for Family: ",wFamilyl['C'])
         print("penalty and width for Genus: ",wGenusl['C'])
         print("penalty and width for Species: ",wSpeciesl['C'])
         print("exact match", exactmatch)
         print("hamming loss",hamminglossl)

 penalty and width for Family:  100
penalty and width for Genus:  1000
penalty and width for Species:  100
exact match 0.8679944418712366
hamming loss 0.06824146981627296
```