Capstone Project 2 - Milestone

Loading Enron Email File

In [1]:
```
import spacy
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.utils import shuffle

from collections import Counter
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

from sklearn import metrics
from sklearn.svm import SVC
from sklearn.svm import LinearSVC

from sklearn.pipeline import Pipeline
from sklearn.decomposition import PCA
from sklearn.decomposition import TruncatedSVD

from sklearn.naive_bayes import MultinomialNB

from sklearn.linear_model import SGDClassifier
from sklearn.linear_model import LogisticRegression

from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [2]:
```
def getFiles():
    ''' Get a list of file from specifiec directory '''
    FileNames = !find * | sort
    #return list of files
    return FileNames
```

In [3]:
```
def getEmailTextList(fileList, label, binCode):
    ''' Iterate list of email .txt files open each file and store text, 
     in 2D list stored in DataFrame as return value'''
    # Initialization
    lineList = []
    emailTextList = pd.DataFrame()

    # Extract file contents
    for file in fileList:
        # Initialize for each file
        text = ''

        # Open file to read - using 'rb' instead of 'r' to read bytes (a
        text = open(file, 'rb').read().decode('latin')
        # split text into lines to get count of lines for content table
        lines = text.splitlines()

        # Create a list of email text, number of lines read, and word co
        lineList.append([text, label, binCode, len(lines), len(text)])

    # Convert lineList to DataFrame
    emailTextList = pd.DataFrame(lineList, \
                                 columns = ['Text', 'Label', 'Ham1/Spam0

    # Return list of email text with corresponding lable and binary code
    return emailTextList
```

In [9]:
```
def readTextList(label, binaryCode):
    ''' Read file in current directory and reaturn a list of emails'''
    fileList = getFiles()
    emailContentList = getEmailTextList(fileList, label, binaryCode)
    return emailContentList
```

Data Wrangling

Read 'Ham' Emails into Data Frame

In [11]:
```
hamTextList = readTextList('ham', 1)
print(len(hamTextList))
```

```
5709
```

Read 'Spam' Emails into Data Frame

```
In [14]: spamTextList = readTextList('spam', 0)
         print(len(spamTextList))
```

```
5241
```

Randomly Distributed Ham & Spam Email Dataframe

```
In [16]: # Collection of all Ham and Spam Emails - Data Frame includes text, labe
         EmailTextList = hamTextList.append(spamTextList)

         # Randomize/shuffle rows distributing ham and spam emails
         EmailTextList = shuffle(EmailTextList)
         EmailTextList.head()
```

Out[16]:

	Text	Label	Ham1/Spam0	Line Count	Text Length
2628	Subject: cruise 3 nts mexico only $ 197 ! - - ...	spam	0	23	1427
1005	Subject: ba & sao paulo\r\n- - - - - - - - - -...	ham	1	10	452
3675	Subject: re : status\r\nclayton ,\r\nwe can di...	ham	1	87	5656
2219	Subject: enlarge your penls\r\nenlarge your pe...	spam	0	3	60
1377	Subject: urgent business\r\n> > from the desk ...	spam	0	26	2538

```
In [18]: EmailTextList.to_csv('EmailList.csv')
```

Train and Test Data Distribution

In [17]:
```
# Split train and test vectors
X_train, X_test, y_train, y_test = train_test_split(EmailTextList['Text'
                                                    EmailTextList['Ham1/
                                                    test_size = 0.25, ra
```

ANALYSIS

- **Select relevant ML model for classification of labeled data**
  https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html
  - Logistic Regression
  - Naive Bayes
  - SVC
  - Linear SVC
  - SGD

- **Create pipeline for each model**
- **List TFIDF and ML model in pipeline**
- **Evaluate model performance F1- score**
- **Select Hyperparameters and values of interest**
- **Review and evaluate model performance F1- score to finalize parameter settings of interest**
- **Create abd display table containing model applied, performance score, and corresponding hyperparameters**
- **Plot model performance for visualization**

In [ ]:
```

```