Homework 5

Nicholas Antonov & Patrick Grasso

December 18, 2016

Contents

1 Decision Trees

Our baseline for deciding whether or not our results were acceptable originate from our first attempt at classifying with a decision tree, in which the resulting tree consisted of a single node (a stump) with the value "N". It occurred to us that you can achieve about **78% accuracy just by classifying every case as not serious**. This is known as the prior distribution, which we used to define a baseline for performance. Any classifier should do better than simply picking "N". (Note: recall and precision for this method are both 0).

1.1 Scikit's `DecisionTreeClassifier`

We tried two different methods of generating decision trees. The first method utilizes Scikit's `DecisionTreeClassifier` and is trained on a tf-idf matrix (rows are the documents, columns are words selected by the vectorizer). Before vectorizing with tf-idf, stop words were removed and all words were stemmed so that similar words like `cats` and `cat` were considered the same.

With an 80/20 train/test split, this classifier achieved around 82-83% accuracy on average, which is only about 4% above the baseline. However, accuracy is not the most important or relevant metric in this scenario. **Precision**, or the number of true positives / (true positives + false positives), indicates how efficient our classifier is with respect to resources. A higher precision means that we are less likely to misclassify non-serious cases as serious. Another metric, **recall**, is the number of true positives / (true positives + false negatives). A higher recall means that we are less likely to miss a serious case and claim it to be non-serious. **Precision** is also the metric used for ROC charts.

Scikit's `DecisionTreeClassifier` had an average recall of 0.256 and an average precision of 0.864. This means that Scikit's decision tree is more likely to miss serious cases, but it does not waste resources with non-serious cases.

1.1.1 Output for the Scikit decision tree

Below is the output of the Scikit decision tree classifier. The top 10 most significant terms are printed with their corresponding relevance. The words appear cut off because the stemming process truncated all words to their common root (adverse, adversity, adversely -> advers; advise, advisory -> advis).

```
$ python decision-tree.py

Train : 13827 (80.00%)
Test  : 3457 (20.00%)
Removing stop words/stemming
Vectorizing with TfidfVectorizer
Vectorization complete. Classifying...

Printing the top 10 significant features
advers                        : 0.4445046663593824
er                            : 0.1660336983992121
hospit                        : 0.11770446059244367
report                        : 0.0980937934053291
emerg                         : 0.06459727209523931
test                          : 0.030503082385091213
went                          : 0.016911883657538047
patient experienc             : 0.012283723770826732
day                           : 0.00913341776104604
mother                        : 0.007675118925593991

tree accuracy  : 0.831935203934
tree recall    : 0.25641025641
tree precision : 0.863636363636
prior accuracy : 0.785652299682
prior recall   : 0.0
prior precision: 0.0

~ confusion ~
reference:
[['TN' 'FP']
 ['FN' 'TP']]

confusion [DecisionTreeClassifier]:
[[2686   30]
 [ 551  190]]
```

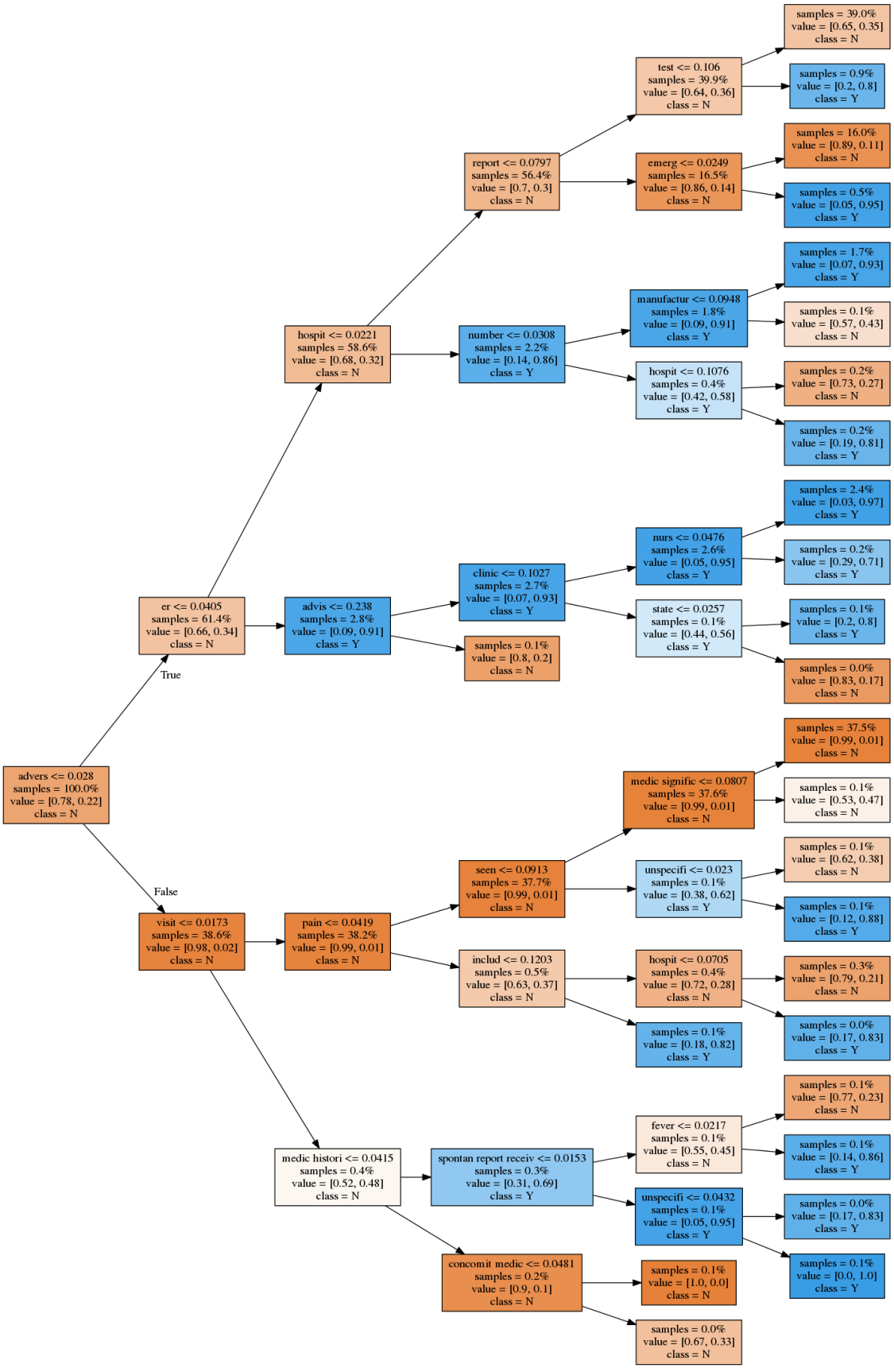

Figure 1: Graphical representation of the Scikit decision tree

1.2 NLTK's `DecisionTreeClassifier`

Second, we tried using NLTK's own `DecisionTreeClassifier`. This classifier took significantly longer to train, although it performed better. NLTK's decision tree achieved an accuracy of 0.847, a recall of 0.388, and a precision of 0.796. This means that the NLTK version of this classifier will catch more serious cases, but at the cost of misclassifying more non-serious cases (not as resource efficient).

1.2.1 Pseudocode and output for the NLTK decision tree

```
$ python nltk-tree.py

Train : 13827 (80.00%)
Test  : 3457 (20.00%)
Vectorizing with CountVectorizer
Vectorization complete. Classifying...
if er == False:
  if emergency == False:
    if hospital == False:
      if seen == False:
        if testing == False:
          if prescribed == False: return 'Y'
          if prescribed == True: return 'Y'
        if testing == True:
          if years == False: return 'Y'
          if years == True: return 'N'
      if seen == True:
        if initial == False:
          if felt == False: return 'N'
          if felt == True: return 'N'
        if initial == True:
          if nausea == False: return 'N'
          if nausea == True: return 'Y'
    if hospital == True:
      if improperly == False:
        if date outcome == False:
          if condition medical == False: return 'Y'
          if condition medical == True: return 'N'
        if date outcome == True:
          if started == False: return 'N'
          if started == True: return 'Y'
      if improperly == True: return 'N'
  if emergency == True:
    if redness swelling == False:
      if mom == False:
        if adverse effects == False:
          if mmr == False: return 'Y'
          if mmr == True: return 'Y'
```

```
        if adverse effects == True: return 'N'
      if mom == True:
        if condition == False: return 'N'
        if condition == True: return 'Y'
    if redness swelling == True:
      if vaccinated == False: return 'N'
      if vaccinated == True: return 'Y'
if er == True:
  if initial spontaneous report == False:
    if administered dose == False:
      if date 08 == False: return 'Y'
      if date 08 == True: return 'N'
    if administered dose == True: return 'N'
  if initial spontaneous report == True: return 'N'


tree accuracy  : 0.846977147816
tree recall    : 0.387617765814
tree precision : 0.795580110497
prior accuracy : 0.785073763379
prior recall   : 0.0
prior precision: 0.0

~ confusion ~
reference:
[['TN' 'FP']
 ['FN' 'TP']]

confusion [DecisionTreeClassifier]:
[[2640   74]
 [ 455  288]]

confusion [DummyClassifier]:
[[2714    0]
 [ 743    0]]
```

2 Text Topics

Another way of classifying documents is by extracting relevant topics for each document and running a classifier on which topics correspond to certain classes. Two of the more common ways of extracting topics from text include Latent Dirichlet Allocation (LDA) and Latent Semantic Indexing (LSI). Both methods were used in order to extract topics from the VAERS dataset, but only one is reported here, as the other did not perform well. A Latent Semantic Indexing model was trained with the tf-idf vectorization of 80% of the symptom texts. The other 20% were used to test the model.

Then, each document was assigned a relevance score for each topic in the model, producing a matrix in which the rows represent documents and the columns topics. A random forest classifier with 10 weak learners and an unrestricted depth was then trained on this topic-relevance data set

to classify symptom texts as serious (Y or N).

2.1 LDA topic extraction output

A sample of words from each topic is printed to provide a general idea or sense of the topic. This method obtained an accuracy of 0.817, a recall of 0.339, and a precision of 0.615. This precision is relatively low compared to the other methods, meaning that it is not very resource-efficient. Its recall is between Scikit's decision tree and NLTK's decision tree, which is indicative of a high testing variance.

Nevertheless, topic extraction appears to perform worse than decision trees for recall and precision. This may be due to the fact that the decision tree classifiers were trained with n-grams up to 3, identifying common phrases in various documents as opposed to individual words. Another factor that might potentially affect the outcome is the number of topics chosen and the classifier used for topic extraction beyond LDA (right now we're using a random forest, but other ensemble classification methods such as boosting can be used).

```
$ python topic-node.py

Train : 13827 (80.00%)
Test  : 3457 (20.00%)
Removing stop words/stemming
Creating tf-idf models
Creating LSI model
Topics:
(0, ['report', 'dose', 'patient', 'unknown', 'medic'])
(1, ['red', 'pain', 'inject', 'arm', 'site'])
(2, ['2014.', 'fluvirin', 'number', 'batch', 'oct'])
(3, ['excurs', 'temperatur', 'degre', 'hour', 'minut'])
(4, ['red', 'allergy/drug', 'pqc', 'swell', 'recombivax'])
(5, ['site', 'inject', 'red', 'pain', 'swell'])
(6, ['compon', 'menveo', 'zostavax', 'fluvirin', 'conjug'])
(7, ['pain', 'rash', 'arm', 'temperatur', 'none'])
(8, ['rash', 'none', 'pain', 'state', 'fever'])
(9, ['none', 'flumist', 'rash', '67', 'servic'])
(10, ['none', 'pain', 'rotateq', 'oral', 'zostavax'])
(11, ['merck', 'flumist', 'proquad', 'gardasil', 'none'])
(12, ['gardasil', 'flumist', '9', 'quadrival', 'intranas'])
(13, ['gardasil', 'pt', 'flumist', 'rash', 'pain'])
(14, ['fever', 'arm', 'rash', 'inject', 'merck'])
(15, ['rash', 'pt', 'zostavax', 'gardasil', 'rotateq'])
(16, ['hb', 'recombivax', 'fever', 'qualiti', 'complaint'])
(17, ['none', 'state', '2015', 'unspecifi', 'pain'])
(18, ['unspecifi', 'inject', 'fever', 'red', 'swollen'])
(19, ['pt', 'swollen', 'flumist', 'red', 'fever'])
(20, ['pt', 'pertin', 'drug', 'swell', 'rotateq'])
(21, ['swell', 'swollen', 'zostavax', 'warm', 'itch'])
(22, ['swell', 'inject', 'swollen', 'fever', 'zostavax'])
(23, ['expir', 'worker', 'healthcar', 'expiri', 'public'])
```

```
(24, ['pt', 'arm', 'area', 'sore', "''"])
(25, ['pain', 'pt', 'hive', 'sore', 'itch'])
(26, ['pneumovax', '23', 'zostavax', 'ii', 'hive'])
(27, ['zostavax', 'assist', 'vaqta', 'expect', 'oral'])
(28, ["''", '``', 'pain', 'proquad', 'arm'])
(29, ['assist', 'certifi', 'expir', 'follow-up', 'pharmacist'])
(30, ['pain', 'itch', 'arm', 'sore', 'touch'])
(31, ['hive', 'pneumovax', 'itch', 'swell', '23'])
(32, ["''", '``', 'ii', 'm-m-r', 'storag'])
(33, ['hive', 'ii', 'varivax', 'vaqta', 'm-m-r'])
(34, ['anaphylaxi', 'bexsero', 'itch', 'state', 'case'])
(35, ['varivax', 'degre', "''", 'pneumovax', '``'])
(36, ['itch', 'hive', 'anaphylaxi', 'touch', 'area'])
(37, ['sore', 'warm', 'touch', 'hive', 'itch'])
(38, ['sore', 'vaqta', 'seizur', 'fever', 'swollen'])
(39, ['itch', 'hive', "''", '``', 'minut'])
(40, ['sore', 'swollen', 'arm', 'fever', 'anaphylaxi'])
(41, ['unspecifi', 'ii', 'bexsero', 'anaphylaxi', 'strength'])
(42, ['seizur', 'fever', 'itch', 'swollen', 'dizzi'])
(43, ["''", 'seizur', 'celsiu', 'varivax', '``'])
(44, ['day', 'varivax', 'erythema', 'left', 'vaqta'])
(45, ['area', 'multipl', 'itch', 'vomit', 'bexsero'])
(46, ['right', 'regist', 'regard', 'anaphylaxi', 'ii'])
(47, ['area', 'vaqta', 'ii', 'touch', 'm-m-r'])
(48, ['administ', 'right', 'assist', 'inadvert', 'pharmacist'])
(49, ['vomit', 'right', 'symptom', 'ach', 'nausea'])
(50, ['left', 'unspecifi', 'vomit', 'right', 'area'])
(51, ['seizur', 'day', 'sore', 'administ', 'provid'])
(52, ['vomit', 'area', 'left', 'swollen', 'given'])
(53, ['right', 'sore', '2012', 'men', 'area'])
(54, ['given', 'bexsero', 'vomit', 'headach', 'ach'])
(55, ['right', 'left', 'given', 'fever', 'shoulder'])
(56, ['given', 'bexsero', 'varivax', 'left', 'expir'])
(57, ['swollen', 'area', 'given', 'arm', 'shoulder'])
(58, ['multipl', 'bexsero', 'area', 'sore', 'unspecifi'])
(59, ['right', 'shingl', 'fever', 'thigh', 'seizur'])
(60, ['fever', 'seizur', 'given', 'vomit', 'cellul'])
(61, ['2014.', 'shingl', 'shoulder', '2014', 'vomit'])
(62, ['headach', 'bodi', 'ach', 'vomit', 'shoulder'])
(63, ['shoulder', 'pain', 'elbow', 'shingl', 'left'])
(64, ['cellul', 'erythema', 'strength', 'healthcar', 'multipl'])
(65, ['cellul', 'shot', 'bodi', 'headach', 'shingl'])
(66, ['leg', 'anaphylaxi', 'headach', 'licens', 'pregnanc'])
(67, ['f', 'leg', 'provid', 'profession', 'right'])
(68, ['itchi', 'swollen', 'red', 'swell', 'cellul'])
(69, ['cellul', 'local', 'given', 'physician', 'right'])
(70, ['itchi', 'pharmacist', 'shot', 'thigh', 'flu'])
(71, ['shingl', '2012', 'profession', 'care', 'health'])
```

```
(72, ['shingl', 'itchi', 'child', 'l', 'headach'])
(73, ['pregnanc', 'leg', 'month', '2012', 'state'])
(74, ['l', 'hot', 'deltoid', 'itchi', 'erythema'])
(75, ['physician', 'shot', 'symptom', 'multipl', 'cellul'])
(76, ['cellul', 'pregnanc', 'month', 'l', 'shingl'])
(77, ['day', 'multipl', 'pregnanc', 'shot', 'cellul'])
(78, ['headach', 'upper', 'l', '2012', 'state'])
(79, ['hot', 'around', 'lump', 'warm', 'hard'])
(80, ['leg', 'shot', 'dizzi', 'data', 'logger'])
(81, ['local', 'thigh', 'reaction', 'call', 'l'])
(82, ['hot', 'local', 'lump', 'hard', 'warm'])
(83, ['itchi', 'local', 'warmth', 'upper', 'leg'])
(84, ['upper', 'administ', 'celsiu', 'state', 'fahrenheit'])
(85, ['around', 'upper', 'develop', 'area', 'flu'])
(86, ['upper', 'around', 'itchi', 'shot', 'cellul'])
(87, ['warmth', 'physician', 'thigh', 'incorrect', 'deltoid'])
(88, ['local', 'warmth', 'tender', 'week', 'erythema'])
(89, ['cm', 'erythema', 'x', 'child', 'warmth'])
(90, ['around', 'chest', 'upper', 'tender', 'lump'])
(91, ['develop', 'warmth', 'tender', 'l', 'hot'])
(92, ['3', 'cm', 'symptom', 'lump', 'flu'])
(93, ['tender', 'fatigu', 'warmth', 'symptom', 'lump'])
(94, ['tender', 'pregnanc', 'muscl', 'itchi', '9'])
(95, ['around', 'inch', 'l', 'area', 'week'])
(96, ['back', 'child', 'extrem', 'upper', 'warmth'])
(97, ['lump', 'around', 'call', 'hard', 'hot'])
(98, ['muscl', 'thigh', 'child', 'physician', 'leg'])
(99, ['warmth', 'dizzi', 'symptom', '2016', 'around'])

Training a RandomForestClassifier on the topic probability matrix

topic accuracy : 0.816893260052
topic recall   : 0.339310344828
topic precision: 0.615
prior accuracy : 0.790280590107
prior recall   : 0.0
prior precision: 0.0

~ confusion ~
reference:
[['TN' 'FP']
 ['FN' 'TP']]

confusion [RandomForestClassifier]:
[[2578  154]
 [ 479  246]]
```

3 Conclusion

Below is a summary of the performance metrics for each method.

Scikit Decision Tree		NLTK Decision Tree		LDA Topic Extraction	
accuracy	0.832	accuracy	0.847	accuracy	0.817
recall	0.256	recall	0.388	recall	0.339
precision	0.864	precision	0.796	precision	0.615

It appears that using a decision tree with binary rules, such as NLTK does, is the optimal solution for this dataset. However, this does not take into account testing variance. Depending on the situation, it might be desirable to maximize either recall or precision. For instance, if the goal is to minimize the number of false positives in order to save resources, decision trees would be the clear choice due to their precision. However, if the goal is to identify as many serious cases as possible without regard to false positives, topic extraction is a suitable choice given its recall.