Project 4: Deception Detection

Scott Cambo (sac355)
Graham Harwood (gdh56)
Jennifer Doughty (jad359)
Joseph Porter (jmp392)

1. General

For our project we continued working with the python programming language. Libraries that we utilized for this project include the NLTK programming package, scipy, numpy, and sci-kit learn. Additionally, we redesigned the unigram Naive Bayes for that we coded in project 2 for author detection to be used for deception detection.

1a. Configuration

The coding section of our project consists of a library that contains most of the functionality while the charBayesMachine, posBayesMachine.py, and the wordBayesMachine.py run the unigram naive bayes machine learning methods. charBayesMachine formulates the naive bayes calculation based on the frequencies of characters. posBayesMachine formulates the naive bayes calculation based on the frequencies of parts of speech. wordBayesMachine formulates the naive bayes calculation based on specific word frequencies. All three are initialized with empty data structures decCount, wordDecCounts, wordCounts, and decProbs. These are populated through the methods setFromReviews and setDecProbs. Calculations are made in naiveBayes takes in a test string and returns what the tag for that review should be. The library includes all feature extraction methods as well as the methods to learn, test, validate, graph, and output all other machine learning methods.

1b. Data Processing

Parsing the data for phase 1 was relatively straightforward. We kept track of our data through a list of dictionaries that represent each review and the attributes of the review including the actual review content, the tokenized review content, the part of speech tuples for the review (parsed using NLTK's POS tagger), and the POS frequencies. The Phase 4 data was more complicated because it included submissions from students who did not use a plain text format in their phase 3 submission. Due to this, we wrote a cleaning method that replaces common ascii characters with the characters that we would expect. The sentiment tag was also included in the dictionary data structure for phase 4 since this was available.

2. Non-programming Phases

2.1 Phase 1

During phase 1 we identified 11 different features that seemed to differentiate between truthful and deceptive reviews. These included number of proper nouns, duplicate adjectives, superlatives, average sentence length, patterns of sentence length, punctuation, purpose of hotel stay, commonalities with hotel publications, compound adjectives, qualifying statements, and sensory imagery. We attempted to design features around all of these that were feasible within the scope of the project. We did some preliminary coding to attempt to differentiate between what we intuitively noticed about the reviews and the ground truth of certain features.

For some features, what we thought were defining characteristics of deceptive reviews turned out to be actually fairly uniform across all of the reviews. One example of this is proper nouns. As we discussed in our phase 1 submission, we thought that a truthful reviewer would leave out information like the hotel name and location data which would be expressed as proper nouns because on a review website this information would be listed at the top of the page. Someone writing opinion spam may not look at the review site and would want to reinforce the specific hotel and location. When we added up the proper nouns across the phase 1 data though the averages were not significantly different between deceptive and truthful reviews. We will discuss which features were more valuable later.

As far as possible features that could be considered as nearly impossible to classify using computational means, we thought that highly specific details in a review such as brand name equipment in the room and specific measurements. These features were particularly prevalent in deceptive reviews. Ott et

al. also mentioned the prevalence of spatial data in truthful reviews which is hard to computationally classify. Spatial data could lose some value in the test set for this project in that student written reviews intentionally included spatial data. We will discuss the value of each feature in section 3.1.
Our final program included the following features (totalling 31):

- Sentence length by both character and tokens
- Number and frequency of discrete words in a review
- Number nouns, verbs, adverbs, adjectives, superlatives, determinants, and pre-determinants
- Number of positive words, negative words, the ratio of the two, and the review's predicted sentiment
- Number of exclamation points (and duplicate exclamation points), ellipsis, semicolons, dollar signs, quotation marks, commas, and periods
- Frequency of words associated with conditionals, family, luxury, location, comfort, business, weddings, spas, and convenience

2.2 Phase 2

As a group we analyzed the phase 2 data looking for features that we had identified from phase 1 as being deceptive. Our results were somewhat encouraging with the 3rd ranked kaggle score (not counting Ashesh.) We were incorrect on the following reviews (indexed such that first review is review 1):
[4, 5, 6, 7, 10, 14, 21, 23, 24, 33, 34, 35, 36, 38, 39, 40, 41, 44, 53, 54, 56, 57, 60, 62, 64, 66, 69, 71, 72, 76, 79, 84, 85, 87, 88, 90, 92, 96, 97, 100, 101, 102, 104, 106, 107, 108, 110, 114, 115, 117]
Overall this has an accuracy of 57.5% which is reasonably near the human judges in the Ott et al. paper. Several factors were common in misclassified reviews:

- Specific hotel related data that sounds more like a litany of features than an actual review. In general we thought this would be a sign of deception in that such features would be obvious from a hotel website but less important to someone who had stayed at the hotel.
- Misspellings also posed a problem in that it was hard to determine whether misspellings occurred as a result of incapable turkers, dissembling turkers, or actual reviewers making mistakes.
- We also found that the more fluidly and accurately a review read the harder it was to classify. In several cases deceptive reviews were especially fluid and convincing but were marked as truthful. This could be due to reader inclination toward "good" writing. In general we were inclined to err toward the gut feeling instead of our specific features which may or may not have helped us in terms of accuracy.

One particular example of a review that was misclassified by our group that posed a significant problem was review 117. This review contains a mix of spatial data and feature litanies. Ott et al. specifically point out spatial data as a feature associated with a truthful reviews. At the same time the review also includes a somewhat overwrought list of features which had been characteristic of deceptive reviews in phase 1. As a result we went with deceptive because we believed that the litany of features sounded more like a brochure than a review. As a result we were inclined to believe it was deceptive. In this case spatial data would have been a better indicator.

Another interesting case was review 35. This review has a couple of the features we generally considered to be deceptive such as exclamation points and a litany structures, but again we were tripped up by an intuitive sense of truthfulness. This to some extent reinforces the findings of Ott et al. in that human intuition is somewhat defective when it comes to making distinctions between deceptive and truthful reviews.

Review 56 is also an interesting case that we labeled incorrectly for phase 2. This review refers to Chicago itself being superior several times. The writer also uses several indefinite descriptive terms such as "down to earth" and "warm feeling." These terms refer to non specific descriptions of the hotel which could imply a truthful review of the hotel in that a hotel employee or Amazon Turker would want to concentrate on specific features of the hotel to mention. These features confused us as we were attempting to classify and lead to a misclassification.

Phase 2 Kaggle Results:
gdh56sac355jad359jmp392: 0.57369

3. Phase 4

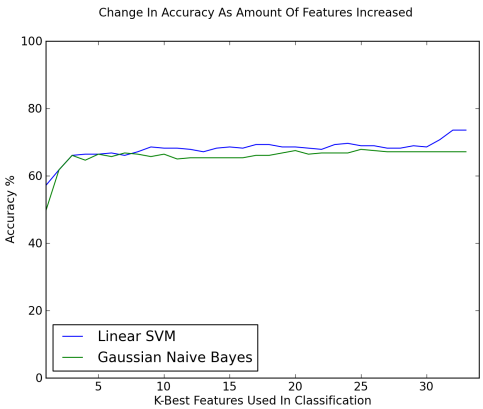
3.1 Feature Performance

We initially began by conceiving of a large amount of features that may be correlated with whether a review is deceitful or not. We used a couple methods to evaluate feature performance and eliminate features with poor correlation in the classifier. The first was to extract each feature independently and use this sole feature as a classifier. The second was to implement a chi-squared error method available from scikit-learn to choose only the k-best features. We have two tables to present our results. The first displays the top 10 best features and their accuracies under 3 different classifiers, Linear SVM, Gaussian Naive Bayes, and a Random Forest Decision Tree classifier.

Feature Validation Accuracy

Rank	Linear SVM	Gaussian Naive Bayes	Decision Tree
1	'$' characters : 63.21%	'$' characters : 63.21%	'.' characters : 63.93%
2	'.' characters : 62.86%	'...' : 62.5%	verbs : 63.57%
3	'...' : 62.25%	'.' characters : 62.14%	'$' characters : 63.21%
4	consecutive '!' : 59.29%	proper nouns : 60.00%	'...' : 62.5%
5	quotations : 58.21%	quotations : 59.64%	determiners : 60.36%
6	proper nouns : 58.21%	consecutive '!' : 59.29%	consecutive '!' : 59.29%
7	adjectives : 57.86%	adjectives : 58.57%	quotations : 58.21%
8	deceptive words : 57.49%	determiners : 57.89%	proper nouns : 57.86%
9	superlative adjectives : 57.14%	superlative adjectives : 57.50%	superlatives : 57.50%
10	convenience words : 57.14%	number of tokens: 57.14%	number of tokens : 57.14%

For Linear SVM and Naive Bayes, the top performing feature was the number of '$' characters that appeared in the review. In doing a quick search of all the reviews that have '$' signs in them, they nearly always refer to the price of the hotel or some commodity at the hotel. Another interesting thing to note is that these reviews also tend to be more deceptive and more negative. Similarly many reviews containing quotation marks are referring to the size of the TV that might be in the room, which is generally a boasting point for many hotels, but not by their patrons. Many of the features that performed well were mentioned in Otto et al. such as the amount of determiners, proper nouns, adjectives, and words relating to convenience (e.g. 'convenient' and 'conveniently')



For further feature analysis, we also ran an experiment seeing how well our methods performed with increasing amounts of features selected by sci kit learns' SelectKBest method using chi-squared scoring. Many of the features selected at each stage of the experiment (e.g. features that use words concerning luxury or family) were not consistent with the results of experiment on how features perform independently. This makes sense given that the chi-squared scoring is different measure of a feature's likelihood of performing well with a classifier. Chi-squared scoring can also be done without knowing the results of a feature's independent performance on a test

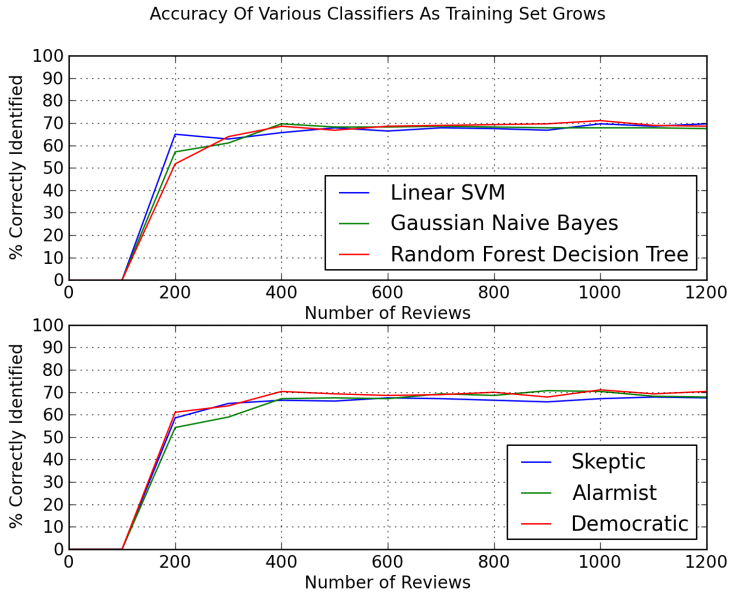
set making it a viable tool for feature elimination.

Student Naive Bayes Results

Classifier Type	Feature	Accuracy
Naive Bayes	Words	68.57%
Naive Bayes	Characters	65.71%
Naive Bayes	Parts of Speech	61.07%

The student created naive bayes method shows that words are the most accurate unigram feature. This aligns with the Ott et al. conclusion particularly when compared to characters and parts of speech. Characters and parts of speech have the advantage of being independent of a speaker's individual vocabulary; however, in this case individual vocabulary appeared to provide more accurate results. This may be because this type of corpus is so specific to a single type of review. Many words may be repeated in the context of hotel reviews especially since all of the hotels are in the same city. As a result in this case word commonalities provide a greater insight due to the jargon associated with both hotels and real estate.

3.2 Classifier Performance



Each machine learning method we examined seems to follow a similar pattern as the size of the training set grows. We tested each learning method by increasing the number of training reviews by 100 and testing accuracy against the validation set. All three reached an asymptote at about 500 training reviews. The linear SVM accuracy ascended most quickly toward the asymptote, but actually decreased around 300 training set reviews before again approaching the asymptote of approximately 70%. The Gaussian Naive Bayes learning method also ascended quickly, peaking at about 400 reviews, then again approaching the asymptote along with the other two learning methods. After 500 reviews are added to the training set, the accuracy stabilizes but there is one additional peak at 1000 reviews. The observed variations and noise are likely due to differences between the batches of 100 reviews added to the training set. This observation is consistent with the Law of Large Numbers, which suggests that increasing the number of observed examples will bring average values of the set closer to that of the true mean values.

The second graph focuses on several boosting methods in which we combine the three classifiers. Skeptic chooses 0 if any of the learning methods return 0. Alarmist chooses 1 if any of the learning methods

chooses 1. Democratic takes the majority vote. For this learning curve model, skeptic scores the lowest until after 400 reviews are added, after which it mirrors alarmist; as skeptic increases in accuracy, alarmist decreases in accuracy. One reason for this shift may be that the first 200 or 300 reviews added have significantly more 0 tagged data points than the later reviews or the training set as a whole. This trend appears to reverse at around 700 training reviews—again consistent with the Law of Large Numbers—where it is possible that there are many more training reviews that have a 1 tag. The two scores then converge as the number of training reviews reaches 1000, possibly indicating a somewhat more even distribution of 1 and 0 tagged reviews.

Overall, the learning curves suggest that a training set of approximately 400 reviews is sufficient in order to train a classifier in this application of deception detection.

3.3 Kaggle Score

Phase 4 Kaggle Results
gdh56sac355jad359jmp392: 0.79137

The methodology used to construct the results achieving the greatest area under the ROC curve, and thus our highest Kaggle score, utilized the SVM and random forest classifiers. We noticed that our submissions achieved higher scores when we submitted values in the range of [0,1] as opposed to {0,1} in which each value is indicative of the confidence of the classification. Classifications close to 0 or close to 1 represent confidently-assigned labels, whereas classifications close to 0.5 represent classification uncertainty. This methodology effectively produces more points on the ROC curve, and was more effective for us in maximizing the area under the curve. The following procedure was used to obtain values for each of the three classification techniques:

- **SVM**: After training, obtain "decision function" and map values onto [0,1] scale.
- **Random Forest**: After training, obtain predicted class probabilities, computed as the mean predicted class probabilities of the trees in the forest.
- **Naive Bayes**: After training, obtain predicted class probabilities. Note that Naive Bayes is known to be a bad estimator, and thus these values are not necessarily good predictions.

The following table shows area under the ROC curve results from Kaggle:

Classification Method	Area Under ROC Curve
SVM	0.75287
Random Forest	0.78604
Naive Bayes	0.72862
SVM & Random Forest (averaged)	0.79137
SVM, Random Forest, & NB (averaged)	0.76251
SVM, RF, & NB with thresholding (averaged)	0.76238

4. Extensions

4.1 Sentiment Based Features

One feature that we implemented for this project was the use of conditional sentences eg. "The room was very large, but the configuration was oppressive." We saw this feature particularly in truthful reviews. This fits because both Turkers and hotel employees would be somewhat hesitant to admit the faults of the hotel when trying to give a positive review and would be unlikely to provide possible positive aspects when writing a negative review. We were initially unsure of ways to extract this feature. We considered both commas and words associated with conditional statements. Commas and semicolons were too prevalent in other grammatical structures that we had seen in phases 1 and 2 to be used to extract this specific feature.

We then consulted a list of conjunctive adverbs and conjunctions in order to find words that could signal a conditional structure. We created a bank of these words and scored sentences based on the presence of these words.

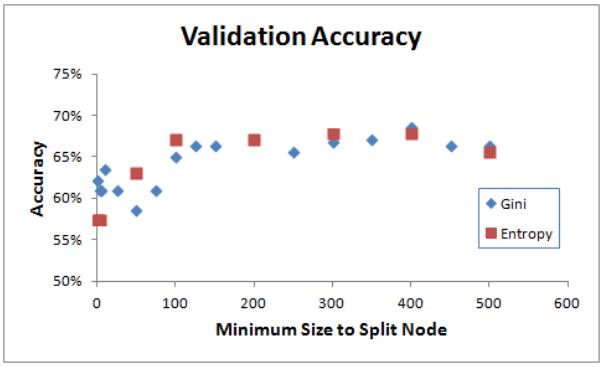
Initially this feature did not appear to be particularly discerning when used in a machine learning algorithm. We then printed the reviews that showed several of the conditional words we found that they were actually more often deceptive but the deceptive reviews were almost exclusively negative while the truthful reviews that included several conditional terms were almost exclusively positive. As a result we decided to make a sentiment predicting method that could be used in conjunction with the conditional feature.

Our sentiment predicting function is based on the inclusion of positively and negatively associated words. We gathered lists from two different sources: "Opinion Mining, Sentiment Analysis, and Opinion Spam Detection" from the University of Illinois at Chicago[1] and "Word Lists" from the University of Notre Dame[2]. We scored each review based on the positive and negative words present in the review and assigned a sentiment based on those scores. This method proved to be 81.4% effective on the validation data and 76.3% accurate on the training data. While not ideal, these achieved the purpose of improving the conditionality feature to factor in the sentiment associated with each review.

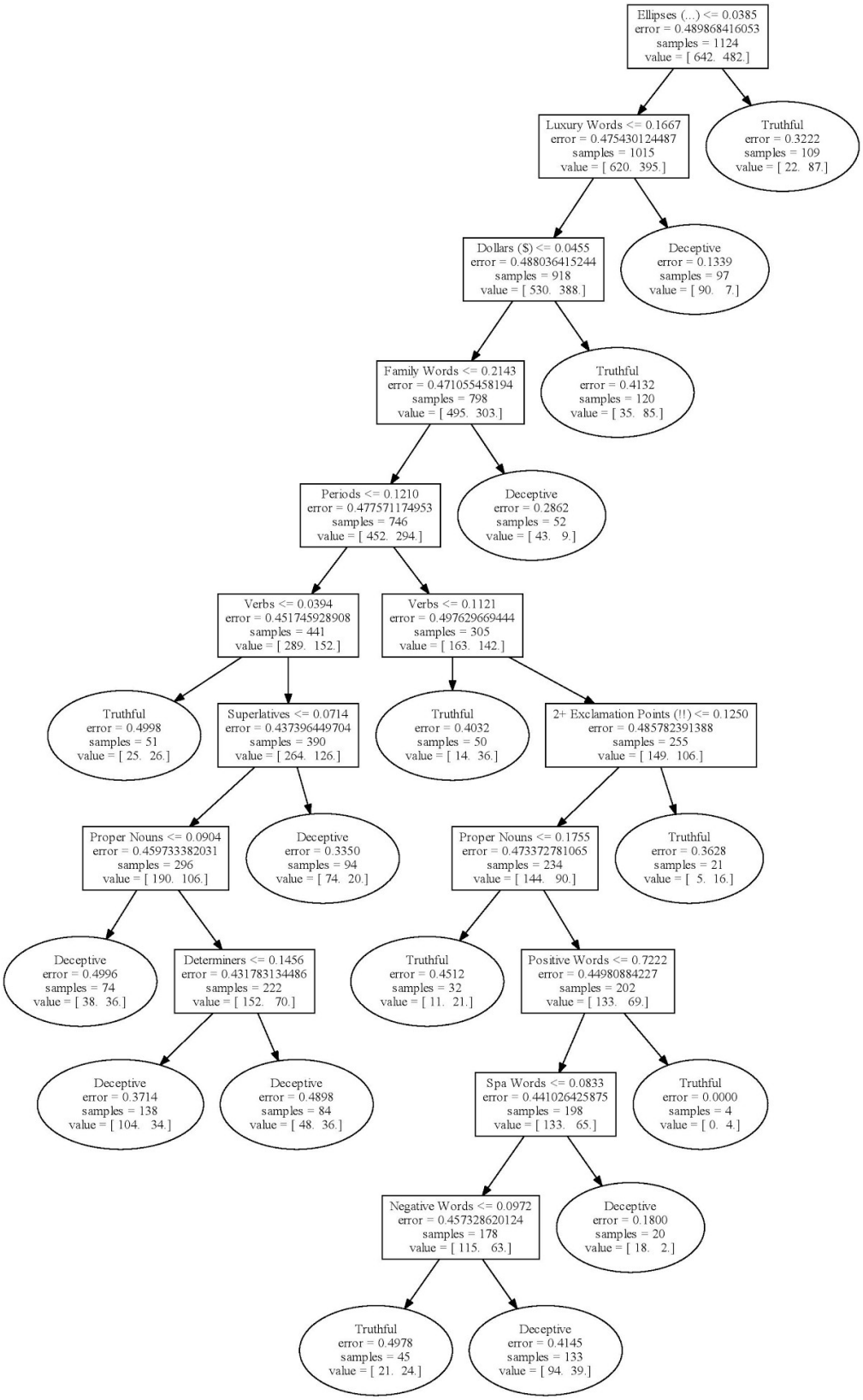
We also included the raw opinion scores in order to observe the effects of the predicted sentiment on the deceptiveness of hotel reviews. This method in particular could provide interesting results in that it examines the number of positive and negative words in relation to each other which could prove to be discerning between deceptive and truthful reviews.

4.2 Decision Tree Learning Classifier

Given that deception detection is a binary classification task for multiple features, decision tree learning is a natural fit. Decision trees are generated by recursively partitioning data sets based on an attribute value test, often in a greedy fashion. The attribute on which to split is chosen by maximizing some particular metric that is representative of how effectively the split purifies the resulting subsets. We utilized scikit-learn to generate decision trees based on Gini impurity and entropy reduction. To prevent overfitting, we pruned the decision trees by allowing partitions only if the node is greater than a particular size. The effectiveness of each split was assessed by computing accuracy for each decision tree on the validation dataset, the results of which are displayed in the figure at right. A reduction in accuracy is noticeable for maximum leaf sizes less than 100, suggesting that those decision trees are overfit to the training data. Non-overfit decision trees resulted in accuracies of approximately 65-69%. One such tree is presented on the following page. The tree performs fifteen splits on thirteen unique features.



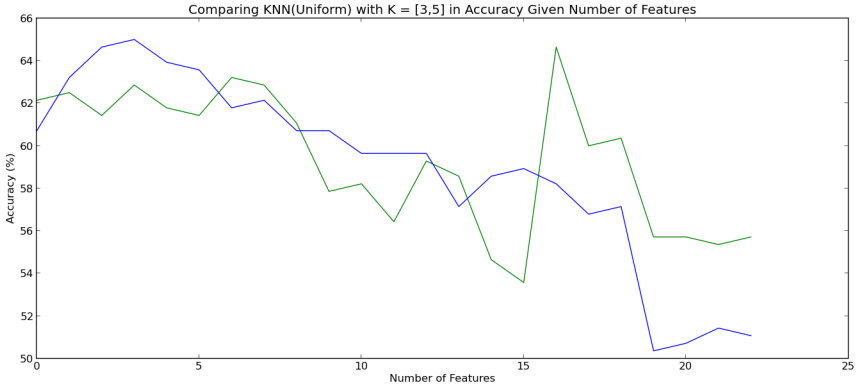
[1] http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html
[2] http://www3.nd.edu/~mcdonald/Word_Lists.html


Ellipses (...) <= 0.0385
error = 0.489868416053
samples = 1124
value = [642. 482.]
Luxury Words <= 0.1667
error = 0.475430124487
samples = 1015
value = [620. 395.]
Truthful
error = 0.3222
samples = 109
value = [22. 87.]
Dollars ($) <= 0.0455
error = 0.488036415244
samples = 918
value = [530. 388.]
Deceptive
error = 0.1339
samples = 97
value = [90.   7.]
Family Words <= 0.2143
error = 0.471055458194
samples = 798
value = [495. 303.]
Truthful
error = 0.4132
samples = 120
value = [35. 85.]
Periods <= 0.1210
error = 0.477571174953
samples = 746
value = [452. 294.]
Deceptive
error = 0.2862
samples = 52
value = [43.   9.]
Verbs <= 0.0394
error = 0.451745928908
samples = 441
value = [289. 152.]
Verbs <= 0.1121
error = 0.497629669444
samples = 305
value = [163. 142.]
Truthful
error = 0.4998
samples = 51
value = [25. 26.]
Superlatives <= 0.0714
error = 0.437396449704
samples = 390
value = [264. 126.]
Truthful
error = 0.4032
samples = 50
value = [14. 36.]
2+ Exclamation Points (!!) <= 0.1250
error = 0.485782391388
samples = 255
value = [149. 106.]
Proper Nouns <= 0.0904
error = 0.459733382031
samples = 296
value = [190. 106.]
Deceptive
error = 0.3350
samples = 94
value = [74. 20.]
Proper Nouns <= 0.1755
error = 0.473372781065
samples = 234
value = [144. 90.]
Truthful
error = 0.3628
samples = 21
value = [5. 16.]
Deceptive
error = 0.4996
samples = 74
value = [38. 36.]
Determiners <= 0.1456
error = 0.431783134486
samples = 222
value = [152. 70.]
Truthful
error = 0.4512
samples = 32
value = [11. 21.]
Positive Words <= 0.7222
error = 0.44980884227
samples = 202
value = [133. 69.]
Deceptive
error = 0.3714
samples = 138
value = [104. 34.]
Deceptive
error = 0.4898
samples = 84
value = [48. 36.]
Spa Words <= 0.0833
error = 0.441026425875
samples = 198
value = [133. 65.]
Truthful
error = 0.0000
samples = 4
value = [0. 4.]
Negative Words <= 0.0972
error = 0.457328620124
samples = 178
value = [115. 63.]
Deceptive
error = 0.1800
samples = 20
value = [18. 2.]
Truthful
error = 0.4978
samples = 45
value = [21. 24.]
Deceptive
error = 0.4145
samples = 133
value = [94. 39.]

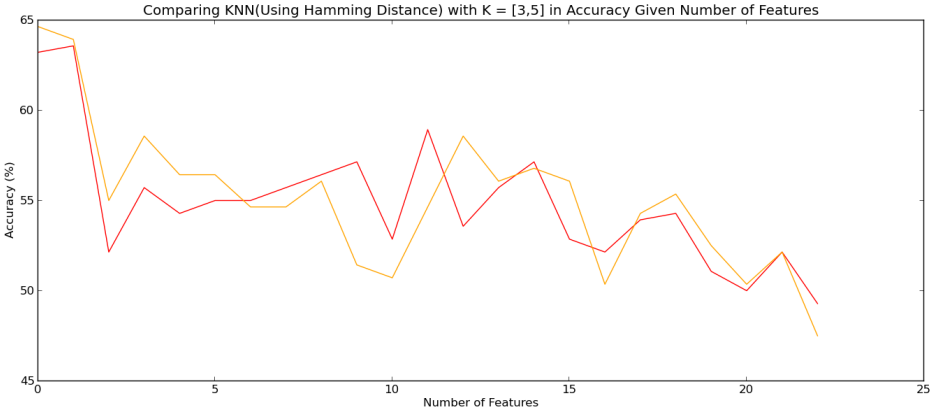
To further improve the classification performance, we investigated scikit-learn's random forest classifier. A random forest classifier generates decision trees with a randomized component on various sub-samples of the dataset; it uses averaging to improve predictive accuracy and control overfitting. Validation accuracy using random forest classification was approximately 68-72%, which is consistently greater than the accuracy achieved using one decision tree alone.

4.3 Expansion of the kNN to use the Hamming Distance

The nature of the kNN algorithm makes it a lazy, instance based learning algorithm and thus can serve as a quick implementing classifier given the vector structure of the data. Here using the scikit-learn KNeighborsClassifer module, we were able to implement a kNN classifier using the default distance measurement as well as Hamming Distance. Upon doing the analysis, we were interested in investigating the relationship between the number of neighbors used in the classification as well as the number of features in the accuracy of the classifier. We observed a value of k at both 3 (green) and 5 (blue), in which we investigated 5 through all features. The highest accuracy of the default kNN algorithm was 65% found at a k value of 5 and using 8 features.



We observed the same relationship using the Hamming Distance metric and found that the highest accuracy of 64.6% found at a k value of 5 and using 5 features. In the diagram the red line is at k of 3 and orange k of 5. This was interesting due to the 3 less features used in the Hamming Distance classifier versus the default kNN classifier.



5. Individual member contribution

Group Member (netID)	**Contribution**
Jennifer Doughty (jad359)	Created features relating to punctuation and word choice, analyzed effectiveness of decision tree and random forest classifiers, considered boosting algorithms, investigated AUC and how to use confidences to increase Kaggle score
Scott Cambo (sac355)	implemented sklearn svm experiments, sklearn naive bayes experiments, all feature analysis, parsing and cleaning methods, feature extraction for words unique to desceptive reviews, plotting, methods for running feature extractions.
Graham Harwood (gdh56)	Created Naive Bayes unigram character, part of speech, and unigram word classifiers, contributed to feature analysis, reflected on phase 2, created sentiment based extension, and conditional feature extraction
Joseph Porter (jmp392)	K-nearest Neighbors, Hamming Distance