classification

June 7, 2017

1 Part 2 -- Classification

The libraries that we are going to use

```
In [15]: import pandas as pd
         from sklearn.naive_bayes import MultinomialNB, BernoulliNB
         from sklearn.feature_extraction.text import TfidfTransformer
         from sklearn.feature_extraction.text import CountVectorizer
         import numpy as np
         from sklearn import svm
         from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier
         from sklearn.metrics import classification_report, accuracy_score, auc
         from sklearn.model_selection import KFold
         from sklearn.decomposition import TruncatedSVD
         from sklearn.linear_model import SGDClassifier
         from sklearn import metrics
         import matplotlib.pyplot as plt1
         import matplotlib.pyplot as plt2

         from wordcloud import STOPWORDS
         from sklearn.feature_extraction.text import ENGLISH_STOP_WORDS
         import csv
         import random
         import math
         import operator
         from operator import itemgetter
         from collections import Counter
```

We read our training and testing data:

```
In [16]: Train_data = pd.read_csv(sep='\t',filepath_or_buffer='train.tsv')
         Test_data = pd.read_csv(sep='\t',filepath_or_buffer='test.tsv')
         target = Train_data['Label']
```

```
In [17]: EvaluationMetric = {
             'Statistic Measure':['Accuracy'],
             'Naive Bayes':[] ,
             'Random Forest':[],
             'SVM':[]}
```

```
In [18]: cross_val_instance = 0

         def cross_validate(clf,train_data,target_data):
             global cross_val_instance    # Needed to modify global copy of a global variable

             kf = KFold(n_splits=10)
             average_accuracy =0
             fold = 0
             for train_index, test_index in kf.split(train_data):
                 cross_val_instance += 1

                 test = train_data.loc[test_index, train_data.columns]
                 train = train_data.loc[train_index, train_data.columns]
                 target = target_data[train_index]
                 clf_cv = clf.fit(train, target)
                 yPred = clf_cv.predict(test)
                 fold += 1
                 print ("Fold " + str(fold)+"\n\n")
                 target = target_data[test_index]
                 accuracy = accuracy_score(target, yPred)

                 print("Accuracy: ", accuracy)
                 average_accuracy+= accuracy
             average_accuracy = average_accuracy/10
             print("Average accuracy = ",average_accuracy)
             return average_accuracy
```

1.1 Data preprocessing

```
In [19]: categories = ["Attribute1","Attribute3","Attribute4","Attribute6","Attribute7","Attribu

         proccessedData_train = Train_data.copy()

         for x in categories:
             converted = pd.Categorical(Train_data[x])
             proccessedData_train[x] = converted.codes

         print(proccessedData_train)
```

```
     Attribute1  Attribute2  Attribute3  Attribute4  Attribute5  Attribute6  \
0             0           6           4           4        1169           4
1             1          48           2           4        5951           0
2             3          12           4           7        2096           0
3             0          42           2           3        7882           0
4             0          24           3           0        4870           0
5             3          36           2           7        9055           4
6             3          24           2           3        2835           2
7             1          36           2           1        6948           0
```

8	3	12	2	4	3059	3
9	1	30	4	0	5234	0
10	1	12	2	0	1295	0
11	0	48	2	9	4308	0
12	1	12	2	4	1567	0
13	0	24	4	0	1199	0
14	0	15	2	0	1403	0
15	0	24	2	4	1282	1
16	3	24	4	4	2424	4
17	0	30	0	9	8072	4
18	1	24	2	1	12579	0
19	3	24	2	4	3430	2
20	3	9	4	0	2134	0
21	0	6	2	4	2647	2
22	0	10	4	0	2241	0
23	1	12	4	1	1804	1
24	3	10	4	3	2069	4
25	0	6	2	3	1374	0
26	3	6	0	4	426	0
27	2	12	1	4	409	3
28	1	7	2	4	2415	0
29	0	60	3	9	6836	0
..	...	...	...	...	...	...
770	0	24	2	1	2812	4
771	0	36	4	7	8065	0
772	3	21	4	1	3275	0
773	3	24	4	4	2223	1
774	2	12	4	0	1480	2
775	0	24	2	0	1371	4
776	3	36	4	0	3535	0
777	0	18	2	4	3509	0
778	3	36	4	1	5711	3
779	1	18	2	6	3872	0
780	1	39	4	4	4933	0
781	3	24	4	0	1940	3
782	1	12	0	8	1410	0
783	1	12	2	0	836	1
784	1	20	2	1	6468	4
785	1	18	2	9	1941	3
786	3	22	2	4	2675	2
787	3	48	4	1	2751	4
788	1	48	3	7	6224	0
789	0	40	4	7	5998	0
790	1	21	2	9	1188	0
791	3	24	2	1	6313	4
792	3	6	4	3	1221	4
793	2	24	2	3	2892	0
794	3	24	2	3	3062	2

795	3	9	2	3	2301	1
796	0	18	2	1	7511	4
797	3	12	4	3	1258	0
798	3	24	3	0	717	4
799	1	9	2	0	1549	4

	Attribute7	Attribute8	Attribute9	Attribute10	...	Attribute13	\
0	4	4	2	0	...	67	
1	2	2	1	0	...	22	
2	3	2	2	0	...	49	
3	3	2	2	2	...	45	
4	2	3	2	0	...	53	
5	2	2	2	0	...	35	
6	4	3	2	0	...	53	
7	2	2	2	0	...	35	
8	3	2	0	0	...	61	
9	0	4	3	0	...	28	
10	1	3	1	0	...	25	
11	1	3	1	0	...	24	
12	2	1	1	0	...	22	
13	4	4	2	0	...	60	
14	2	2	1	0	...	28	
15	2	4	1	0	...	32	
16	4	4	2	0	...	53	
17	1	2	2	0	...	25	
18	4	4	1	0	...	44	
19	4	3	2	0	...	31	
20	2	4	2	0	...	48	
21	2	2	2	0	...	44	
22	1	1	2	0	...	48	
23	1	3	2	0	...	44	
24	2	2	3	0	...	26	
25	2	1	2	0	...	36	
26	4	4	3	0	...	39	
27	2	3	1	0	...	42	
28	2	3	2	2	...	34	
29	4	3	2	0	...	63	
..	...	...	...	...	...	...	
770	4	2	1	0	...	26	
771	2	3	1	0	...	25	
772	4	1	2	0	...	36	
773	4	4	2	0	...	52	
774	0	2	2	0	...	66	
775	2	4	1	0	...	25	
776	3	4	2	0	...	37	
777	3	4	1	2	...	25	
778	4	4	2	0	...	38	
779	0	2	1	0	...	67	

780	3	2	2	2	...	25
781	4	4	2	0	...	60
782	2	2	2	0	...	31
783	1	4	1	0	...	23
784	0	1	0	0	...	60
785	2	4	2	0	...	35
786	4	3	2	0	...	40
787	4	4	2	0	...	38
788	4	4	2	0	...	50
789	2	4	2	0	...	27
790	4	2	1	0	...	39
791	4	3	2	0	...	41
792	2	1	3	0	...	27
793	4	3	0	0	...	51
794	4	4	2	0	...	32
795	1	2	1	0	...	22
796	4	1	2	0	...	51
797	1	2	1	0	...	22
798	4	4	3	0	...	54
799	1	4	2	0	...	35

	Attribute14	Attribute15	Attribute16	Attribute17	Attribute18	\
0	2	1	2	2	1	
1	2	1	1	2	1	
2	2	1	1	1	2	
3	2	2	1	2	2	
4	2	2	2	2	2	
5	2	2	1	1	2	
6	2	1	1	2	1	
7	2	0	1	3	1	
8	2	1	1	1	1	
9	2	1	2	3	1	
10	2	0	1	2	1	
11	2	0	1	2	1	
12	2	1	1	2	1	
13	2	1	2	1	1	
14	2	0	1	2	1	
15	2	1	1	1	1	
16	2	1	2	2	1	
17	0	1	3	2	1	
18	2	2	1	3	1	
19	2	1	1	2	2	
20	2	1	3	2	1	
21	2	0	1	2	2	
22	2	0	2	1	2	
23	2	1	1	2	1	
24	2	1	2	2	1	
25	0	1	1	1	1	

26	2	1	1	1	1
27	2	0	2	2	1
28	2	1	1	2	1
29	2	1	2	2	1
..	...	...	...	...	...
770	2	0	1	2	1
771	2	1	2	3	1
772	2	1	1	3	1
773	0	1	2	2	1
774	0	2	3	0	1
775	2	0	1	2	1
776	2	1	2	2	1
777	2	1	1	2	1
778	2	1	2	3	1
779	2	1	1	2	1
780	2	1	2	2	1
781	2	1	1	2	1
782	2	1	1	1	1
783	0	1	1	1	1
784	2	1	1	3	1
785	2	1	1	1	1
786	2	1	1	2	1
787	2	1	2	2	2
788	2	2	1	2	1
789	0	1	1	2	1
790	2	1	1	2	2
791	2	1	1	3	2
792	2	1	2	2	1
793	2	2	1	2	1
794	2	0	1	2	1
795	2	0	1	2	1
796	2	2	1	2	2
797	2	0	2	1	1
798	2	1	2	2	1
799	2	1	1	0	1

	Attribute19	Attribute20	Label	Id
0	1	0	1	10101
1	0	0	2	10102
2	0	0	1	10103
3	0	0	1	10104
4	0	0	2	10105
5	1	0	1	10106
6	0	0	1	10107
7	1	0	1	10108
8	0	0	1	10109
9	0	0	2	10110
10	0	0	2	10111

11	0	0	2	10112
12	1	0	1	10113
13	0	0	2	10114
14	0	0	1	10115
15	0	0	2	10116
16	0	0	1	10117
17	0	0	1	10118
18	1	0	2	10119
19	1	0	1	10120
20	1	0	1	10121
21	0	0	1	10122
22	0	1	1	10123
23	0	0	1	10124
24	0	1	1	10125
25	1	0	1	10126
26	0	0	1	10127
27	0	0	1	10128
28	0	0	1	10129
29	1	0	2	10130
..	...	...	...	...
770	0	0	1	10871
771	1	0	2	10872
772	1	0	1	10873
773	0	0	1	10874
774	0	0	1	10875
775	0	0	2	10876
776	1	0	1	10877
777	0	0	1	10878
778	1	0	1	10879
779	1	0	1	10880
780	0	0	2	10881
781	1	0	1	10882
782	1	0	1	10883
783	0	0	2	10884
784	1	0	1	10885
785	1	0	1	10886
786	0	0	1	10887
787	1	0	1	10888
788	0	0	2	10889
789	1	0	2	10890
790	0	0	2	10891
791	1	0	1	10892
792	0	0	1	10893
793	0	0	1	10894
794	1	0	1	10895
795	0	0	1	10896
796	1	0	2	10897
797	0	0	1	10898

```
798            1            0      1  10899
799            0            0      1  10900

[800 rows x 22 columns]
```

```
In [20]: proccessedData_test = Test_data.copy()

         for x in categories:
             converted = pd.Categorical(Test_data[x])
             proccessedData_test[x] = converted.codes

         print(proccessedData_test)
```

```
     Attribute1  Attribute2  Attribute3  Attribute4  Attribute5  Attribute6  \
0             1          18           4           4        1795           0
1             0          20           4           3        4272           0
2             3          12           4           4         976           4
3             1          12           2           0        7472           4
4             0          36           2           0        9271           0
5             1           6           2           4         590           0
6             3          12           4           4         930           4
7             1          42           1           1        9283           0
8             1          15           0           0        1778           0
9             1           8           2           9         907           0
10            1           6           2           4         484           0
11            0          36           4           1        9629           0
12            0          48           2           5        3051           0
13            0          48           2           0        3931           0
14            1          36           3           0        7432           0
15            3           6           2           5        1338           2
16            3           6           4           4        1554           0
17            0          36           2           2       15857           0
18            0          18           2           4        1345           0
19            3          12           2           0        1101           0
20            2          12           2           4        3016           0
21            0          36           2           3        2712           0
22            0           8           4           0         731           0
23            3          18           4           3        3780           0
24            0          21           4           0        1602           0
25            0          18           4           0        3966           0
26            3          18           0           9        4165           0
27            0          36           2           1        8335           4
28            1          48           3           9        6681           4
29            3          24           3           9        2375           2
..          ...         ...         ...         ...         ...         ...
169           1          15           2           6        1514           1
170           3          24           2           0        7393           0
```

171	0	24	1	0	1193	0
172	0	60	2	9	7297	0
173	3	30	4	4	2831	0
174	2	24	2	4	1258	2
175	1	6	2	4	753	0
176	1	18	3	9	2427	4
177	3	24	3	0	2538	0
178	1	15	1	0	1264	1
179	1	30	4	3	8386	0
180	3	48	2	9	4844	0
181	2	21	2	0	2923	1
182	0	36	2	1	8229	0
183	3	24	4	3	2028	0
184	0	15	4	3	1433	0
185	2	42	0	9	6289	0
186	3	13	2	4	1409	1
187	0	24	2	1	6579	0
188	1	24	4	4	1743	0
189	3	12	4	7	3565	4
190	3	15	1	4	1569	1
191	0	18	2	4	1936	4
192	0	36	2	3	3959	0
193	3	12	2	0	2390	4
194	3	12	2	3	1736	0
195	0	30	2	1	3857	0
196	3	12	2	4	804	0
197	0	45	2	4	1845	0
198	1	45	4	1	4576	1

	Attribute7	Attribute8	Attribute9	Attribute10	...	Attribute12	\
0	4	3	1	2	...	0	
1	4	1	1	0	...	1	
2	4	4	2	0	...	2	
3	0	1	1	0	...	0	
4	3	2	2	0	...	2	
5	1	3	3	0	...	0	
6	4	4	2	0	...	0	
7	0	1	2	0	...	3	
8	1	2	1	0	...	0	
9	1	3	3	0	...	0	
10	3	3	3	2	...	0	
11	3	4	2	0	...	2	
12	2	3	2	0	...	2	
13	3	4	2	0	...	3	
14	2	2	1	0	...	1	
15	2	1	0	0	...	0	
16	3	1	1	0	...	2	
17	0	2	0	1	...	2	

```
18            2           4           3           0  ...           0
19            2           3           3           0  ...           0
20            2           3           3           0  ...           2
21            4           2           2           0  ...           1
22            4           4           2           0  ...           0
23            1           3           0           0  ...           2
24            4           4           3           0  ...           2
25            4           1           1           0  ...           0
26            2           2           2           0  ...           2
27            4           3           2           0  ...           3
28            2           4           2           0  ...           3
29            2           4           2           0  ...           2
..          ...         ...         ...         ...  ...         ...
169           2           4           2           2  ...           0
170           2           1           2           0  ...           1
171           0           1           1           1  ...           3
172           4           4           2           1  ...           3
173           2           4           1           0  ...           2
174           2           3           1           0  ...           2
175           2           2           1           2  ...           0
176           4           4           2           0  ...           1
177           4           4           2           0  ...           2
178           2           2           3           0  ...           1
179           3           2           2           0  ...           1
180           0           3           2           0  ...           2
181           2           1           1           0  ...           2
182           2           2           2           0  ...           1
183           3           2           2           0  ...           1
184           2           4           1           0  ...           1
185           1           2           0           0  ...           1
186           0           2           1           0  ...           0
187           0           4           2           0  ...           3
188           4           4           2           0  ...           1
189           1           2           2           0  ...           1
190           4           4           2           0  ...           2
191           3           2           3           0  ...           2
192           0           4           2           0  ...           1
193           4           4           2           0  ...           2
194           3           3           1           0  ...           0
195           2           4           0           0  ...           1
196           4           4           2           0  ...           2
197           2           4           2           0  ...           3
198           0           3           2           0  ...           2

     Attribute13  Attribute14  Attribute15  Attribute16  Attribute17  \
0             48            0            0            2            1
1             24            2            1            2            2
2             35            2            1            2            2
```

3	24	2	0	1	0
4	24	2	1	1	2
5	26	2	1	1	1
6	65	2	1	4	2
7	55	0	2	1	3
8	26	2	0	2	0
9	26	2	1	1	2
10	28	0	1	1	1
11	24	2	1	2	2
12	54	2	1	1	2
13	46	2	2	1	2
14	54	2	0	1	2
15	62	2	1	1	2
16	24	2	0	2	2
17	43	2	1	1	3
18	26	0	1	1	2
19	27	2	1	2	2
20	24	2	1	1	2
21	41	0	1	1	2
22	47	2	1	2	1
23	35	2	1	2	3
24	30	2	1	2	2
25	33	0	0	3	2
26	36	1	1	2	2
27	47	2	2	1	2
28	38	2	2	1	2
29	44	2	1	2	2
..	...	...	...	...	...
169	22	2	1	1	2
170	43	2	1	1	1
171	29	2	0	2	0
172	36	2	0	1	2
173	33	2	1	1	2
174	57	2	1	1	1
175	64	2	1	1	2
176	42	2	1	2	2
177	47	2	1	2	1
178	25	2	0	1	2
179	49	2	1	1	2
180	33	0	0	1	3
181	28	0	1	1	3
182	26	2	1	1	2
183	30	2	1	2	1
184	25	2	0	2	2
185	33	2	1	2	2
186	64	2	1	1	2
187	29	2	2	1	3
188	48	2	1	2	1

189	37	2	1	2	1
190	34	0	1	1	1
191	23	2	0	2	1
192	30	2	1	1	3
193	50	2	1	1	2
194	31	2	1	1	1
195	40	2	1	1	3
196	38	2	1	1	2
197	23	2	2	1	2
198	27	2	1	1	2

	Attribute18	Attribute19	Attribute20	Id
0	1	1	0	10902
1	1	0	0	10903
2	1	0	0	10904
3	1	0	0	10905
4	1	1	0	10906
5	1	0	1	10907
6	1	0	0	10908
7	1	1	0	10909
8	1	0	0	10910
9	1	1	0	10911
10	1	0	0	10912
11	1	1	0	10913
12	1	0	0	10914
13	2	0	0	10915
14	1	0	0	10916
15	1	0	0	10917
16	1	1	0	10918
17	1	0	0	10919
18	1	0	0	10920
19	1	1	0	10921
20	1	0	0	10922
21	2	0	0	10923
22	1	0	0	10924
23	1	1	0	10925
24	1	1	0	10926
25	1	1	0	10927
26	2	0	0	10928
27	1	0	0	10929
28	2	1	0	10930
29	2	1	0	10931
..	...	...	...	...
169	1	0	0	11071
170	2	0	0	11072
171	1	0	0	11073
172	1	0	0	11074
173	1	1	0	11075

```
174            1            0            0  11076
175            1            0            0  11077
176            1            0            0  11078
177            2            0            0  11079
178            1            0            0  11080
179            1            0            0  11081
180            1            1            0  11082
181            1            1            0  11083
182            2            0            0  11084
183            1            0            0  11085
184            1            0            0  11086
185            1            0            0  11087
186            1            0            0  11088
187            1            1            0  11089
188            1            0            0  11090
189            2            0            0  11091
190            2            0            0  11092
191            1            0            0  11093
192            1            1            0  11094
193            1            1            0  11095
194            1            0            0  11096
195            1            1            0  11097
196            1            0            0  11098
197            1            1            0  11099
198            1            0            0  11100

[199 rows x 21 columns]
```

```
In [21]: proccessedData_train= proccessedData_train.drop(['Label'],axis=1) #afairoume to column
         print(proccessedData_train.columns)

         print(proccessedData_test.shape, proccessedData_train.shape)
         print(proccessedData_test.head(), proccessedData_train.head())
```

```
Index(['Attribute1', 'Attribute2', 'Attribute3', 'Attribute4', 'Attribute5',
       'Attribute6', 'Attribute7', 'Attribute8', 'Attribute9', 'Attribute10',
       'Attribute11', 'Attribute12', 'Attribute13', 'Attribute14',
       'Attribute15', 'Attribute16', 'Attribute17', 'Attribute18',
       'Attribute19', 'Attribute20', 'Id'],
      dtype='object')
(199, 21) (800, 21)
   Attribute1  Attribute2  Attribute3  Attribute4  Attribute5  Attribute6  \
0           1          18           4           4        1795           0
1           0          20           4           3        4272           0
2           3          12           4           4         976           4
3           1          12           2           0        7472           4
4           0          36           2           0        9271           0
```

```
   Attribute7  Attribute8  Attribute9  Attribute10  ...    Attribute12  \
0           4           3           1            2  ...              0
1           4           1           1            0  ...              1
2           4           4           2            0  ...              2
3           0           1           1            0  ...              0
4           3           2           2            0  ...              2

   Attribute13  Attribute14  Attribute15  Attribute16  Attribute17  \
0           48            0            0            2            1
1           24            2            1            2            2
2           35            2            1            2            2
3           24            2            0            1            0
4           24            2            1            1            2

   Attribute18  Attribute19  Attribute20     Id
0            1            1            0  10902
1            1            0            0  10903
2            1            0            0  10904
3            1            0            0  10905
4            1            1            0  10906

[5 rows x 21 columns]    Attribute1  Attribute2  Attribute3  Attribute4  Attribute5  Attribute6
0           0           6           4           4        1169           4
1           1          48           2           4        5951           0
2           3          12           4           7        2096           0
3           0          42           2           3        7882           0
4           0          24           3           0        4870           0

   Attribute7  Attribute8  Attribute9  Attribute10  ...    Attribute12  \
0           4           4           2            0  ...              0
1           2           2           1            0  ...              0
2           3           2           2            0  ...              0
3           3           2           2            2  ...              1
4           2           3           2            0  ...              3

   Attribute13  Attribute14  Attribute15  Attribute16  Attribute17  \
0           67            2            1            2            2
1           22            2            1            1            2
2           49            2            1            1            1
3           45            2            2            1            2
4           53            2            2            2            2

   Attribute18  Attribute19  Attribute20     Id
0            1            1            0  10101
1            1            0            0  10102
2            2            0            0  10103
3            2            0            0  10104
```

```
4            2            0            0  10105

[5 rows x 21 columns]
```

1.2 Support Vector Machines (SVM) Classification

```
In [22]: RANDOM_STATE = 123
         classifier = svm.LinearSVC(multi_class = "ovr",random_state=RANDOM_STATE)
         classifier.fit(proccessedData_train,target)

         predicted = classifier.predict(proccessedData_test)

         print("LinearSVC with linear kernel and c=0.2:")

         print(predicted)
```

```
LinearSVC with linear kernel and c=0.2:
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
```

```
In [23]: average_acc= cross_validate(classifier,proccessedData_train,target)
         EvaluationMetric['SVM'].append(average_acc)
```

```
Fold 1


Accuracy:  0.75
Fold 2


Accuracy:  0.775
Fold 3


Accuracy:  0.6375
Fold 4


Accuracy:  0.75
Fold 5


Accuracy:  0.7375
```

```
Fold 6


Accuracy:  0.7
Fold 7


Accuracy:  0.65
Fold 8


Accuracy:  0.5875
Fold 9


Accuracy:  0.75
Fold 10


Accuracy:  0.675
Average accuracy =  0.70125
```

1.3 Random Forest (RF) Classification

```
In [24]: RANDOM_STATE = 123

         rndf = RandomForestClassifier(warm_start=True, oob_score=True, max_features="sqrt", ran
         rndf.set_params(n_estimators=30)
         rndf.fit(proccessedData_train,target)

         predicted = rndf.predict(proccessedData_test)
         print(predicted)
         # for x in range(10):
         #     print(test_data['Title'][x] + "---->" + categories[predicted[x]])
```

```
[1 2 1 1 2 1 1 2 2 1 1 2 2 2 2 1 1 2 2 1 1 1 1 1 1 1 1 2 1 1 2 2 1 1 2 1 1
 1 1 2 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 2 1 1 1 2 1 1 1 2 1 1 2 1 1 1 1 1 1
 1 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1
 1 1 2 2 1 1 2 1 1 1 2 1 2 2 2 2 1 2 2 1 1 1 2 2 1 1 2 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 2 1 1 1 1 1 2 1 1 1 1 1 2
 1 1 1 1 1 1 1 2 1 1 1 1 2 2]
```

```
In [25]: average_acc = cross_validate(rndf,proccessedData_train,target)
         EvaluationMetric['Random Forest'].append(average_acc)
```

```
Fold 1
```

```
Accuracy:  1.0
Fold 2


Accuracy:  1.0
Fold 3


Accuracy:  1.0
Fold 4


Accuracy:  1.0
Fold 5


Accuracy:  1.0
Fold 6


Accuracy:  1.0
Fold 7


Accuracy:  1.0
Fold 8


Accuracy:  0.9875
Fold 9


Accuracy:  1.0
Fold 10


Accuracy:  1.0
Average accuracy =  0.99875


/home/marinos/.local/lib/python3.5/site-packages/sklearn/ensemble/forest.py:303: UserWarning: Wa
  warn("Warm-start fitting without increasing n_estimators does not "
```

1.4 Naive Bayes (NB) Classification

We use Multinomial Naive Bayes for our implementation:

```
In [26]: mnb = MultinomialNB().fit(proccessedData_train, target)

         predicted = mnb.predict(proccessedData_test)

         print(predicted)
```

```
[1 2 1 2 2 1 1 2 1 1 1 2 1 2 2 1 1 2 1 1 1 1 1 2 1 2 2 2 2 1 1 2 2 1 1 1 1
 1 1 2 1 1 1 1 2 2 1 1 1 1 2 1 1 2 1 1 1 2 2 2 1 1 2 1 1 1 1 2 1 2 1 1 1 2
 1 1 2 1 2 2 2 1 1 2 1 1 2 2 2 1 1 1 2 1 2 1 1 1 2 1 1 2 1 1 1 2 2 2 1 2 2
 1 1 1 2 1 2 1 1 1 2 1 1 2 1 1 2 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 2 1 1 1
 2 2 1 2 2 1 1 2 1 2 1 1 1 2 1 1 1 1 2 2 2 1 2 1 2 1 1 1 1 1 1 2 2 1 2 1 1
 2 1 2 1 2 1 1 2 1 1 2 1 1 2]
```

```
In [27]: average_acc = cross_validate(mnb,proccessedData_train,target)
         EvaluationMetric['Naive Bayes'].append(average_acc)
```

```
Fold 1


Accuracy:  0.6375
Fold 2


Accuracy:  0.675
Fold 3


Accuracy:  0.6875
Fold 4


Accuracy:  0.575
Fold 5


Accuracy:  0.6625
Fold 6


Accuracy:  0.625
Fold 7


Accuracy:  0.625
Fold 8
```

```
Accuracy:  0.4625
Fold 9


Accuracy:  0.575
Fold 10


Accuracy:  0.6
Average accuracy =  0.6125
```

1.5 10-fold Cross Validation

We evaluate and store the performance of each of the above methods using 10-fold Cross Validation with accuracy as a meter.

```
In [28]: EvaluationMetric_10fold = pd.DataFrame(data=EvaluationMetric)
         EvaluationMetric_10fold = EvaluationMetric_10fold.ix[::, ['Statistic Measure','Naive Ba

         EvaluationMetric_10fold.to_csv(path_or_buf='EvaluationMetric_10fold.csv', sep='\t', ind
         EvaluationMetric_10fold

Out[28]:   Statistic Measure  Naive Bayes  Random Forest      SVM
         0          Accuracy       0.6125        0.99875  0.70125
```

From the above, we observe that the Random Forest (RF) classification method is much better, in terms of accuracy, than Naive Bayes and Support Vector Machines methods. It is almost always right (~99% accuracy).