In [1]:

```
import sys
!{sys.executable} -m pip install xgboost
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from xgboost import XGBClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier, VotingClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.datasets import make_classification
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score
```

```
Requirement already satisfied: xgboost in /anaconda3/lib/python3.7/s
ite-packages (0.90)
Requirement already satisfied: numpy in /anaconda3/lib/python3.7/sit
e-packages (from xgboost) (1.16.2)
Requirement already satisfied: scipy in /anaconda3/lib/python3.7/sit
e-packages (from xgboost) (1.2.1)
```

In [2]:

```
df = pd.read_csv('train.csv')
columns = list(df.columns)
non_medical = columns[0:79]
medical = columns[79:127]

med  = df[medical]
med = med.sum(axis = 1)

df['Product_Info_2'] = pd.Categorical(df['Product_Info_2'])
dfDummies = pd.get_dummies(df['Product_Info_2'], prefix = 'P2')

train = df[non_medical]
train = train.drop(columns = 'Product_Info_2')
train['Response'] = df['Response']
train['Keyword'] = med
train = pd.concat([train, dfDummies], axis=1)
train.head
```

Out[2]:

```
<bound method NDFrame.head of           Id  Product_Info_1  Product_
Info_3  Product_Info_4  Product_Info_5  \
0          2               1              10        0.076923
2
```

2				
1	5	1	26	0.076923
2				
2	6	1	26	0.076923
2				
3	7	1	10	0.487179
2				
4	8	1	26	0.230769
2				
5	10	1	26	0.230769
3				
6	11	1	10	0.166194
2				
7	14	1	26	0.076923
2				
8	15	1	26	0.230769
2				
9	16	1	21	0.076923
2				
10	17	1	26	0.128205
2				
11	18	1	26	0.230769
2				
12	19	1	26	0.102564
2				
13	20	2	26	0.487179
2				
14	22	1	26	0.487179
2				
15	23	1	26	0.000000
2				
16	24	2	26	0.487179
2				
17	25	1	26	0.384615
2				
18	26	1	26	0.076923
2				
19	27	1	26	0.487179
2				
20	29	1	26	0.435897
2				
21	31	1	26	1.000000
2				
22	32	1	26	0.230769
2				
23	33	1	26	0.179487
2				
24	34	1	26	0.487179
2				
25	35	1	26	0.230769
2				
26	37	1	26	1.000000
2				
27	39	1	26	0.230769

					2
28	40	1	26	0.487179	2
29	41	1	26	1.000000	2
...	...	...	...	...	...
59351	79115	1	26	0.000000	2
59352	79116	1	10	0.230769	2
59353	79117	1	26	0.589744	2
59354	79118	1	26	0.487179	2
59355	79119	1	26	0.230769	2
59356	79120	1	10	0.076923	2
59357	79121	1	26	1.000000	2
59358	79122	1	26	0.282051	2
59359	79123	1	26	0.230769	2
59360	79124	1	26	1.000000	2
59361	79126	1	26	0.230769	2
59362	79127	1	26	0.230769	2
59363	79128	1	4	0.076923	2
59364	79130	1	26	0.076923	2
59365	79131	1	29	0.076923	2
59366	79132	1	26	0.282051	2
59367	79133	1	26	0.179487	2
59368	79134	1	26	0.230769	2
59369	79135	1	26	0.179487	2
59370	79136	1	26	0.230769	2
59371	79137	1	26	0.487179	2
59372	79138	1	26	0.487179	2
59373	79139	2	29	0.487179	2

```
59374  79140                1               26         0.307692

2
59375  79141                1               26         0.076923
2
59376  79142                1               10         0.230769
2
59377  79143                1               26         0.230769
2
59378  79144                1               26         0.076923
2
59379  79145                1               10         0.230769
2
59380  79146                1               26         0.076923
2

       Product_Info_6  Product_Info_7   Ins_Age        Ht        Wt
...  \
0                   1               1  0.641791  0.581818  0.148536
...
1                   3               1  0.059701  0.600000  0.131799
...
2                   3               1  0.029851  0.745455  0.288703
...
3                   3               1  0.164179  0.672727  0.205021
...
4                   3               1  0.417910  0.654545  0.234310
...
5                   1               1  0.507463  0.836364  0.299163
...
6                   3               1  0.373134  0.581818  0.173640
...
7                   3               1  0.611940  0.781818  0.403766
...
8                   3               1  0.522388  0.618182  0.184100
...
9                   3               1  0.552239  0.600000  0.284519
...
10                  3               1  0.537313  0.690909  0.309623
...
11                  3               1  0.298507  0.690909  0.271967
...
12                  3               1  0.567164  0.618182  0.163180
...
13                  3               1  0.223881  0.781818  0.361925
...
14                  3               1  0.328358  0.636364  0.142259
...
15                  3               1  0.626866  0.672727  0.330544
...
16                  3               1  0.208955  0.745455  0.246862
...
17                  3               1  0.268657  0.636364  0.228033
...
```

18	3	1	0.388060	0.781818	0.309623
...					
19	3	1	0.223881	0.600000	0.138075
...					
20	3	1	0.388060	0.745455	0.246862
...					
21	1	1	0.537313	0.709091	0.370293
...					
22	3	1	0.179104	0.800000	0.539749
...					
23	3	1	0.164179	0.745455	0.288703
...					
24	1	1	0.164179	0.818182	0.435146
...					
25	3	1	0.268657	0.781818	0.368201
...					
26	3	1	0.507463	0.654545	0.299163
...					
27	3	1	0.134328	0.763636	0.215481
...					
28	3	1	0.492537	0.618182	0.276151
...					
29	3	1	0.582090	0.654545	0.278243
...					
...	...	...	...	...	...
...					
59351	3	1	0.134328	0.781818	0.351464
...					
59352	3	1	0.358209	0.618182	0.246862
...					
59353	1	1	0.179104	0.781818	0.382845
...					
59354	1	1	0.402985	0.763636	0.341004
...					
59355	3	1	0.223881	0.745455	0.361925
...					
59356	3	1	0.522388	0.600000	0.299163
...					
59357	1	3	0.582090	0.781818	0.351464
...					
59358	3	1	0.238806	0.727273	0.372385
...					
59359	3	1	0.447761	0.781818	0.424686
...					
59360	3	1	0.194030	0.654545	0.146444
...					
59361	1	1	0.268657	0.727273	0.267782
...					
59362	3	1	0.253731	0.781818	0.351464
...					
59363	3	1	0.746269	0.563636	0.205021
...					
59364	3	1	0.552239	0.727273	0.177824

```
...

59365                3                     1  0.641791  0.709091  0.284519
...
59366                3                     1  0.582090  0.781818  0.320084
...
59367                3                     3  0.373134  0.600000  0.320084
...
59368                1                     1  0.417910  0.727273  0.299163
...
59369                3                     1  0.611940  0.745455  0.451883
...
59370                3                     1  0.238806  0.763636  0.330544
...
59371                1                     1  0.537313  0.709091  0.343096
...
59372                3                     1  0.477612  0.763636  0.305439
...
59373                3                     1  0.208955  0.800000  0.257322
...
59374                3                     1  0.164179  0.690909  0.288703
...
59375                3                     1  0.477612  0.654545  0.271967
...
59376                3                     1  0.074627  0.709091  0.320084
...
59377                3                     1  0.432836  0.800000  0.403766
...
59378                3                     1  0.104478  0.745455  0.246862
...
59379                3                     1  0.507463  0.690909  0.276151
...
59380                3                     1  0.447761  0.781818  0.382845
...

       P2_B2  P2_C1  P2_C2  P2_C3  P2_C4  P2_D1  P2_D2  P2_D3  P2_D4
P2_E1
0          0      0      0      0      0      0      0      1      0
0
1          0      0      0      0      0      0      0      0      0
0
2          0      0      0      0      0      0      0      0      0
1
3          0      0      0      0      0      0      0      0      1
0
4          0      0      0      0      0      0      1      0      0
0
5          0      0      0      0      0      0      1      0      0
0
6          0      0      0      0      0      0      0      0      0
0
7          0      0      0      0      0      0      1      0      0
0
8          0      0      0      0      0      0      0      1      0
```

										0
9	0	0	0	0	0	0	0	0	0	1
10	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	1	0
12	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	1	0	0	0	0
14	0	0	0	0	0	0	0	0	1	0
15	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	1	0
17	0	0	0	0	0	0	0	1	0	0
18	0	0	0	0	0	0	0	1	0	0
19	0	0	0	0	0	0	0	0	1	0
20	0	0	0	0	0	0	1	0	0	0
21	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	1	0
23	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	1	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	1	0	0
28	0	0	0	0	0	0	0	0	1	0
29	0	0	0	0	0	0	0	1	0	0
...	...	...	...	...	...	...	...	...	...	...
59351	0	0	0	0	0	0	0	0	0	0
59352	0	0	0	0	0	0	1	0	0	0
59353	0	0	0	0	0	0	0	0	1	0
59354	0	0	0	0	0	0	1	0	0	0

59355	0	0	0	0	0	0	0	1	0	0
59356	0	0	0	0	0	0	0	1	0	0
59357	0	0	0	0	0	0	0	1	0	0
59358	0	0	0	0	0	0	0	0	1	0
59359	0	0	0	0	0	0	0	1	0	0
59360	0	0	0	0	0	0	0	0	1	0
59361	0	0	0	0	0	0	0	0	0	0
59362	0	0	0	0	0	0	0	0	1	0
59363	0	0	0	0	0	0	1	0	0	0
59364	0	0	0	0	0	0	1	0	0	0
59365	0	0	0	0	0	1	0	0	0	0
59366	0	0	0	0	0	1	0	0	0	0
59367	0	0	0	0	0	0	0	0	0	1
59368	0	0	0	0	0	0	0	0	1	0
59369	0	0	0	0	0	1	0	0	0	0
59370	0	0	0	0	0	0	0	1	0	0
59371	0	0	0	0	0	0	0	1	0	0
59372	0	0	0	0	0	0	0	1	0	0
59373	0	0	0	0	0	0	0	0	1	0
59374	0	0	0	0	0	0	0	0	1	0
59375	0	1	0	0	0	0	0	0	0	0
59376	0	0	0	0	0	1	0	0	0	0
59377	0	0	0	0	0	0	0	1	0	0
59378	0	0	0	0	0	0	0	0	0	1
59379	0	0	0	0	0	0	1	0	0	0
59380	0	0	0	0	0	0	0	0	0	0

```
[59381 rows x 99 columns]>
```

In [3]:

```
train.Keyword
```

Out[3]:

```
0        0
1        0
2        0
3        1
4        0
5        2
6        0
7        0
8        1
9        2
10       4
11       1
12       1
13       1
14       2
15       3
16       1
17       0
18       1
19       0
20       0
21       0
22       0
23       0
24       0
25       0
26       1
27       0
28       2
29       2
        ..
59351    0
59352    1
59353    3
59354    1
59355    1
59356    2
59357    0
59358    0
59359    1
59360    1
59361    0
59362    0
59363    0
59364    1
59365    0
59366    2
```

```
59367    1
59368    0
59369    6
59370    0
59371    1
59372    4
59373    0
59374    0
59375    1
59376    0
59377    0
59378    1
59379    2
59380    0
Name: Keyword, Length: 59381, dtype: int64
```

In [4]:

```
df = pd.read_csv('test.csv')
columns = list(df.columns)
non_medical = columns[0:79]
medical = columns[79:127]

med  = df[medical]
med = med.sum(axis = 1)

df['Product_Info_2'] = pd.Categorical(df['Product_Info_2'])
dfDummies = pd.get_dummies(df['Product_Info_2'], prefix = 'P2')

test = df[non_medical]
test = test.drop(columns = 'Product_Info_2')
test['Keyword'] = med
test = pd.concat([test, dfDummies], axis=1)
```

In [5]:

```
clf = LogisticRegression(solver='lbfgs', multi_class='multinomial',
                           random_state=1, max_iter = 100)
clf2 = RandomForestClassifier(n_estimators=50, random_state=1)
clf3 = GaussianNB()
clf4 = AdaBoostClassifier(n_estimators=100, random_state=0)
clf5 = DecisionTreeClassifier(max_depth=20, min_samples_split=20,
     random_state=0)
clf6 = ExtraTreesClassifier(n_estimators=100, max_depth=20,
     min_samples_split=20, random_state=0)
clf7 = XGBClassifier()
```

In [6]:

```
clf1 = VotingClassifier(estimators=[
         ('lr', clf), ('rf', clf2), ('gnb', clf3), ('ada', clf4), ('5', clf5), (
'6', clf6), ('7', clf7)], voting='hard')
```

In [7]:

```
X = train
X = X.drop(columns  = 'Id')
y = X['Response']
X = X.fillna(-1)

X = X.drop(columns = 'Response')

X_train, X_test, y_train, y_test = train_test_split(X.values, y.values, test_siz
e=0.3, random_state=0)

scaler = StandardScaler()
scaler.fit(X)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

In [8]:

```
clf1 = clf1.fit(X_train, y_train)
print(clf1.score(X_train,y_train))
```

```
/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic
.py:947: ConvergenceWarning: lbfgs failed to converge. Increase the
number of iterations.
  "of iterations.", ConvergenceWarning)

0.7230909878265891
```

In [9]:

```
predictions_test = clf1.predict(X_test)
print(clf1.score(X_test, y_test))
```

```
0.5551501543642997
```

In [10]:

```
from sklearn.metrics import classification_report, confusion_matrix
CM = confusion_matrix(y_test, predictions_test)
print(CM)
print(classification_report(y_test, predictions_test))
```

```
[[ 406  313   20   39  133  347  159  406]
 [ 211  561   19   40  195  376  172  389]
 [  27   40   91   73   18   53    3   10]
 [  27   10   21  244    0   47    4   57]
 [  73  222    1    0  754  292   72  155]
 [ 161  225    0    8  141 1717  353  743]
 [  78   64    0    2   18  490  830 1002]
 [  33   29    0    2   18  306  228 5287]]
              precision    recall  f1-score   support

           1       0.40      0.22      0.29      1823
           2       0.38      0.29      0.33      1963
           3       0.60      0.29      0.39       315
           4       0.60      0.60      0.60       410
           5       0.59      0.48      0.53      1569
           6       0.47      0.51      0.49      3348
           7       0.46      0.33      0.39      2484
           8       0.66      0.90      0.76      5903

    accuracy                           0.56     17815
   macro avg       0.52      0.45      0.47     17815
weighted avg       0.53      0.56      0.53     17815
```

In [11]:

```
test_noID = test.drop(columns = ['Id'])
test_noID = test_noID.fillna(-1)

scaler = StandardScaler()
scaler.fit(test_noID)
test_noID = scaler.transform(test_noID)
predictions_test = clf1.predict(test_noID)

test['Response'] = predictions_test
submission = test[['Id', 'Response']]
submission.set_index('Id', inplace = True)
submission.to_csv('Submission.csv', float_format='%.0f')
print(submission)
```

```
       Response
Id
1             7
3             8
4             6
9             8
```

12	8
13	8
21	8
28	8
30	7
36	8
38	8
43	8
45	4
48	8
50	4
51	8
54	7
55	8
59	8
62	1
63	8
66	8
69	8
82	8
83	6
84	6
86	8
89	8
90	2
92	8
...	...
79004	7
79007	8
79020	6
79022	1
79027	1
79028	8
79031	8
79035	1
79038	8
79047	8
79048	5
79051	6
79054	5
79060	6
79064	8
79065	5
79067	8
79071	5
79072	6
79073	8
79080	6
79083	2
79084	8
79085	6
79089	8
79093	8

```
79099          8
79102          1
79125          2
79129          6

[19765 rows x 1 columns]
```

In [ ]:

```

```

In [ ]:

```

```