STAT 154: Hillary Clinton Email Sender Classification Project

by

Kurt Cakmak, Yuting Kou, Yingying Li, Shu Xu

In 2015, controversies arose over a leak showing that Former Secretary of State, Hillary Clinton, while leading the State Department, had used her personal email accounts on private servers to relay both classified and unclassified material. The goal of this project is to perform text-based classification on a subset of the released unclassified "Clinton Emails" to determine the sender corresponding to each email.

Preliminary analysis of these emails shows in *Table 1* reveal that there are 3,505 emails in total across five different classes of senders. There are approximately 685, 1,023, 1,241, 271 and 281 emails belonging to senders one, two, three, four and five, respectively.

Sender	# Emails	Frequency
1	685	19.54%
2	1023	29.19%
3	1241	35.41%
4	271	7.85%
5	281	8.01%

Table 1: Email Frequency Belonging to Each Sender

Implementation of the project is divided into five parts:

(1) Part one: Discusses feature creation, filtering and power feature selection using the original 3,505 emails to determine a model size with high classification accuracy.
(2) Part two: Discusses feature selection methods implemented to improve classification accuracy using the Random Forest and SVM classifiers.
(3) Part three: Discusses the results of applying the K-Means Clustering algorithm onto the model determined in part one.
(4) Part four: Compares the results of all three classification algorithms and determines the final model with the highest test classification accuracy rate.

We now proceed part one to create our working model using the training email data set.

Part One: Feature Creation, Filtering, Power Feature Additions

The Python coding language was used for feature creation and filtering. After loading the training data, HRC_train.csv, into Python, a data frame with two columns was created for all 3,505 emails based on (1) label, the sender of each email, and (2) content, the actual correspondence between Secretary Clinton and the email sender of the email. Next, the NLTK library in Python was imported to tokenize the content of each email, separating words, digits and punctuations. We then proceed to implement two differing models for feature creation and filtering. Our working word feature matrix for part two will be chosen from among the following two methods.

Method One:

Continuing to utilize the NLTK library along with other essential Python libraries for text parsing, we proceed to perform feature creation by removing all punctuation and numerical digits from each of the tokenized emails. Stop-words are also removed from the remaining tokenized words using these libraries, including all single letter instances from the English alphabet and non-email-essential words such as 'fw', 'subject', 'case', 'doc', and 'unclassified'. Moreover, the Python Library Enchant was applied to remove any non-English words reduce the number of features across all emails. In our last step in parsing the data, the Stemming library was applied on all remaining tokenized email words.

In order to form a feature matrix using all unique words from the 3,505 emails, we create dictionaries for each of tokenized emails, mapping each feature with its frequency in that email. Next, using the SKLearn package in Python, we performed Bag of Words (BOW) to extract all unique stemmed words among these dictionaries combined and added them as the feature columns of our matrix. The rows of our matrix correspond to each email and the frequency that each unique feature occurs within the email. Consequently, many features will have a frequency of 0 for any given email. Our final feature matrix using method one thus uses *** features in total.

Method Two:

Individual assessment of various emails by each sender reveal that certain features, such as punctuation, numeric values and especially foreign names and cities are prominent to certain senders. Hence, in method two, we proceed to create a word feature matrix which retains all unique occurrences of these features. We only remove stop-words from each of the emails, and then perform stemming on the remaining tokenized words. Using the same process as in method one, we create our feature matrix from these tokenized, stemmed emails. Lastly, feature filtering is performed on our temporary word matrix to eliminate inessential features, specifically features which occur minimally throughout all 3,505 emails for the five senders. We decide to remove any words in our matrix occurring less than 10 times overall. The filter value 10 is initially selected because it substantially reduces the size of our feature matrix while also retaining features with low term frequency which still could have important document frequency for a given sender or senders. In *Table 2*, our temporary feature matrix using finally reduces to 7,233 unique features, including the response column.

Step	Total # of features
Raw	3505 by 50943
Remove Stop words	3505 by 39712
Stemming	3505 by 30604
Filter out words that occur less than 10 times	3505 by 7233
Adding power features	3505 by 7239

Table 2: Total Number of Features during Feature Creation and Filtering

Power Features Additions:

Name of Power Feature in Model	Power Feature Description
“len_of_email”	How many words in each email
“unique words”	How many unique words in each email
phrases	Phrases, topics pertaining to a specific sender
PCA1, PCA2	Top two Principal Components

Table 3: Power Feature Additions and Descriptions

Power features are also determined to increase sender classification accuracy (*see Table 3*). Analysis of several emails reveals that email length varies among certain senders and thus could be a potentially useful feature for classification. Using email length as a power feature in our model determined from model two, we use Random Forest in part two to compute the OOB error rate. Unfortunately, the OOB error rate was approximately 75%. However, when analyzing the top 500 features used by the Random Forest classifier, email length was still a critical feature in sender classification. Thus we include the addition of this power feature in our temporary model going forward.

Additionally, other unique features pertaining to each sender were considered and experimented with our model from method two as “power features”. Analysis of each senders’ emails reveals that certain email discussions and topics were specific to various senders. Thus, important phrases pertaining to these topics were extracted for a certain sender and added as a column in our model. These phrases combined various features already present in our word matrix. Then an indicator variable was performed over each email to determine if the entire phrase occurred within the email. If all features were present, then the product of the word count corresponding to each of these features was calculated and added into our power feature column for an email. Otherwise, zero was placed in the power feature column instead. Specifically, as our misclassification rate using Random Forest was greatest for senders one and four, 30% and 15% respectively, important phrases, such as “benghazi libya”, “wave attack”, “ambassador steven”, frequently occurring within sender one’s email were added as power features. The addition of these power features into our model from method two increased the OOB rate for sender classification by roughly 1%, while the misclassification rate remained nearly unchanged for senders one and four. Therefore, we do not consider the inclusion of these power features in our temporary working model going forward.

Additional Implementations:

In addition to using Bag of Words for both methods described above, other implementations were considered for feature creation and filtering to improve our test classification accuracy rate.

The first additional model implemented Word2Vec to perform topic modeling by mapping each word into a fixed-sized vector. The main advantage of representation the features in this vector form is to group similar words “close” together in the vector space. This “closeness” is partly evaluated by the number of neighbors taken into account for each feature in a given context. Thus, our model transforms each email into a vector using the average of all words in it. There are 2 main parameters to tune: D (the dimension of the vector space), and C (the length of context taken into account). We experimented with various values for D, ranging from 100 to 500, and C, ranging from 5 to 30. The overall best combination of our tuning parameters was D = 300 and C = 15. However, the highest accuracy rate was merely 68.5%, far from ideal.

The second additional model we implemented was TF-IDF. TF-IDF, standing for term frequency-inverse document frequency, is a feature vectorization method reflecting the importance of a word to an email in the corpus of written texts. IDF is a numerical measure of the quantity of information a word provides. In the algorithm for IDF, taking the log decreases the importance of a word appearing across many emails in all classes. Applying the TF-IDF transformation on our frequency matrix from method two before adding power features, the OOB error rate from Random Forest using the tuning parameters mtry = 100, B = 300), was 23.59%, slightly better than without TF-IDF: 24.22%.

Part 2: Model Selection, Feature Selection, Random Forest & SVM Classifiers

Model Selection from Methods One and Two:

In order to select the model from part one with the lowest test prediction error rate, OOB error analysis and 5-fold CV classification accuracy is performed using the Random Forest and Support-Vector-Machine classifiers, respectively. The OOB error rate using Random Forest returns 65% and 75% (see *Table 4*) for methods one and two, respectively, and the 5-fold CV classification accuracy rate using the optimal cost parameter for a linear-kernel Support-Vector-Machine classifier returns 62% and 65% for methods one and two, respectively. The linear-kernel was selected as it had the best performing test classification accuracy compared to the radial and polynomial kernels. Thus, since the classification accuracy was higher for both Random Forest and SVM using method two, we select the the feature matrix from method two as our working model and proceed to perform feature selection on this model.

Step	Total # of features used	Total Accuracy	Accuracy per sender class	Top Ten Features by Importance
RF	7239	75.78 %	55%, 82%, 81%, 63%, 90%	call, depart, full, fyi, message, mini, origin, part, len_of_email, unique_words

Table 4: Results of Random Forest Classifier on Method Two

Feature Selection:

Using the word matrix determined from method two with 3,505 emails and 7,234 features, including the response column, as our working model for sender classification, we perform feature selection to eliminate unessential word features and improve classification accuracy across all senders.

Five dimension-reduction methods are considered for feature selection. However, before each method is applied to our working model, we first determine a "target" feature size for our condensed matrix. As we would prefer the number of features in our new model to be less than the number of training observations, our target feature size is consequently initialized at 3,500 features. Decreasing our target size by increments of 500, in total we seek to attain five various word matrices, the lowest of which having 1,500 features.

In order to attain these five differing target feature sizes, an exact threshold value must be determined for each feature selection model. All features falling short of the threshold in each model will be eliminated such that the new word matrix matches the target feature dimension. The final threshold for each feature selection method will be manually calculated by observing the resulting number of features over a series of threshold value candidates.

Moreover, once the optimal threshold (see *Table 5*) is determined for each method under the respective target feature size, all five methods will be further combined together to perform a final feature selection on our working model so that only all relevant features remain. Hence, our new word feature matrices will most likely have feature dimensions less than our original target sizes, as the thresholds for multiple feature selection methods can be simultaneously satisfied.

Finally, we discuss the five dimension-reduction methods implemented and their respective threshold values under each target feature size:

(1) The first feature selection method analyzes the column sum of each feature by determining if the total frequency of a certain word across all 3,505 emails and 5 senders satisfies a given threshold. In order to retain important features that have low frequency across these emails yet could be prominent to a specific sender, the threshold is continually set at 20 (exclusive) for all five target feature sizes.

(2) The second feature selection method focuses on individual senders by analyzing the ratio of the total frequency count for each feature with the total number of emails

belonging to that sender. The goal of the second method thus is to determine features important across all senders through word frequency

(3) Like the second method, this method also focuses on individual senders, however, analyzes the ratio of the total number of emails containing a certain feature with the total number of emails belonging to that sender. The goal of method three consequently is to determine and retain features with non-minimal document frequency for each sender. Any features calculated to have sizeable word frequencies for a sender using the previous method, yet occur unsustainably across all the sender's emails thus are filtered out under this method.

(4) Similar to method three, this method also calculates the ratio of the total number of emails containing a certain feature for each sender with the total number of emails belonging to that sender, however, then analyzes the difference in the maximum and minimum ratios among the five senders for a given feature. The goal of method four thus is to determine and retain features which not only have substantial document frequency but also are more unique and applicable to a specific senders' emails.

(5) As in method three, the ratio of the total number of emails containing a certain feature for each sender with the total number of emails belonging to that sender is calculated. Afterwards, variance between these ratios for all five senders is analyzed to determine if a feature is unique or more applicable to a certain sender or senders across his/her emails.

Target size	(1)	(2)	(3)	(4)	(5)
1500	10	0.10	0.07	0.07	6
2000	10	0.07	0.05	0.05	3
2500	10	0.06	0.04	0.04	2.3
3000	10	0.05	0.03	0.03	1.5
3500	10	0.03	0.02	0.02	0.7

Table 5: Optimal thresholds for feature filtering under each target feature size

Lastly, for each target feature size, we combine all five methods with their respective threshold values and perform a final feature selection on our working model. Our resulting candidate word matrices have a feature dimension of 1,161, 1,637, 1,983, 2,387 and 3,280 including the response column, in lieu of the original 1,500, 2,000, 2,500, 3,000 and 3,500 target feature sizes, respectively. We then proceed to use Random Forest Classification on these five given candidate matrices to determine which model will yield the greatest test accuracy rate for sender classification.

Selecting Best Model Size Using Random Forest:

For each of the word feature models determined through feature selection, we perform simple Random Forest Classification and analyze the OOB accuracy to determine the feature size with the best potential test classification accuracy. We set both *mtry*, the number of features tried at each node, and *ntree*, the number of trees, fixed to $2\sqrt{P}$, where $P = Number\ of\ Features$, and 300, respectively, before performing the classification. Analyzing the result in *Figure 1*, we observe that the difference in OOB error rate for different feature sizes is minimal; Each model returns an OOB rate of roughly 22%, or 78% classification accuracy. However, the model with feature size 2,387 appears to perform better in the first half part of training, which is reasonable since included the most information and thus had the lowest OOB error rate overall: 21.88%. Therefore, we choose this working model to proceed.

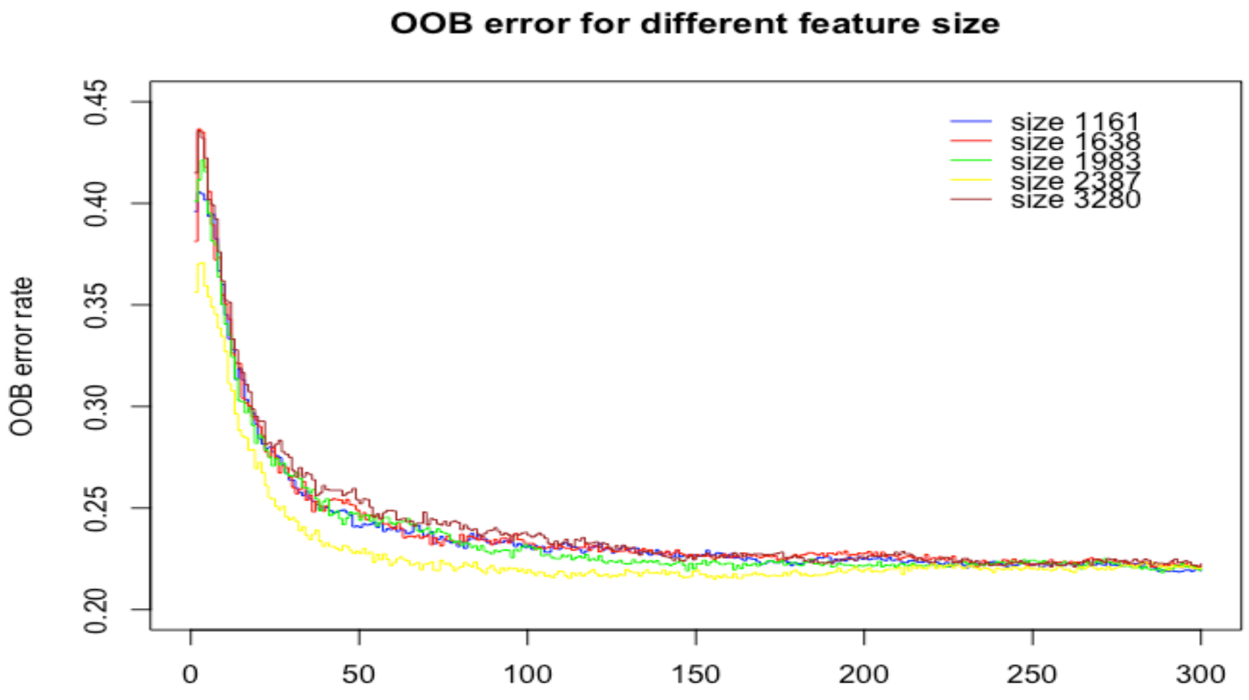

Figure 1: OOB error rate for the five candidate-models determined through feature filtering

Using the 2,387 feature matrix dimension determined through OOB accuracy, we tune the parameters *mtry* and *ntree*, using *mtry* = $\{\ log_2(P), \sqrt{P}, 2\sqrt{P}, \frac{P}{2}\ \}$ and *B (ntree)* = *[1:300]*, to determine the optimal values maximizing our test classification accuracy (see *Figure 2*). The tuned parameters returned for *mtry* and ntree are $\frac{P}{2}$ and 300, respectively, and the model has a OOB error rate of approximately 20.49 % (see *Table 6*). It is important to note that, compared to OOB error rate in the beginning of part two, we improved our classification accuracy by nearly 4% through feature selection while eliminating over 4,000 extra, unnecessary features.

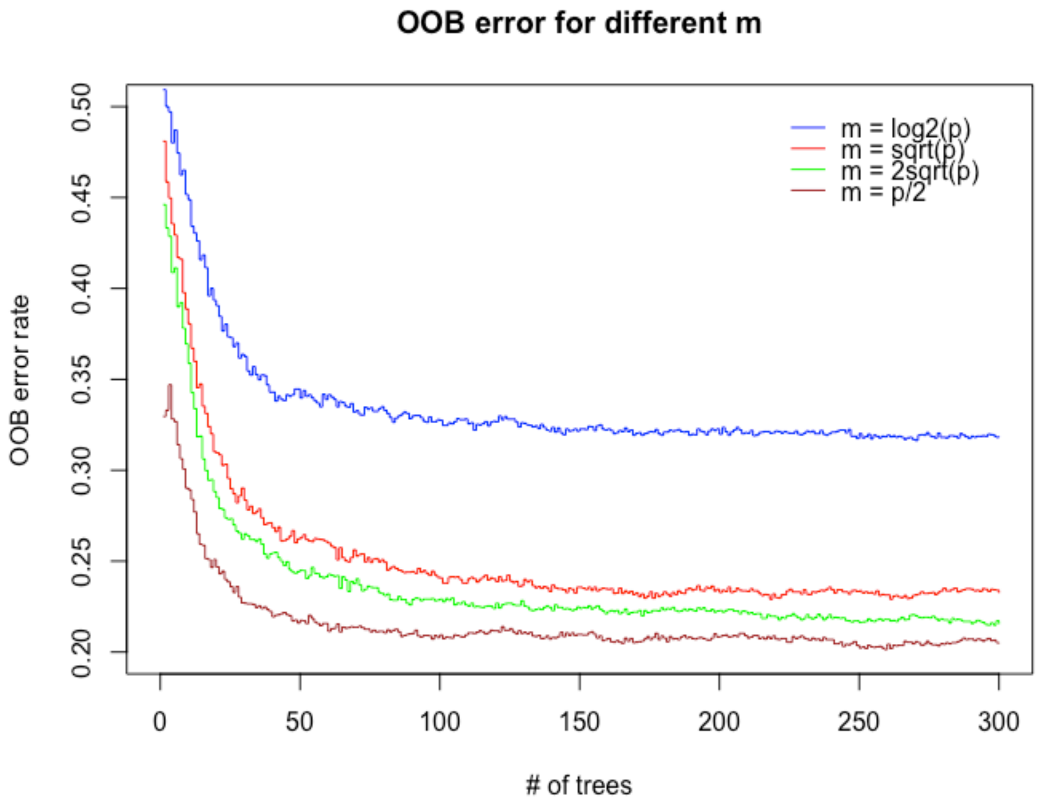

Figure 2: Results of parameter tuning on 2,387 feature matrix

Step	Total # of features used	Total Accuracy	Accuracy per sender class xx%, yy% ,zz%, aa%, bb%	Top Ten Features by Importance

RF(with feature selection)	2387	79.51 %	76%, 81%, 85%, 70%, 92%	cdm, fyi, call, Haiti, jake, list, mini, part, len_of_email, unique_words
RF(with feature selection)	500	79.54	67%, 81%, 84%, 69%, 93%	cdm, fyi, Haiti, jake, list, mini, part, sid, len_of_email, unique_words

Table 6: Overall classification accuracy results using feature selection and supervised feature selection

Supervised Feature Selection with Random Forest Using CV:

The 2, 387 feature matrix determined through feature selection and simple Random Forest Classification still retains a moderately high feature dimension after tuning. Our goal is to further eliminate any inessential features from this matrix while maintaining or improving test classification accuracy. Hence, we perform 5-fold cross validation on a variety of top feature selections. First we randomly divide the observations into 5 folds, and leave one out each time to perform Random Forest with *mtry (m)* = $\{\sqrt{P}, 2\sqrt{P}, \frac{P}{2}\}$ and *B* = *(150, 300)* as our tuning parameters on the top 300, 500 and 800 most important features determined by Random Forest on our 2,387 feature model using the same parameters. In total, there are 18 combinations and CV classification accuracies (see *Table 7* below).

m	B	# of features (# of most important features)		
		300	500	800
$\sqrt{P}$	300	21.88%	22.54%	22.68%
$2\sqrt{P}$	300	21.28%	21.08%	21.83%
$\frac{P}{2}$	300	20.46%	20.57%	20.94%
$\sqrt{P}$	150	22.31%	22.68%	22.57%
$2\sqrt{P}$	150	21.11%	22.00%	21.97%
$\frac{P}{2}$	150	21.06%	**20.46%**	20.60%

Table 7: Parameter tuning results for supervised feature selection with Random Forest using CV

Observing *Table 7*, we find that the model using the top 500 features with the tuned parameters $m = \frac{P}{2}$ and $B = 150$ return the lowest CV classification error rate. Our final feature matrix using Random Forest with supervised feature selection uses approximately $\frac{500}{2,387}$ (21 %) of the features determined through the original five-method feature selection.

Choosing Best Random Forest Classifier Overall:

Although the Random Forest Classifier model using supervised feature selection with CV had a slightly higher CV classification accuracy compared to the optimal Random Forest Classifier using the 2,377 feature matrix, we decide to select the model using the 2,377 features and tuning parameters $m = \frac{P}{2}$ and $B = 300$. The CV classification error rates are roughly the same for both, approximately 20.5%, however, the model using the 2,377 feature matrix provides more information for prediction on our test data set. By selecting this model, we decrease Bias in the selection of our model and concerns of overfitting we might have on our training set.

SVM Classification:

Using the feature matrix with 7,233 features determined from part one, we again apply feature filtering to remove features occurring less than 50 times overall and reduce the frequency matrix dimension to 2,362 features such that only the important features remain. We then divide the data into training and test sets, with the training set containing 80% of the data, and proceed to fit an SVM Classifier on the training set. Since there are five classes of senders, one-versus-one classification is implemented on our training set so there are $\binom{5}{2}$, or 10, two-class SVM classifiers for each 5-class SVM classifier that we implement.

As there are three possible kernels for our SVM classifier, linear, radial and polynomial, we use the Caret Package in R to perform 5-fold cross-validation and choose the kernel with the highest CV classification accuracy. Analyzing the boxplots of the ranges in classification accuracy for each kernel in *Figure 3*, the SVM classifier with a polynomial kernel has the highest *mean* classification accuracy rate, 74%, followed by the linear kernel with 72%. We select the SVM classifier using the linear kernel, as it is less computationally intensive and still has a relatively high *mean* classification accuracy rate.

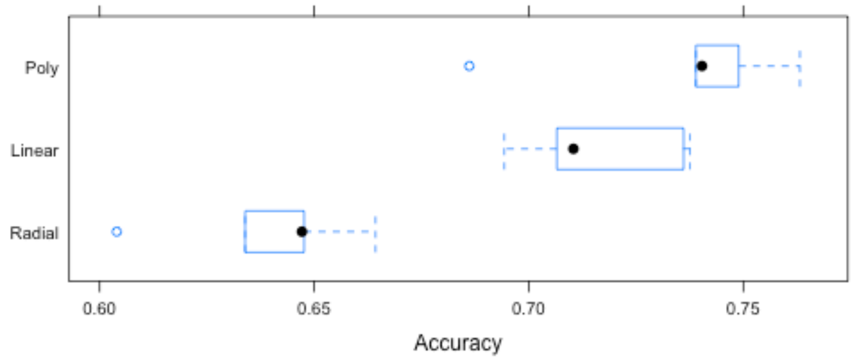

Figure 3: Boxplot of classification accuracies for SVM with linear, radial and polynomial kernels

Next, we manually perform 5-fold CV on our linear SVM classifier using the training set to select the tuning parameter *Cost* that returns the lowest test CV misclassification rate. Running the linear SVM classifier with cost candidate's *c = {0.0001, 0.001, 0.01, 0.1, 1, 10, 100}*, 0.01 returned returned the highest classification accuracy, roughly 77% (see *Table 8*).

C (error penalty)	Accuracy
C = 100	0.7210
C = 10	0.7210
C = 1	0.7210
C = 0.1	0.7372
C = 0.01	**0.7708**
C = 0.001	0.7079
C = 0.0001	0.3784
C = 0.00001	0.5315

Table 8: Tuning parameter results for Cost for linear SVM

We then proceed to fit this linear SVM classifier with the optimal cost parameter onto our entire training set. We calculate our best training accuracy rate to be 85% (see *Table 9*).

Classifier	Total # of features used	Total Accuracy	Accuracy per sender class	Top Ten Features by Importance

SVM	2362	85%	Class 1: 76% Class 2: 85% Class 3: 94% Class 4: 67% Class 5: 88%	Sid, Party, Tories, Most, Labor, Leader, Xpress, Cingular, Conservatives, polls

Table 9: Classification accuracy results for linear SVM after feature filtering

SVM Classification with Supervised Feature Selection

For supervised feature selection using SVM, we choose to implement the linear kernel as we determined in the previous section that it has a high classification accuracy while being the least computationally intensive. We then fit the linear SVM classifier to the training data, then select the top {100, 200, 300, 400, 500} features based on importance. Using these features, we again fit a linear SVM classifier and then apply the classifier on the test set. As shown in *Table 10*, the highest test accuracy among these models is 56%, obtained with the top 200 features.

p (number of features)	Accuracy
p = 100	53.86%
p = 200	**56.00%**
p = 300	54.01%
p = 400	55.00%
p = 500	53.51%

Table 10: 5-fold CV Test Accuracy Scores for SVM with Supervised Feature Selection

Since the test accuracy drops after supervised feature selection, our final SVM classifier is obtained without supervised feature selection.

Part 3: K-means Clustering

Extracting the 100 word features with highest importance selected from the 2,377 feature matrix determined through feature selection and optimal Random Forest classifier in part two, we proceed to use the K-means clustering algorithm on these top features. We specifically implement the 5-mean clustering algorithm, as we have five different classes of senders, and run the algorithm multiple times to observe the set of predictions which have the least within-cluster sum of squares across all five clusters. Implementing 5-Means Clustering with ten different initial clustering assignments (see *Table 11*), the lowest within-cluster sum of squares for all clusters of senders was 338, 552 and had a classification accuracy rate of approximately 43%.

378954	338552	407536	338552	378954
389721	350244	338552	367948	350244

Table 11: Within-cluster sum of squares when performing 5-means clustering with different initial cluster assignments

Hence, the K-Means algorithm is minimally effective in predicting sender classification, and misclassifies a large percentage of each senders' original emails, especially senders one and four, as seen in *Table 12*.

	Predicted 1	Predicted 2	Predicted 3	Predicted 4	Predicted 5
Actual 1	14	127	541	0	3
Actual 2	30	312	679	0	2
Actual 3	5	112	1123	1	0
Actual 4	1	22	194	58	0
Actual 5	32	113	130	0	6

Table 12: Confusion Matrix for 5-means clustering using the cluster assignment with the lowest within-cluster sum of squares

Part 4: Choosing the Final Model

After determining the final feature matrices for the SVM, Random Forest and K-Means Clustering Classifiers with their corresponding classification accuracy rates, we proceed to select a final model and classifier to predict on our test data set. First, however, we decide to eliminate the 5-Means Clustering algorithms from consideration as it returned the lowest classification accuracy, 43%, when compared to the remaining two classifiers.

Next, among the SVM and Random Forest classifiers, we analyze the final 5-fold CV classification accuracy rate using the tuned parameters for both. The linear SVM classifier with an optimal cost $c = 0.01$ using the 2, 362 feature matrix returned the highest CV classification accuracy of 77 %. This was the highest accuracy rate among all SVM classifiers and tested models. Our Random Forest classifier, using the feature matrix with 2,377 features and tuning parameters $\frac{P}{2}$ and 300 for *mtry (m)* and *ntree (B)*, respectively, returned a CV classification accuracy rate of approximately 78.12 %, one of the highest among all Random Forest classifiers and tested feature matrices.

Hence, we choose the Random Forest classifier using the top 2,377 features and parameters $m = \frac{P}{2}$ and $B = 300$ as our final model for email classification since it had the highest test predication accuracy rate overall. We then base our final predictions for the test set using this model and classifier, and would except our actual test classification accuracy rate to be near 80%, on average.

Conclusion:

At the conclusion of this text-classification project, not only were we able to determine a final model and classifier which produced an approximate 80% test classification accuracy rate, but we were also able to learn about and experiment with many new implementation methods in order to improve classification accuracy.

In part one, in addition to learning and using the Bag of Words frequency determiner to create our feature matrix, we also implemented the TF-DIF and Word2Vec methods for feature creation and filtering and thus experimented with another branch of text-classification, Topic Modeling. Moreover, we learned that punctuation, numerical values and foreign words can also be a critical component for text classification.

Next, in part two, we developed our own feature selection algorithms to select features pertinent across all or certain senders to both reduce the dimension of our feature size and improve sender classification accuracy. The most revealing aspect of the project also came in part two where we used supervised feature selection with a Random Forest Classifier to observe that the top 500 features across all 3,505 emails, instead of 7, 232 features initially in part one, could be similarly used to calculate a high classification accuracy rate. Determining a model which had low dimension for feature size. yet high sender classification accuracy was the most challenging component of the project itself.

Overall, the project was an enlightening experience that increased our understanding of the variety of statistical learnings methods which can be implemented to effectively summarize and model previously observed data in order to make conclusions about future, unseen data sets. We would like to thank Professor Nusrat Rabbee and our Graduate Student Instructor, Omid Solari for all their guidance throughout the course of the class and project. This project would not have been possible without their support.

References:

1. "Feature Extraction and Transformation - RDD-based API." *Apache Spark*. Apache Spark, n.d. Web. 02 Dec. 2016. <http://spark.apache.org/docs/latest/mllib-feature-extraction.html>.

2. Rehurek, Radim. "Deep Learning with Word2Vec." *Gensim: Topic Modelling for Humans*. N.p., n.d. Web. 02 Dec. 2016. <https://radimrehurek.com/gensim/models/word2vec.html>.