Machine Learning Engineer Nanodegree

Capstone Project

Camille Wilkens
March 10th, 2017

I. Definition

Project Overview

This project is based on identifying sentiment analysis specifically related to financial fraud within the Enron email corpus. Sentiment analysis is a way to evaluate emails to determine if the email is favorable, unfavorable, or neutral, and to what degree the potential for fraud is likely within the email. The Enron email corpus is a collection of over 500,000 publicly available email messages that were generated by 150 employees of the Enron corporation.[iv] These emails were obtained during the investigation of Enron's collapse in 2001. This analysis will focus on individuals that have been identified as "Persons of Interest" (POIs) which means individuals that have been convicted, pleaded guilty, cases were overturned, took settlements and/or indicted in the Enron scandal.[i] The words used within the Enron email corpus will be compared to a financial sentiment dictionary[v] to detect the potential of financial fraudulent activities within email communications. A financial sentiment dictionary is a list of words that have a negative, positive, or other implication within the finance domain. These finance "keywords" will provide a more meaningful difference in how words are used within the emails to assist in identifying potential financial fraudulent activities. The objective of this analysis is to determine what percentage of the non-POI email messages contained indicators of financial fraudulent activities based on the fraudulent characteristics of the POI email messages. The assumption is that non-POI employees had insight into financial fraudulent behavior of the POI employee through email messages.

Problem Statement

In order to solve the problem of what percentage of the non-POI email messages contained indicators of financial fraudulent activities based on the fraudulent characteristics of the POI email messages several steps will need to be performed using classification and regression algorithms. Based on the characteristics of the problem where I need to classify two sets of email messages, non-POI email messages that contain indicators of financial fraudulent activities or non-POI messages that do not contain indicators of financial fraudulent activities based on the fraudulent characteristics of the POI email messages can be solved with Classification models.

I would also like to establish a relationship between POI/Non-POI and the number or percentage of financial negative words used in each email message. This type of counting or the use of percentage of financial negative words can be solved with Regression models. A similar approached documented in "***Featurizing Text: Converting Text into Predictors for Regression Analysis***" [ii], explains the process of assigning a positive or negative label to a text that captures the text's opinion towards its main subject matter.

Strategy to achieve results:

1. Acquire Enron Email Corpus
2. Acquire Financial Sentiment Dictionary – This project will focus only on the Negative Financial & Litigation Financial Word lists from the Financial Sentiment Dictionary. [vi] [vii] This dictionary will provide the negative financial and litigation financial word lists used to indicate if an email message has financial fraudulent characteristics
3. Acquire POI email list – This email address list will be compared to the email message sender and will be marked as "POI" email message if these email addresses match.
4. Perform Data Exploration on the Enron Email Corpus, Financial Sentiment Dictionary files and the POI email list datasets. Identify any data abnormities, outliers and missing columns
5. Extract relevant email objects from the Enron Email Corpus [To, From, Date, Message Body, etc] and create missing fields, cleanse data identified in the Data Exploration Step 4 and remove outliers that were identified.
6. Extract email messages between 1999 & 2002
7. Perform keyword matching with the Financial Sentiment Dictionary and compare email messages with the financial keywords
8. Create new Features based on the keyword matching:
   - # of Financial Negative words found in email message
   - # of Financial Litigation words found in email message
   - Total words found in email message
   - % of Bad Words Found in email message (# of Financial Negative words found in email message/Total Words Found in email message)

- % of Litigation Words Found (# of Litigation Negative words found in email message/Total Words Found in email message)

9. Prepare data for Classification Models - If the percentage of financial negative words within the email message is greater than zero a new classification label will be marked as "YES_Negative" and if the percentage of financial negative words within the email message is equal to zero the label will be marked as "No_Negative.
10. Create Test & Training Sets for Classification Models - training test sets will be performed on email messages marked as "POI" (X_Train) and the testing set will be performed on the "non-POI" email messages (X_Test).
11. Develop classification models, test on prepared dataset and identify the best fit model based on the accuracy score
12. Fine tune Classification Model parameters as required, analyze the accuracy scores form each Classification Model's test sets and adjust the model's parameters until there is no further improvement in accuary score
13. Create Test & Training Sets for Regression Models - training test sets will be performed on email messages marked as "POI" (X_Train) and the testing set will be performed on the "non-POI" email messages (X_Test).
14. Develop regression models, test on prepared dataset and identify the best fit model based on the accuracy score
15. Fine tune Regression Model parameters as required, analyze the R2 scores from each Regression Model's test sets and adjust the model's parameters until there is no further improvement in accuracy score
16. Use the best regression & classification model to see if justifies the present of financial negative words in the non-POI email messages that are also present in POI email messages.

Metrics

In order to perform sentiment analysis, I will be using multiple classification and regression models. To measure the performance of the classification models, I will be using the accuracy score. The reasoning behind using the accuracy score versus another performance metric such as the F1 Score is that in sampling the training data (POI email messages) the dataset has a very balanced number (51%) of positive vs. negative examples. If this case had been reversed (the training set had a high unbalanced number of positive vs. negative examples), I would have used the F1 Score instead of the accuracy score. I will fit the models on the training dataset and then get the accuracy score of the testing datasets for each of the classification models. My baseline accuracy score will be defined in the **Benchmark** Section. I will also fine tune the model's parameters looking to achieve the best accuracy score for each model.

For the regression models, I will be using the R2 score to analyze how well the models are performing. The reason for considering the R^2 score is that it provides a measure of how well future samples are likely to be predicted by the models. The best possible score is 1.0 which perfectly predicts the target variable and it can be negative which indicates that the model is no better than one that simply predicts the mean of the target variable. I will fit the regression models on the training dataset, predict the models on the testing dataset and finally calculate the R2 Score on the testing dataset. My baseline score will be defined in the **Benchmark** Section. I will also fine tune the model's parameters looking to achieve the best R2 score for each model.

II. Analysis

Data Exploration

There will be 4 input datasets used for this project – Enron email corpus, list of financial negative words, list of financial litigation words, and a list of POI's email addresses.

1. The email dataset used for this project is the Enron email dataset from Kaggle. [iii] This data contains approximately 500,000 emails generated by employees of the Enron Corporation. It was obtained by the Federal Energy Regulatory Commission during its investigation of Enron's collapse. This is the May 7, 2015 Version of dataset, as published at https://www.cs.cmu.edu/~./enron/ - Data files copyright their respective owners.[iv]

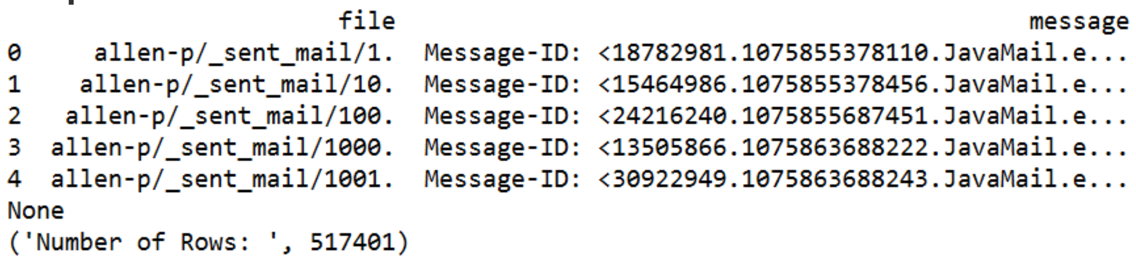
   Download Location: https://www.kaggle.com/wcukierski/enron-email-dataset

   The download file is called emails.csv and contains two columns: file & message

   file: contains the original directory and filename of each email. The root level of this path is the employee to whom the emails belong.
   message: contains the email text

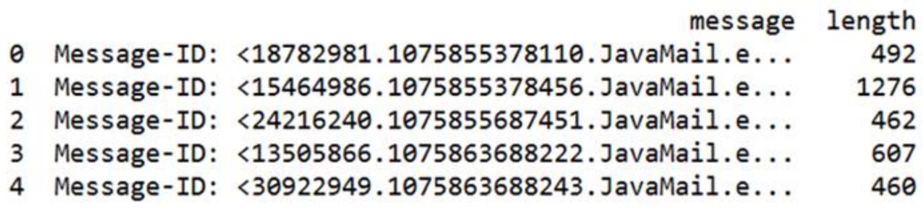
Total Number of Rows: 517,401

Sample Data:



```
                        file                                            message
0     allen-p/_sent_mail/1.  Message-ID: <18782981.1075855378110.JavaMail.e...
1    allen-p/_sent_mail/10.  Message-ID: <15464986.1075855378456.JavaMail.e...
2   allen-p/_sent_mail/100.  Message-ID: <24216240.1075855687451.JavaMail.e...
3  allen-p/_sent_mail/1000.  Message-ID: <13505866.1075863688222.JavaMail.e...
4  allen-p/_sent_mail/1001.  Message-ID: <30922949.1075863688243.JavaMail.e...
None
('Number of Rows: ', 517401)
```

Sample Length of Message Column:



```
                                         message  length
0  Message-ID: <18782981.1075855378110.JavaMail.e...     492
1  Message-ID: <15464986.1075855378456.JavaMail.e...    1276
2  Message-ID: <24216240.1075855687451.JavaMail.e...     462
3  Message-ID: <13505866.1075863688222.JavaMail.e...     607
4  Message-ID: <30922949.1075863688243.JavaMail.e...     460
```

Summary of Email Messages:

Total Count of Messages:	517,401
Unique Message Count:	517,401
Total Number of Messages <= 3,000 characaters	405,916
Total Number of Messages >3,000 characters:	111,241
Total Number of Messages > 2 million characters:	1

The initial assessment of this data has determined that the emails.csv file will need to be transformed from its original raw content. These transformations are necessary to extract the relevant email objects that are needed to create the missing features required for the Classification & Regression models and to create the feature transformation required for the Classification models. Additional columns such as Date, From, & Message Body will be extracted from the message columns. The "From" field is required to identify which messages are from a POI, and "Message Body" will be utilized in matching email messages to the Financial Sentiment Dictionary to determine the number of financial "Negative" words and the number of financial "Litigation" words used within an email message.

This assessment has also identified messages that contain an excessive amount of characters per message. 21.5% of the Enron Email corpus contained messages over 3,000 characters per message. In one case, the email message contained over 2 million characters. Upon further investigation, it was due to a very large attachment in the email message body. In the **Data Preparation** stage, these characters will be restricted to a set number of characters per message. This analysis will be focused on the first 3,000 characters in the message and the scope of this analysis will not include email attachments or extremely large email messages. Restricting the amount of characters used per message will not only help with the performance of the keyword matching but also remove any potential erroneous data that could be stored in these very large messages.

The financial negative words list that will be used for this analysis is from the Loughran and McDonald Financial Sentiment Dictionary[v]. This dataset was downloaded from GitHub.[vi]

Dataset Name: Fin-Neg – This file provides a negative financial word list and contains two columns: word & year. "Word" is the negative financial word and "Year" is the year it was added to the dictionary.

Download Location: https://github.com/jperla/sentiment-data/blob/master/finance/LoughranMcDonald_Negative.csv

Sample Data:

```
    ABANDON  2009
  ABANDONED  2009
 ABANDONING  2009
ABANDONMENT  2009
ABANDONMENTS 2009
   ABANDONS  2009
```

Total Number of Rows: 2349

```
('Number of Rows: ', 2349)
```

The column "word" is required from the Fin-Neg file to match the financial negative words with the words found in the message body of the email. This data will be used to generate the feature "Bad Word Count" which is the number of times a financial negative word was found in the email and will be also used to create the feature called "Percentage Bad Words Count" which is the percentage of negative financial words found within all words used in the email message body (Bad Word Count/Total Word Count).

2. The financial litigation words list that will be used for this analysis is from the Loughran and McDonald Financial Sentiment Dictionary[v]. This dataset was downloaded from GitHub.[vii]

   Dataset Name: Fin_Lit – These are a list of words reflecting a propensity for legal contest and contain two columns: word & year. Word is the financial litigation word and Year is the year it was added to the dictionary.

   Download Location: https://github.com/jperla/sentiment-data/blob/master/finance/LoughranMcDonald_Litigious.csv

   Sample Data:

```
0  ABOVEMENTIONED  2011
1        ABROGATE  2009
2       ABROGATED  2009
3       ABROGATES  2009
4      ABROGATING  2009
```

   Total Number of Rows: 871

```
('Number of Rows in Fin-Lit: ', 871)
```

The column "word" is required from the Fin_Lit file to match the financial litigation words with the words found in the message body of the email. This data will be used to generate the feature "Lit Word Count" which is the number of times a financial litigation word was found in the email message body and will also be used to create the feature called "Percentage Lit Words Count" which is the percentage of financial litigation words found within all words used in the email message body (Lit Word Count/Total Word Count).

3. POI email address list was generated from the Final Project of Udacity's Introduction to Machine Learning Class - poi_email_addresses.py.[i]

   Sample Data:

```
0  kenneth_lay@enron.net
1  kenneth_lay@enron.com
2   klay.enron@enron.com
3  kenneth.lay@enron.com
4         klay@enron.com
```

Total Number of Rows: 91

```
('Number of Rows: ', 91)
```

The POI email address list will be compared to the email address of the sender – "From" for each email message. If these email addresses match, this email message will be marked as a "POI" email message.

Exploratory Visualization

Under exploratory visualization, I focused on understanding the key fields of the email dataset. These visualizations provided insight into the number of messages and the number of financial negative words over a period of time. ***Figure 1***, provides a high level overview, it is important to understand how many messages occurred by year. This will determine how many email messages the models will be processing but also displays when the largest number of email communications occurred which was between 2000 and 2001.

Figure 1

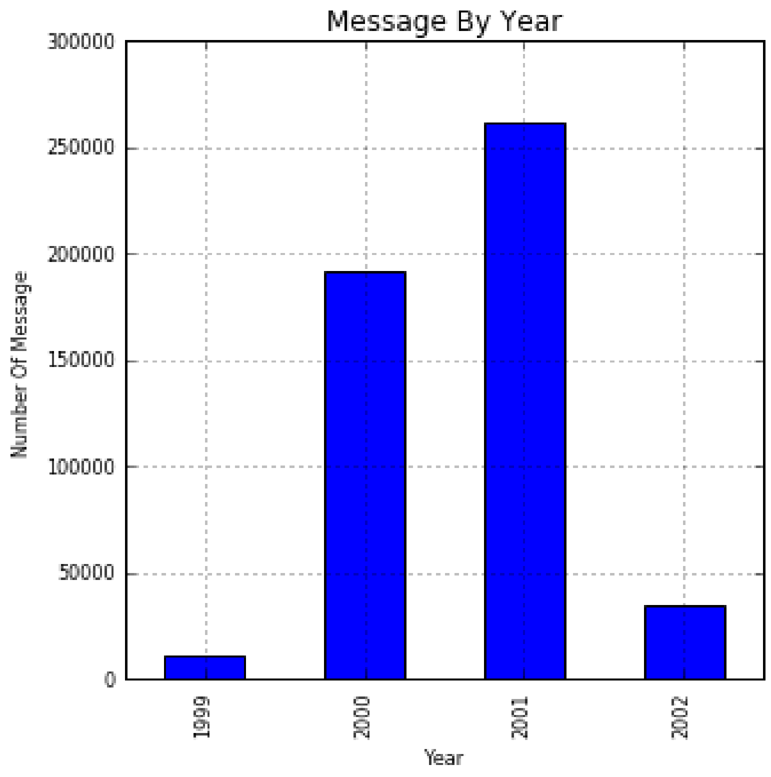


Figure 2, breaks this down a little further, by breaking the number of email messages by month & year. This graph shows three distinctive bell curves of higher

than average email messages - [2000-8 through 2000-9], [2001-7], and [2001-12 through 2002-1].

Figure 2

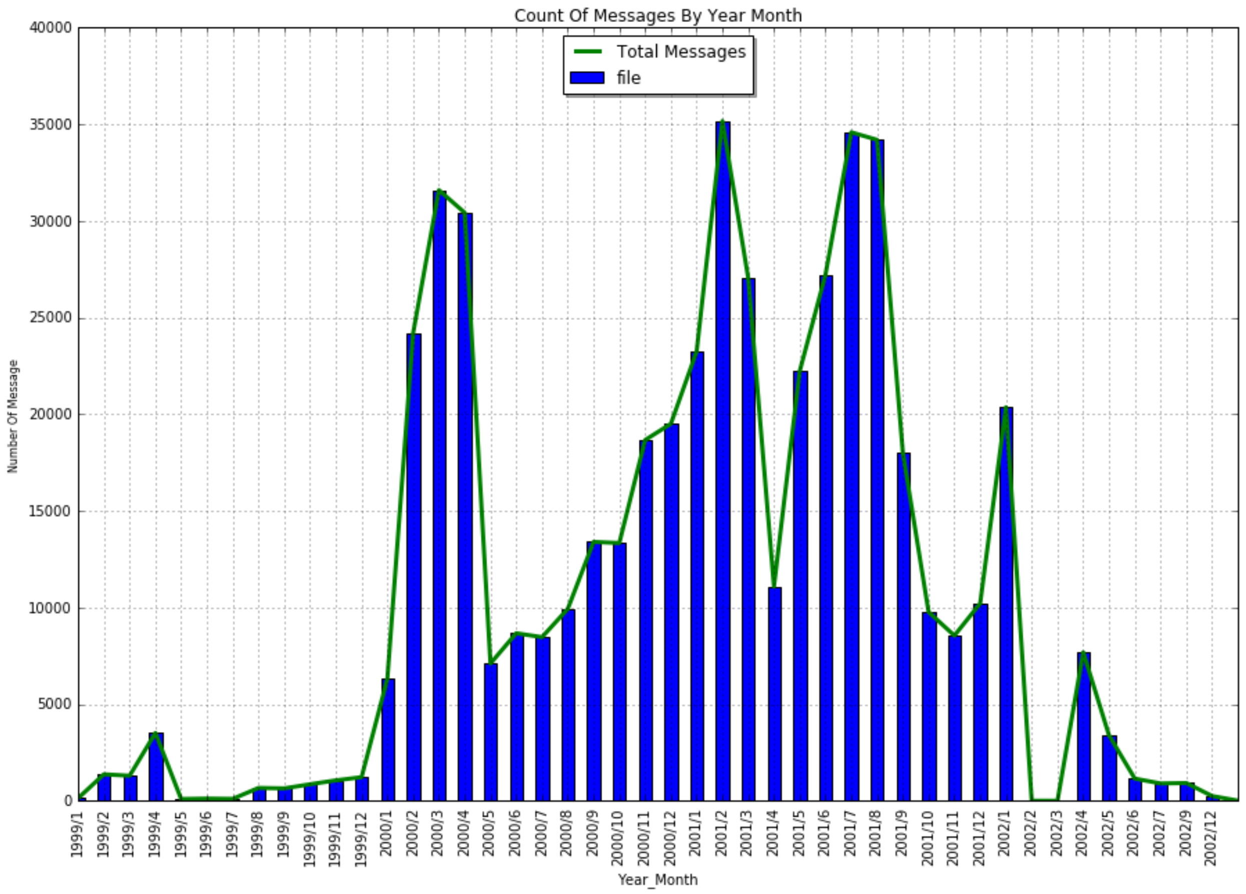


Figure 3, provides insight into how many email messages were identified containing financial negative words in the email message body. It's interesting that this graph also contains three distinctive bell curves; however the timeframes are not identical. The three peaks for the total number of financial negative words used in the email messages where from [2000-3] which is 5 months before the first peak of the increased number of emails [2000-8], second peak was [2000-12] which is 1 year before the second peak of increased total emails [2001-12] and the third peak was [2001-7 through 2001-8] which is 5 months before the third peak of increased total number of emails [2001-12 through 2002-1].

Figure 3

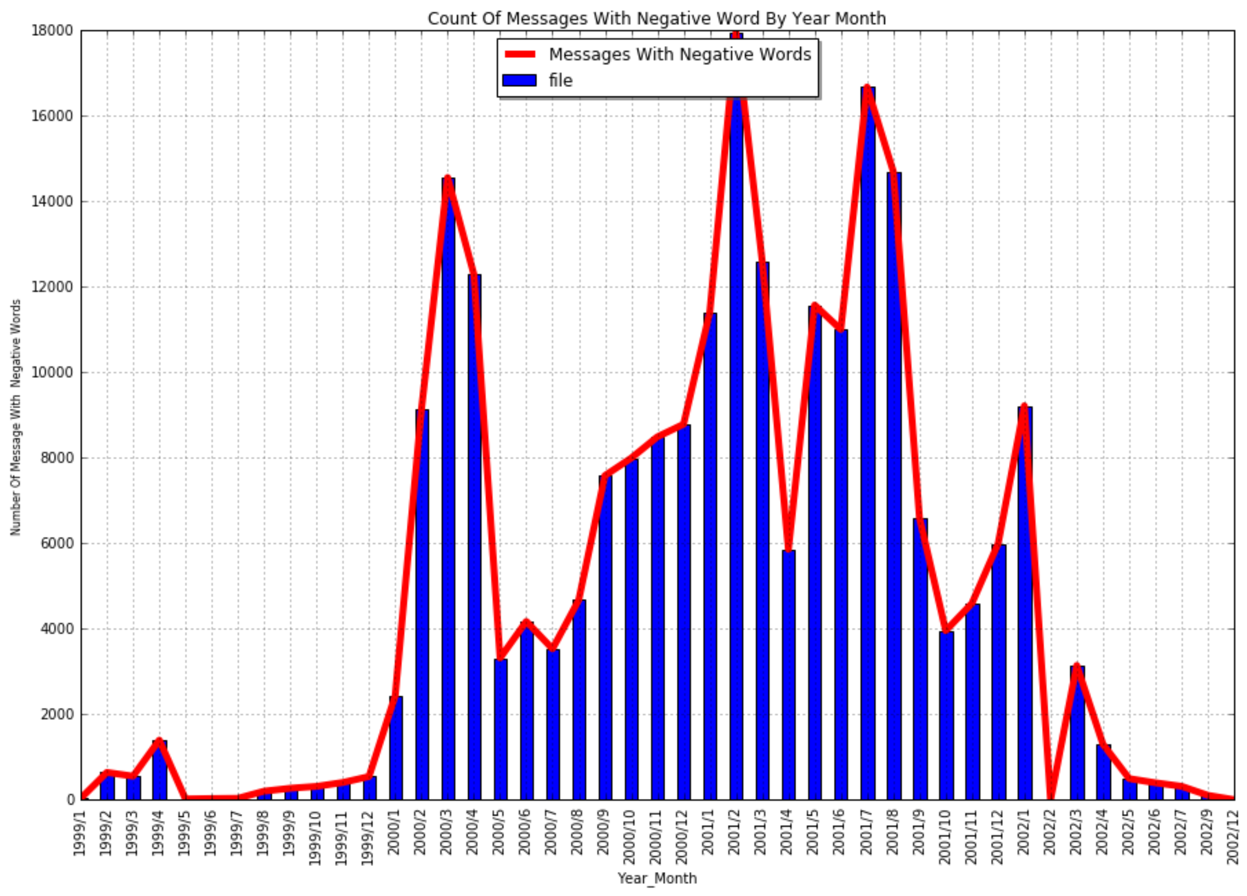
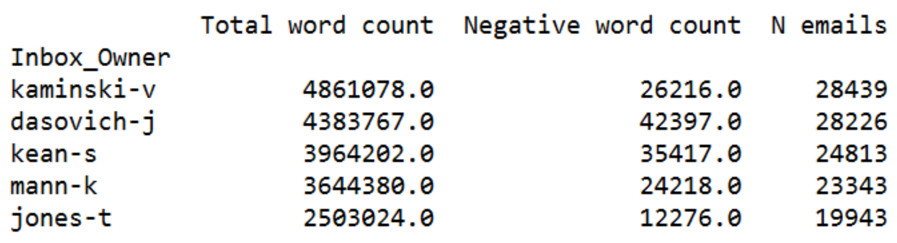


Figure 4, this table displays a high level summary of the top Inbox Owner's containing the highest number of email messages, the total number of words used in the email messages, and the number of financial negative words used in the total email messages.

Figure 4


```
             Total word count  Negative word count  N emails
Inbox_Owner
kaminski-v          4861078.0              26216.0     28439
dasovich-j          4383767.0              42397.0     28226
kean-s              3964202.0              35417.0     24813
mann-k              3644380.0              24218.0     23343
jones-t             2503024.0              12276.0     19943
```

This insight helps indicate that my original assumption that non-POI employees had insight into financial fraudulent behavior of the POI employee through email messages could be correct.

# of POI Email Messages:	5,055
# of Non-POI Email Messages:	493,410
# of POI Email Messages with Financial Negative Words:	2581
% of POI Email Messages with Negative words:	51.0%

Algorithms and Techniques

For this analysis, I will be using both Classification and Regression models from scikit-learn[viii] to predict that a percentage of the emails contained in the non-POI email messages contained indicators of financial fraudulent activities based on the fraudulent characteristics of the POI email messages. For the classification models, I will be applying three algorithms that are commonly used for email text classifications: Support Vector Machines (SVM) – Support Vector Classification (SVC), KNeighborsClassifier, and Quadratic discriminant analysis (QDA). For the regressions models, I will be applying Decision Tree Regressor, Bagging Regression & Gradient Boosting Regression

Classification Models:

For the classification models, I will transform the number of negative words within an email message into a label called Classifcation1 with two types of classification - "YES_Negative" (Email Message contains Financial Negative words) and "NO_negative" (Email Message does not contain financial negative words).

Split the data into Training & Testing subsets: Training set is all the email messages where ('POI') == True, and the Testing set is where all the email messages = ('POI') == False.

The inputs/features I will be using in the Classification Models:

Total_Words_count - Total words found in email message
Bad_Words_count - # of Financial Negative words found in email message
Lit_Words_count - # of Financial Litigation words found in email message

Percentage_Lit_Words_count - # of Financial Litigation words (Lit_Words_count) found in email message/Total Word Count Found in email message

The output/target variable will be "Classification1"

SVC[ix]

Description: SVM is a supervised learning model that given a set of training data builds a model that assigns unseen data into one category or the other. It is a distance-based method that defines the margins to determine which points should be classified in which category. Support vectors are the data points that lie closest to the decision surface. The goal is to have the widest margin possible between the support vectors; this is called the margin maximization. Unseen data is then mapped and predicted to belong to one category or the other based on which side of the gap they fall on.

Good Candidate for Problem: SVC is a good candidate for this data because of the number of features and size of the dataset. SVC also provides very accurate classifiers which will maximize the correctness.

A potential disadvantage with SVC could be that it's hard to scale for samples beyond 10,000.

Initially, I will be using SVC's default parameters and will make adjustments as required i.e. poor performing model, if necessary I will Fine tune the model using GridSearchCV[x].

Default Parameters:
(*C=1.0*, *kernel='rbf'*, *degree=3*, *gamma='auto'*, *coef0=0.0*, *shrinking=True*, *probability=False*, *tol=0.001*, *cache_size=200*, *class_weight=None*, *verbose=False*, *max_iter=-1*, *decision_function_shape=None*, *random_state=None*)

To measure the success of this model, I will be using the Accuracy Score.[xi]

KNeighborsClassifier[xii]

Description: KNeighborsClassifier is a supervised learning model which uses pattern recognition to classify objects based on the closest training examples in the feature space.

Good Candidate for Problem: KNeighborsClassifier is a good candidate for this data because of the size of the training set and works well on basic recognition problems. KNeighborsClassifier is commonly used for email classifications. I will also be using a smaller training set (email messages marked as POI) and KNeighborsClassifier performs well with a smaller training set.

Initially, I will be using KNeighborsClassifier's default parameters and will make adjustments as required i.e. poor performing model; if necessary I will fine tune the model using GridSearchCV[x]

Default Parameters:
(*n_neighbors=5*, *weights='uniform'*, *algorithm='auto'*, *leaf_size=30*, *p=2*, *metric='minkowski'*, *metric_params=None*, *n_jobs=1*, ***kwargs*)

To measure the success of this model, I will be using the Accuracy Score. [xi]

QDA[xiii]

Description: A supervised learning classifier with a quadratic decision boundary, generated by fitting class conditional densities to the data and using Bayes' rule. The model fits a Gaussian density to each class.

Good Candidate for Problem: QDA is a good candidate for this data because they have closed-form solutions that can be easily computed, are inherently multiclass, and have proven to work well in practice. Also there are no parameters to tune for these algorithms.[xiv]

Initially, I will be using QDA's default parameters and will make adjustments as required i.e. poor performing model; if necessary I will fine tune the model using GridSearchCV[xxiii]

Default Parameters: (*priors=None*, *reg_param=0.0*, *store_covariances=False*, *tol=0.0001*)

To measure the success of this model, I will be using the Accuracy Score.[xi]

Regression Models

For the regression models, I will transform the number of occurrences of financial negative words into a numerical number called "Percentage of Negative Words" which is the number of financial negative words in an email message divided by total number of words in the email message.

Split the data into Training & Testing subsets: Training set is all the email messages where ('POI') == True, and the Testing set is where all the email messages = ('POI') == False.

The inputs/features I will be using in the Regression Models:

Total_Words_count - Total words found in email message
Bad_Words_count - # of Financial Negative words found in email message
Lit_Words_count - # of Financial Litigation words found in email message
Percentage_Lit_Words_count - # of Financial Litigation words (Lit_Words_count) found in email message/Total Word Count Found in email message

The target variable will be the Percentage of Negative Words.

Decision Tree Regressor [xv]

Description: Decision trees Regressor is when the predicted outcome can be considered a real number i.e. percentage of negative words.

Good Candidate for Problem: Decision Tree Regressor is simple to use, good for fast exploration, works well for prediction type of problems and output is easy to interpret.

Initially, I will be using the Decision Tree Regressor's default parameters and will make adjustments as required i.e. poor performing model, if necessary I will fine tune the model using GridSearchCV[x]

Default Parameters:
(*criterion='mse'*, *splitter='best'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features=None*, *random_state=None*, *max_leaf_nodes=None*, *min_impurity_split=1e-07*, *presort=False*)

To measure the success of this model, I will be using the R2 Score. [xvi]

Bagging Regressor [xvii]

Description: Bagging Regressor is an ensemble meta-estimator that fits base regressors each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction[xvii].

Good Candidate for Problem: Bagging Regressor reduces the variance of a black-box estimator (e.g., a decision tree), by introducing randomization into its construction procedure and then making an ensemble out of it.[xvii]

Initially, I will be using the Bagging Regressor's default parameters and will make adjustments as required i.e. poor performing model; if necessary I will fine tune the model using GridSearchCV.[x]

Default Parameters:
(*base_estimator=None*, *n_estimators=10*, *max_samples=1.0*, *max_features=1.0*, *bootstrap =True*, *bootstrap_features=False*, *oob_score=False*, *warm_start=False*, *n_jobs=1*, *random_s tate=None*, *verbose=0*)

To measure the success of this model, I will be using the R2 Score.[xvi]

Gradient Boosting Regressor [xviii]

Description: Gradient Boosting Regressor is ensemble learning method and predicts by combining the outputs from individual trees.

Good Candidate for Problem: The training set is small, because Gradient Boosting Regressor builds trees one at a time and each new tree helps to correct errors made by the previously trained tree, it can be very costly to run on the large training sets. It also has better predictive power.

Initially, I will be using the Gradient Boosting Regressor 's default parameters and will make adjustments as required i.e. poor performing model; if necessary I will fine tune the model using GridSearchCV.[x]

Default Parameters:
(*loss='ls'*, *learning_rate=0.1*, *n_estimators=100*, *subsample=1.0*, *criterion='friedman_mse'*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_depth=3*, *min_impurity_split=1e-*

07, *init=None*, *random_state=None*, *max_features=None*, *alpha=0.9*, *verbose=0*, *max_leaf_nodes=None*, *warm_start=False*, *presort='auto'*)

To measure the success of this model, I will be using the R2 Score.[xvi]

Benchmark

To determine if the hypothesis listed in the **Problem Statement** is true, that a percentage of the email messages contained in the non-POI email messages contained indicators of financial fraudulent activities based on the fraudulent characteristics of the POI email messages, I will be looking for an accuracy score greater than .5 in the classification models and an R2 score greater than .5 for the regression models. The reasoning behind this benchmark is based on the following assumptions:

1. An organization's attitude and activities are driven by the senior management and publicly available news
2. Any email message which is financial negative for the organization are only communicated by the senior management (POIs) to only a few confidential employees
3. These confidential employees will further communicate the information to their close friends
4. The information mentioned above is again passed to the next level of friends and employees, and etc
5. Based on the chain of information flowing, I am assuming that after a certain period of time 50% of the corporation will be aware of the financial negative information. Based on this, I will look for a model with an accuracy score of 0.5 or greater in the classification models and a R2 score of .5 or greater in the regression models.

Also to validate my assumptions, after sampling the email messages identified as POI, I was able to determine that roughly 50% of these email messages contained financial negative words used in the email messages and looking to achieve this same percentage in the non-POI email messages.

III. Methodology

Data Preprocessing

There are three steps that are required in the data preprocessing phase. These steps are performed not only for performance purposes for the keyword matching of negative financial and financial litigation words but also necessary for the feature transformations that required for the classification and regression algorithms. During the **Data Exploration** phase, abnormalities in the data were found in the email messages with the emails containing special characters or punctuations and messages containing excessive amount of characters within the email messages.

Step 1: Removed special characters from the original message field. These special characters were identified and replaced with ‘ ‘. For example, any occurrence of “- -“ was replaced with null value.

During Step 1, the column message was also restricted to 3,000 characters per message. As stated earlier in the **Data Exploration** phase, this analysis will be focused on the first 3,000 characters in the message and the scope of this analysis will not include email attachments. Restricting the amount of characters used per message will not only help with the performance of the keyword matching but also remove any potential erroneous data that may be in these very large messages. In order to achieve this, I have used a parameter dtype=(str,3000) in the pandas read_csv method.

Step 2: Extracted relevant email objects from the messages field. Message-ID, Subject, X-From, X-To, X-cc, X-bcc, X-Origin, Message_Body, To, From & Date fields were extracted from the original messages field and all words were made lower case which is required for the word matching of the POI email addresses and financial negative & litigation words. The POI email address list will be compared to the email address of the sender – “From” for each email message. Created new field called POI and if these email addresses match, this email message will be marked as a “POI” email message- ('POI') == True and marked ('POI') == False if these email messages do not match.

Lastly, in the final dataframe, all non-required fields were dropped – Subject, X-From, X-From, X-To, X-cc, X-bcc, and X-Origin and only email messages between 1999 – 2002 were included, all other emails messages were removed (1,219 email messages) as per the project scope.

Created new fields for future use, these fields will be updated during the keyword matching – Step 3:
Total_Words_count - Total words found in email message
Bad_Words – The Financial Negative word(s) found in the email message
Bad_Words_count - # of Financial Negative words found in email message
Percentage_Bad_Words_count - % of Bad Words Found in email message (# of Financial Negative words found in email message/Total Words Found in email message)

Lit_Words – The financial litigation word(s) found in email message
Lit_Words_count - # of Financial Litigation words found in email message
Percentage_Lit_Words_count - # of Financial Litigation words (Lit_Words_count) found in email message/Total Word Count Found in email message

Step 3: Performed keyword matching with the Financial Sentiment Dictionary - LoughranMcDonald_Negative & LoughranMcDonald_Litigious files[v] (included in the scope of this project).

Updated the following fields with the Financial Negative and Litigation Words found in each email message:

Total_Words_count - Total words found in email message
Bad_Words – The Financial Negative word(s) found in the email message
Bad_Words_count - # of Financial Negative words found in email message
Percentage_Bad_Words_count - % of Bad Words Found in email message (# of Financial Negative words found in email message/Total Words Found in email message)
Lit_Words – The financial litigation word(s) found in email message
Lit_Words_count - # of Financial Litigation words found in email message
Percentage_Lit_Words_count - # of Financial Litigation words (Lit_Words_count) found in email message/Total Word Count Found in email message

Process for Step 3:

1. For each email message, the email message body is split and the total words in the message body of the email message is counted and stored in the Total_Words_count field.

2. The splitted word is then compared to the financial negative (bad word) dataframe(dataframe created from the LoughranMcDonald_Negative.csv file) and the number words of that match the bad word dataframe are counted and stored in the Bad_Words_count field, the actual financial negative word is stored in the Bad_words field.

3. The Percentage_Bad_Words_count is calculated by dividing (Bad_word_count/Total_words_count) * 100

4. The splitted word is then compared to the financial litigation word (lit word) dataframe(dataframe created from the oughranMcDonald_Litigious.csv file) and the number words of that match the lit words dataframe are counted and stored

in the Lit_Words_count field, the actual financial negative word is stored in the Lit_words field.

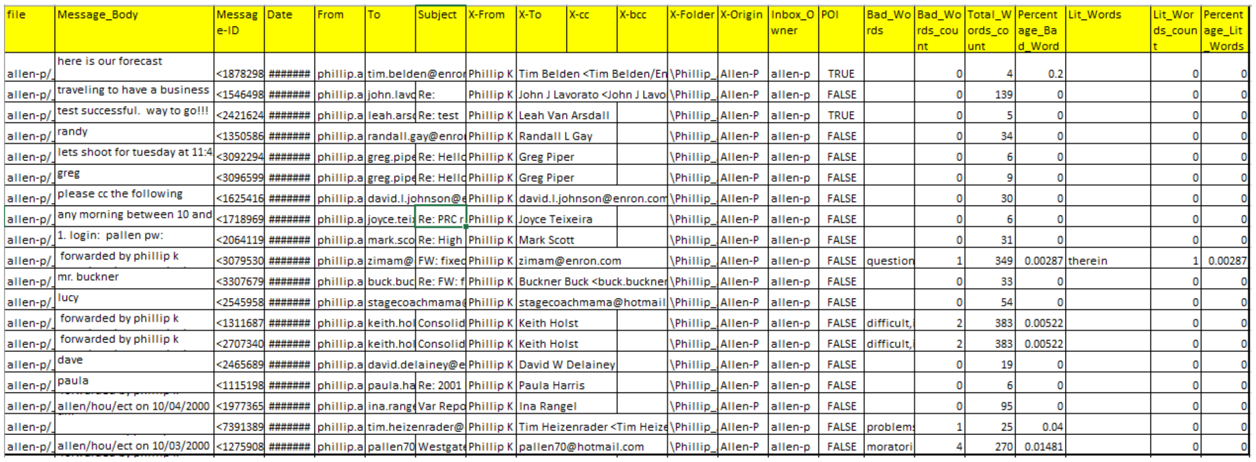
5. The Percentage_Lit_Words_count is calculated by dividing (Lit_word_count/Total_words_count) * 100

```
('shape of the dataframe:', (516182, 22))
('file', 516182)
('Message_Body', 283835)
('Message-ID', 516182)
('Date', 223876)
('From', 20293)
('To', 58832)
('Subject', 158030)
('X-From', 27639)
('X-To', 73667)
('X-cc', 32545)
('X-bcc', 122)
('X-Folder', 5455)
('X-Origin', 309)
('Inbox_Owner', 150)

('POI', 2)
('Bad_Words', 32603)
('Bad_Words_count', 24)
('Total_Words_count', 615)
('Percentage_Bad_Words_count', 2940)
('Lit_Words', 8676)
('Lit_Words_count', 21)
('Percentage_Lit_Words_count', 2000)
```

Figure 5 shows a subset of the final dataset (email_csv_with_bad_words.csv) after merging the POI indicator, counting of financial negative words and calculating the percentage of financial negative words.

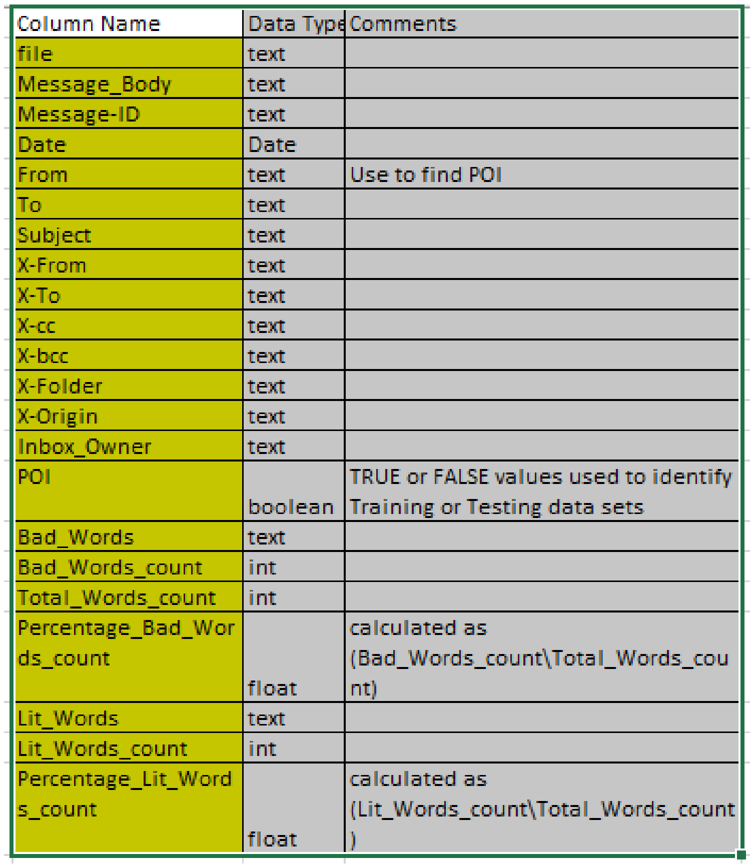
Figure 5



file	Message_Body	Message-ID	Date	From	To	Subject	X-From	X-To	X-cc	X-bcc	X-Folder	X-Origin	Inbox_Owner	POI	Bad_Words	Bad_Words_count	Total_Words_count	Percentage_Bad_Word	Lit_Words	Lit_Words_count	Percentage_Lit_Words
allen-p/_	here is our forecast	<1878298	#######	phillip.a	tim.belden@enron		Phillip K	Tim Belden <Tim Belden/En			\Phillip_	Allen-P	allen-p	TRUE		0	4	0.2		0	0
allen-p/_	traveling to have a business	<1546498	#######	phillip.a	john.lavc	Re:	Phillip K	John J Lavorato <John J Lavo			\Phillip_	Allen-P	allen-p	FALSE		0	139	0		0	0
allen-p/_	test successful. way to go!!!	<2421624	#######	phillip.a	leah.arsc	Re: test	Phillip K	Leah Van Arsdall			\Phillip_	Allen-P	allen-p	TRUE		0	5	0		0	0
allen-p/_	randy	<1350586	#######	phillip.a	randall.gay@enron		Phillip K	Randall L Gay			\Phillip_	Allen-P	allen-p	FALSE		0	34	0		0	0
allen-p/_	lets shoot for tuesday at 11:4	<3092294	#######	phillip.a	greg.pipe	Re: Hello	Phillip K	Greg Piper			\Phillip_	Allen-P	allen-p	FALSE		0	6	0		0	0
allen-p/_	greg	<3096599	#######	phillip.a	greg.pipe	Re: Hello	Phillip K	Greg Piper			\Phillip_	Allen-P	allen-p	FALSE		0	9	0		0	0
allen-p/_	please cc the following	<1625416	#######	phillip.a	david.l.johnson@e		Phillip K	david.l.johnson@enron.com			\Phillip_	Allen-P	allen-p	FALSE		0	30	0		0	0
allen-p/_	any morning between 10 and	<1718969	#######	phillip.a	joyce.tei	Re: PRC r	Phillip K	Joyce Teixeira			\Phillip_	Allen-P	allen-p	FALSE		0	6	0		0	0
allen-p/_	1. login: pallen pw:	<2064119	#######	phillip.a	mark.sco	Re: High	Phillip K	Mark Scott			\Phillip_	Allen-P	allen-p	FALSE		0	31	0		0	0
allen-p/_	forwarded by phillip k	<3079530	#######	phillip.a	zimam@	FW: fixed	Phillip K	zimam@enron.com			\Phillip_	Allen-P	allen-p	FALSE	question	1	349	0.00287	therein	1	0.00287
allen-p/_	mr. buckner	<3307679	#######	phillip.a	buck.buc	Re: FW: f	Phillip K	Buckner Buck <buck.buckner			\Phillip_	Allen-P	allen-p	FALSE		0	33	0		0	0
allen-p/_	lucy	<2545958	#######	phillip.a	stagecoachmama@		Phillip K	stagecoachmama@hotmail			\Phillip_	Allen-P	allen-p	FALSE		0	54	0		0	0
allen-p/_	forwarded by phillip k	<1311687	#######	phillip.a	keith.hol	Consolid	Phillip K	Keith Holst			\Phillip_	Allen-P	allen-p	FALSE	difficult,	2	383	0.00522		0	0
allen-p/_	forwarded by phillip k	<2707340	#######	phillip.a	keith.hol	Consolid	Phillip K	Keith Holst			\Phillip_	Allen-P	allen-p	FALSE	difficult,	2	383	0.00522		0	0
allen-p/_	dave	<2465689	#######	phillip.a	david.delainey@e		Phillip K	David W Delainey			\Phillip_	Allen-P	allen-p	FALSE		0	19	0		0	0
allen-p/_	paula	<1115198	#######	phillip.a	paula.ha	Re: 2001	Phillip K	Paula Harris			\Phillip_	Allen-P	allen-p	FALSE		0	6	0		0	0
allen-p/_	allen/hou/ect on 10/04/2000	<1977365	#######	phillip.a	ina.range	Var Repo	Phillip K	Ina Rangel			\Phillip_	Allen-P	allen-p	FALSE		0	95	0		0	0
allen-p/_		<7391389	#######	phillip.a	tim.heizenrader@		Phillip K	Tim Heizenrader <Tim Heize			\Phillip_	Allen-P	allen-p	FALSE	problem	1	25	0.04		0	0
allen-p/_	allen/hou/ect on 10/03/2000	<1275908	#######	phillip.a	pallen70	Westgate	Phillip K	pallen70@hotmail.com			\Phillip_	Allen-P	allen-p	FALSE	moratori	4	270	0.01481		0	0

Figure 6 displays the datatype details.

Figure 6



Column Name	Data Type	Comments
file	text	
Message_Body	text	
Message-ID	text	
Date	Date	
From	text	Use to find POI
To	text	
Subject	text	
X-From	text	
X-To	text	
X-cc	text	
X-bcc	text	
X-Folder	text	
X-Origin	text	
Inbox_Owner	text	
POI	boolean	TRUE or FALSE values used to identify Training or Testing data sets
Bad_Words	text	
Bad_Words_count	int	
Total_Words_count	int	
Percentage_Bad_Words_count	float	calculated as (Bad_Words_count\Total_Words_count)
Lit_Words	text	
Lit_Words_count	int	
Percentage_Lit_Words_count	float	calculated as (Lit_Words_count\Total_Words_count)

Implementation

The first step of the implementation was to prepare the preprocessed data (defined in the **Data Preprocessing** section) for the classification models and split this data into training and testing subsets.

Classification Model Steps:

1) Get Classification Data: Extracted only the following fields from the email_csv_with_bad_words.csv file: (generated from the **Data Preprocessing** steps**)** file: ["Date","POI",'file','Percentage_Bad_Words_count','Bad_Words_count', 'Total_Words_count','Lit_Words_count','Percentage_Lit_Words_count']

2) Prepare Classification Data:

- Created new output/target variable called Classification1 with two types of classification – 'YES_Negative' where the Percentage_Bad_Words_count > 0 and 'No_Negative' where Percentage_Bad_Words_count >= 0, using the LabelEncoder() fitted and transformed the Classification1 label

- Defined the Features for the Classification Model: 'Bad_Words_count', 'Total_Words_count','Lit_Words_count','Percentage_Lit_Words_count'

  Total_Words_count - Total words found in email message
  Bad_Words_count - # of Financial Negative words found in email message
  Lit_Words_count - # of Financial Litigation words found in email message
  Percentage_Lit_Words_count - # of Financial Litigation words (Lit_Words_count) found in email message/Total Word Count Found in email message

- Split the data into Training & Testing subsets: Training set is all the email messages where ('POI') == True, and the Testing set is where all the email messages = ('POI') == False.

  Size of Training Set: 5055
  Size of Test Set: 493,410

  - Defined X_train, y_train, X_test, y_test

    X_train = Training set Features
    X_test = Testing set Features
    y_train = Training set's Classification1
    y_text = Testing set's Classifcation1

The second step of the implementation was to prepare the preprocessed data (defined in the **Data Preprocessing** section) for the regression models and split this data into training and testing subsets.

Regression Model Steps

1) Get Regression Data: Extracted only the following fields from the email_csv_with_bad_words.csv file: (generated from the **Data Preprocessing**

steps**):** ["Date","POI",'file','Percentage_Bad_Words_count','Bad_Words_count', 'Total_Words_count','Lit_Words_count','Percentage_Lit_Words_count']

2) Prepare Regression Data:
   - Features for the Regression Model: 'Bad_Words_count', 'Total_Words_count','Lit_Words_count','Percentage_Lit_Words_count'

     Total_Words_count - Total words found in email message
     Bad_Words_count - # of Financial Negative words found in email message
     Lit_Words_count - # of Financial Litigation words found in email message
     Percentage_Lit_Words_count - # of Financial Litigation words (Lit_Words_count) found in email message/Total Word Count Found in email message

   - Split the data into Training & Testing subsets: Training set is all the email messages where ('POI') == True, and the Testing set is where all the email messages = ('POI') == False.

     Size of Training Set: 5055
     Size of Testing Set: 493,410

   - Define X_train, y_train, X_test, y_test

     X_train = Training set Features
     X_test = Testing set Features
     y_train = Training set's ['Percentage_Bad_Words_count']
     y_text = Testing set's ['Percentage_Bad_Words_count']

Model Execution

Classification Model Parameters:

1. SVC was executed with the default Parameters:
   (*C=1.0*, *kernel='rbf'*, *degree=3*, *gamma='auto'*, *coef0=0.0*, *shrinking=True*, *probability=False*, *tol=0.001*, *cache_size=200*, *class_weight=None*, *verbose=False*, *max_iter=-1*, *decision_function_shape=None*, *random_state=None*)

   To measure the success of this model, I used the Accuracy Score.[xix]

2. KNeighborsClassifier was executed with the default Parameters:
(*n_neighbors=5*, *weights='uniform'*, *algorithm='auto'*, *leaf_size=30*, *p=2*, *metric='minkowski'*, *metric_params=None*, *n_jobs=1*, ***kwargs*)

   To measure the success of this model, I used the Accuracy Score. [xi]

3. QDA[xx] was executed with the default Parameters:
(*priors=None*, *reg_param=0.0*, *store_covariances=False*, *tol=0.0001*)

   To measure the success of this model, I used the Accuracy Score.[xxi]

Regression Model Parameters:

1. For the Decision Tree Regressor, I trained the model using the grid search technique to identify the optimal parameters for this model. I set the max depth parameters from (1 -10) with presort set to True.

   [{'max_depth':(1,2,3,4,5,6,7,8,9,10),'presort':['True']}]

   R2 Score was used to measure the models' performance.

2. **Bagging Regressor** was executed with the default parameters:
(*base_estimator=None*, *n_estimators=10*, *max_samples=1.0*, *max_features=1.0*, *bootstrap=True*, *bootstrap_features=False*, *oob_score=False*, *warm_start=False*, *n_jobs=1*, *random_state=None*, *verbose=0*)

   R2 Score was applied to measure the success of this model.

3. Gradient Boosting Regressor was executed with the default parameters:
(*loss='ls'*, *learning_rate=0.1*, *n_estimators=100*, *subsample=1.0*, *criterion='friedman_mse'*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_depth=3*, *min_impurity_split=1e-07*, *init=None*, *random_state=None*, *max_features=None*, *alpha=0.9*, *verbose=0*, *max_leaf_nodes=None*, *warm_start=False*, *presort='auto'*)

   R2 Score was applied to measure the success of this model. [xvi]

Refinement

Classification Model: SVC: My initial solution used SVC's default parameters which I achieved an accuracy score of 0.97020327922012117. After tuning these parameters, I received an accuracy score of 1.0 using linear kernel type that was used in the algorithm and setting C (Penalty parameter C of the error term) = 100.0.

SVC was executed with these parameters:
SVC(kernel="linear",C=100.0)

Final Tuned SVC Parameters Selected: SVC(C=100.0, cache_size=200, class_weight=None, coef0=0.0, decision_function_shape=None, degree=3, gamma='auto', kernel='linear', max_iter=-1, probability=False, random_state=None, shrinking=True, tol=0.001, verbose=False))

Regression Model – Decision Tree Regressor: My initial solution used the Decision Tree Regressor's default parameters which I achieved a R2 Score of 0.81266953479600978. After tuning these parameters, I was able to achieve a higher R2 Score of 0.81334950843346998 using GridSearchCV which fits to the data to find the best parameters.

GridSearchCV Parameters:
[{'max_depth':(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20),'presort':['True']}]

I also transformed the R2 score into a scoring function that was passed into the GridSearchCV.

Final Tuned Decision Tree Regressor Parameters Selected:

('Decision Tree Regressor Parameters ', GridSearchCV(cv=None, error_score='raise', estimator=DecisionTreeRegressor(criterion='mse',max_depth=None,max_features=None, max_leaf_nodes=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, presort=False, random_state=None,splitter='best'), fit_params={}, iid=True, n_jobs=1, param_grid=[{'presort': ['True'], 'max_depth': (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20)}],pre_dispatch='2*n_jobs', refit=True, scoring=make_scorer(r2_score), verbose=0))

IV. Results

Model Evaluation, Validation & Justification

For the classification model's evaluation and validation, I was looking to achieve a .5 or greater accuracy score that was defined in the ***Benchmark*** Section. All three of the classification models I selected achieved .5 or greater however SVM/SVC produced the highest score of 1.0. I was able to achieve and exceed my benchmark with fine tuning the model parameters for the SVC model and using the default parameters for KNeighborsClassifier and QDA.

Classification Results:

Classification Model	Accuracy Score	Benchmark
KNeighborsClassifier	0.89850225978395248	>= .5
SVC	1.0	>= .5
QDA	0.54068016456901968	>= .5

These results confirms my assumption that roughly 50% of the POI email messages contained financial negative words used in the email message and looking to achieve this same percentage or greater in the non-POI email messages.

For the regression model's evaluation, validation and justification, I was looking to achieve a .5 or greater r2 score that was defined in the ***Benchmark*** Section. All three of the regression models I selected achieved .5 or greater however Gradient Boosting Regressor & Bagging Regressor produced the highest R2 score of .814. All three of these regression models produced almost the identical R2 score which helps validate the robustness of these models. I did use GridSearch[x] on the Decision Tree Regressor to help fit the data to find best parameters with a max depth of 10 which was discussed in the **Refinement** Section.

Regression Results:

Regression Model	R2 Score	Benchmark
Decision Tree Regressor	0.81334950843346998	>= .5
Bagging Regressor	0.8142039596713827	>= .5
Gradient Boosting Regressor	0.81408503838378865	>= .5

V. Conclusion

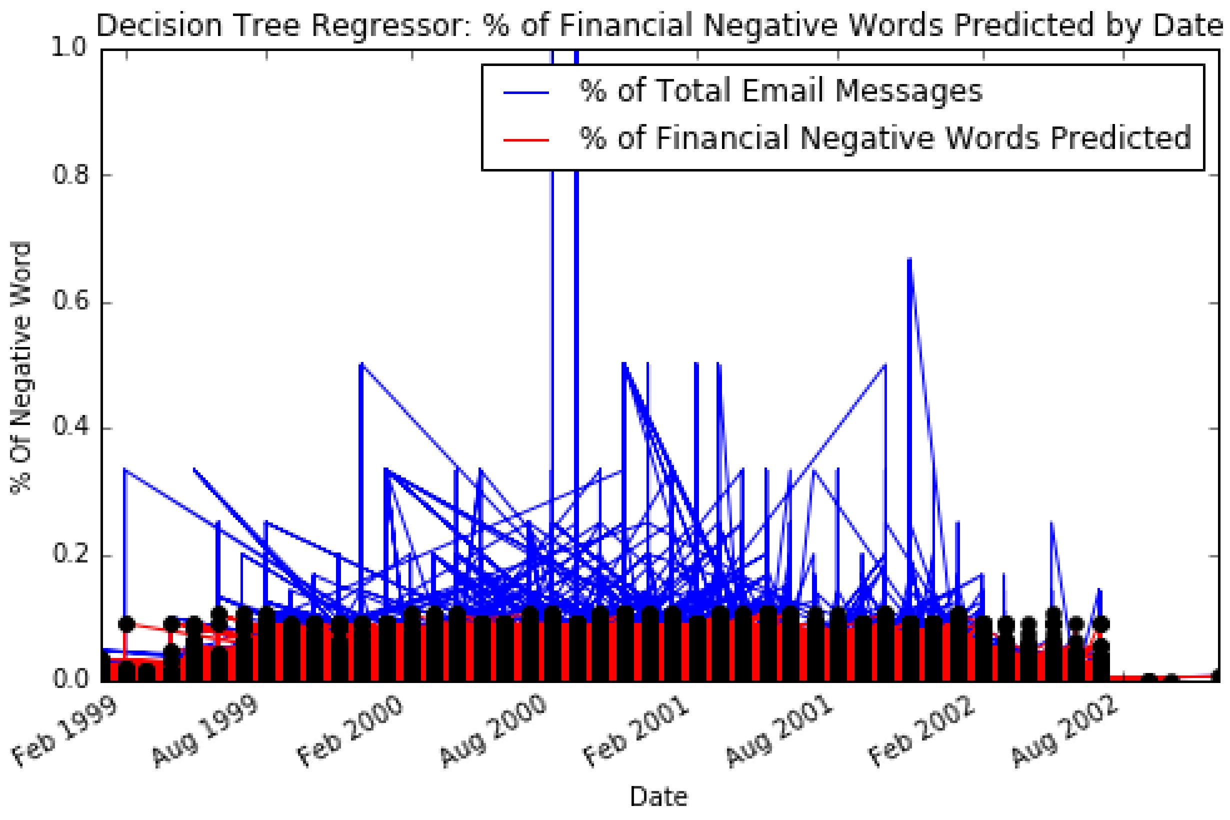
Free-Form Visualization

In this section, I will focus on the prediction results of the decision tree regressor model and the decision tree regressor model's performance.

Prediction Results:

Figure 7 visualizes the predicted % of financial negative words by date. These predictions produced from the testing set were created from the Decision Tree Regressor Model. This graph plots the % of total messages, with the % of predicted financial negative words by date.

Figure 7

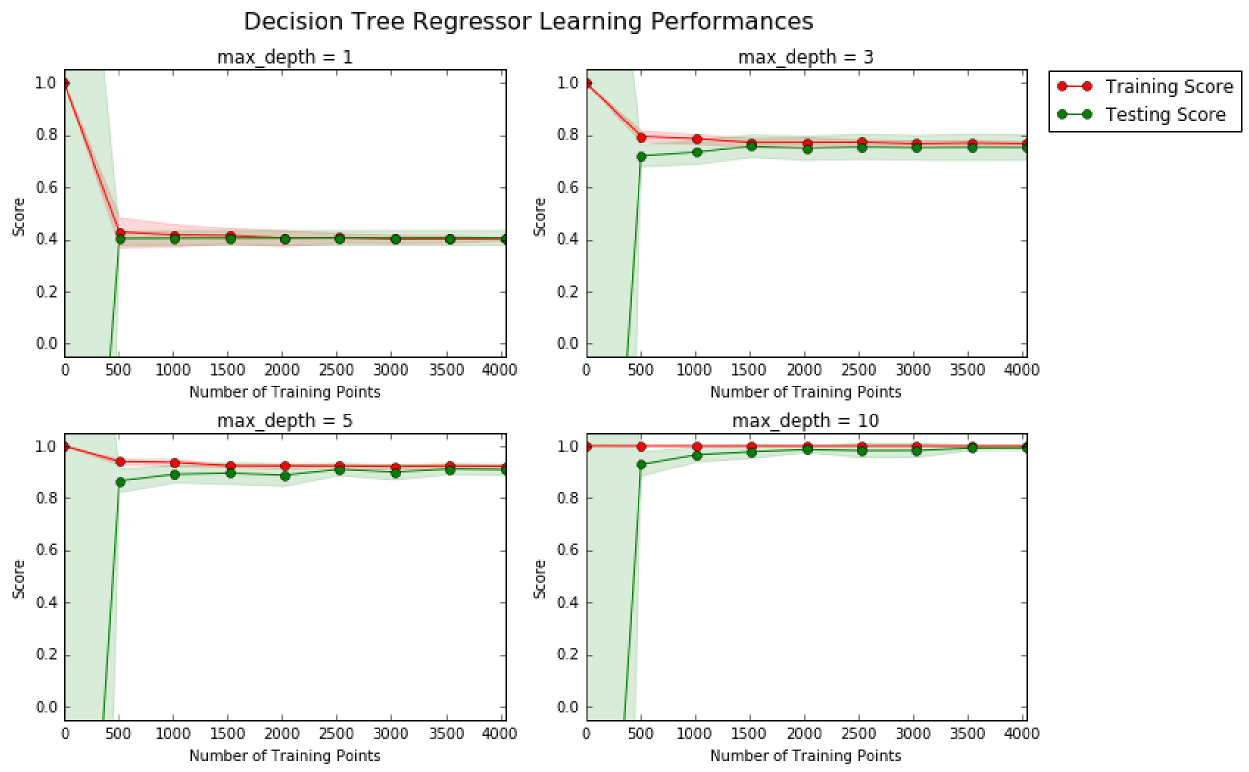


As you can see between 1999 and 2002, roughly 50% of the non-POI email messages (testing set) have been predicted to contain negative finance words based on the POI email messages (training set) which confirms my original assumption defined in the ***Problem Statement*** section.

Model Performance and Evaluation

For model performance, I am choosing the Decision Tree Regressor model to analyze the performance of the model. Figure 8, displays four graphs for the decision tree regressor model with different maximum depths (1, 3, 5, & 10). These graphs visualize the performance of several models with varying sizes of training data. Each graph visualizes the learning curves of the model for both training and testing as the size of the training set is increased. The model is scored on both the training and testing sets using the R2 score.[xxii]

Figure 8

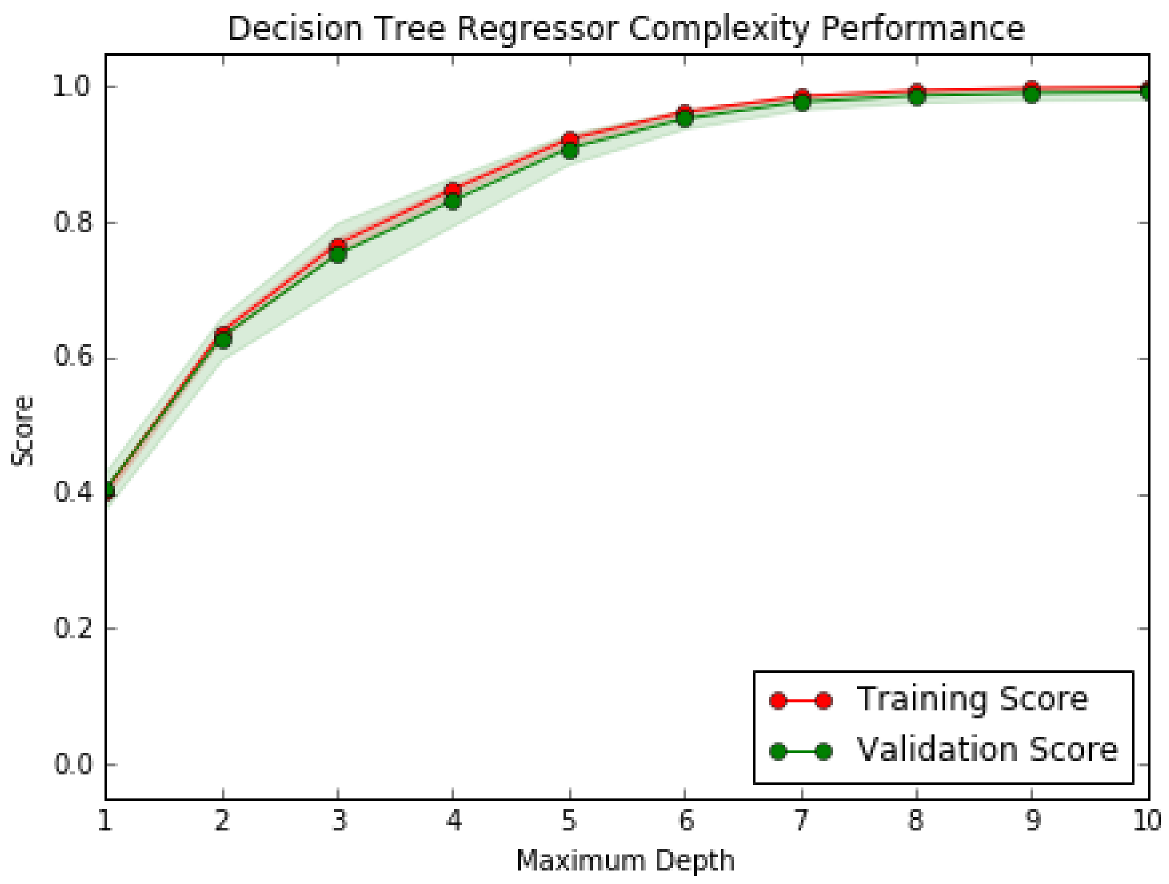


Max_Depth 1,3,5 & 10, we can see that under 500 training points, there is large gap between the training and testing score indicating a high variance. As we start to increase the training points (over 500 training points) the variance between the training score and testing score begins to lower and the number of errors becomes. Also, around 1000 training points, both training and testing curves plateau and the gap between these rates become consistent - validating the model has learned as much about the data ie low errors and the training and testing curves converge.

Decision Tree Regressor Model Complexity

Figure 9, displays a graph for a decision tree regressor model that has been trained and validated on the training data using different maximum depths. The graph produces two complexity curves — one for training and one for validation. The shaded regions of both the complexity curves represent the uncertainty in those curves, and the model is scored on both the training and validation sets using the R2 score.

Figure 9



When a model is trained with max depth of 1, it is suffering from a high bias (the model is underfitting, indicated by the low R2 score and training score. From a visual perspective, this complexity curve shows a low R2 score on both the training and validation scores at max depth of 1(high bias). Max Depth of 9, shows the highest R2 with minimum possible complexity.

Reflection

To summarize this entire project from end-to-end, started with defining at an overview level what this project would be based on, which I selected identifying sentiment analysis related to financial fraud within an email corpus (Enron). Next, was to take this high level overview and break into a problem statement that I was looking to solve was what percentage of the non-POI email messages contained indicators of financial fraudulent activities based on the fraudulent characteristics of the POI email messages and define the strategy necessary to solve this problem, for my project it was listing all the steps necessary to perform classification and regression models on the Enron Email Corpus as well as incorporating the use of a financial sentiment dictionary & list POI email addresses. The next phase was to determine how I was going to measure the performance of the classification and regression models to prove if these models were successful at solving the problem which I selected to use the accuracy score for classification models and r2 score for the regression models.

After I defined my objective, problem statement and the metrics I was going to use, I started the Analysis phase of this project. Starting with the data exploration, I analyzed the raw data from the Enron email corpus, financial sentiment dictionary and the list of POI email addresses to provide insight into these datasets and created visualizations to better understand what this data looked like. After this step, I choose the algorithms that I was going to use to solve the problem - three classification models (SVC, QDA, & KNeighborsClassifier) and three regression models (Decision Tree Regressor, Bagging Regression & Gradient Boosting Regression) and defined the techniques I was going to use on these models. The last component of the Analysis phase was to define and provide the reasoning on what my benchmark would be for both the classification models and regression models, that a percentage of the email messages contained in the non-POI email messages contained indicators of financial fraudulent activities based on the fraudulent characteristics of the POI email messages, I will be looking for an accuracy score greater than .5 in the classification models and an R2 score greater than .5 for the regression models.

After the Analysis phase, next came the Methodology phase starting with Data Preprocessing. This was my most difficult aspect of the project, preprocessing all the input datasets required for the models. Initially, I wrote my code to perform all the steps in cleaning the entire Enron corpus, comparing and matching sentiment financial keywords and matching on the POI email address. The first run took over a week to execute as it was processing all this data in one big process. I broke the code into smaller pieces (3 different steps) however it still took about 3 weeks from end-to-end until I had a solution that could pre-process the data in a timely solution. After this I was excited to start the next phase – Implementation where I was able to prepare my preprocessed data for the models and execute the models with the parameters I defined earlier to see if I achieved my benchmarks. All my models performed very well however the most interesting aspect of this project was being able to improve my Decision Tree Regressor model performance score during the refinements stage using GridSearch.

Under the Results phase, I evaluated each of my final models to make sure it was aligning with my solution expectations, which all three of my classifications and all three of my regression models identified that roughly 50% of the non-POI email messages (testing set) have been predicted to contain negative finance words based on the POI email messages (training set) which confirmed my original assumption defined in the ***Problem Statement*** section and fits with my expectations.

Lastly, under Conclusion – I provided the prediction results from my decision tree regressor model, that shows roughly 50% of the non-POI email messages (testing set) have been predicted to contain negative finance words based on the POI email messages (training set). I also provided visualizations on the model performance and evaluation of my Decision Tree Regressor model which validates the quality of the model.

Improvement

As far as improvements, I would add the use of scikit-learn's TfidfVectorizer[xxiii] which transforms the text into feature vectors. Tfid translates into 'term frequency–inverse document frequency' which captures how important a word is with a numerical measurement. TF is a similar concept to the bag of words which captures words and counts the frequency of the words in an email message and IDF captures the weighting by how often the word occurs in the email corpus, and how rare a word occurs in the email messages.[xxiv] For future improvements, I would also include the analysis of attachments and/or extremely large message (remove the character restriction of 3,000 characters per message) to include all message content and expanding the use of the Loughran and McDonald Financial Sentiment Dictionary by applying the remaining files to this analysis.

Lastly, I would perform testing using recurrent neural network (RNN) to identify the sentence formation versus just using the dictionary word.

---

[i] http://usatoday30.usatoday.com/money/industries/energy/2005-12-28-enron-participants_x.htm

[ii] Featurizing Text: Converting Text into Predictors for Regression Analysis by Dean P. Foster, Mark Liberman Robert A. Stine at Department of Statistics The Wharton School (http://www-stat.wharton.upenn.edu/~stine/research/regressor.pdf) and "Lexicon-Based Methods for Sentiment Analysis" work item published in 2011 by Maite Taboada(Simon Fraser University), Julian Brooke(University of Toronto),Milan Tofiloski(Simon Fraser University),Kimberly Voll (University of British Columbia) Manfred Stede(University of Potsdam).

[iii] https://www.kaggle.com/wcukierski/enron-email-dataset

[iv] https://www.cs.cmu.edu/~./enron/

[v] Loughran, T. & McDonald, B. (2011). When is a liability not a liability? Textual Analysis, Dictionaries and 10-Ks. The Journal of Finance, 66(1), 35-66.

[vi] https://github.com/jperla/sentiment-data/blob/master/finance/LoughranMcDonald_Negative.csv

[vii] https://github.com/jperla/sentiment-data/blob/master/finance/LoughranMcDonald_Litigious.csv

[viii] http://scikit-learn.org/stable/index.html

[ix] http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[x] http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

[xi] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html

[xii] http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[xiii] http://scikit-learn.org/stable/modules/lda_qda.html

[xiv] http://scikit-learn.org/0.15/modules/lda_qda.html

[xv] http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html

[xvi] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html

[xvii] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingRegressor.html

[xviii] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingRegressor.html

[xix] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html

[xx] http://scikit-learn.org/stable/modules/lda_qda.html

[xxi] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html

[xxii] Udacity's Machine Learning Nanodegree – The Boston Housing Project included components from the visuals.py code

[xxiii] http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

[xxiv] https://www.udacity.com/course/viewer#!/c-ud120/l-2892378590/e-3012738650/m-3015918703