Team 8 Final Project - Amazon Kindle Reviews

In [40]:
```python
# imports
import warnings
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import re
import json
import nltk
import spacy
import string
import unicodedata
from bs4 import BeautifulSoup
from nltk.stem import WordNetLemmatizer
import emoji
from IPython import display
display.set_matplotlib_formats('svg')
warnings.filterwarnings('ignore')\

from sklearn import metrics
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from string import punctuation
from sklearn.feature_extraction.text import CountVectorizer
from nltk.corpus import stopwords
sw = stopwords.words("english")
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer


import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM,Dense, Dropout, SpatialDropout1D
from tensorflow.keras.layers import Embedding

# These libraries may be useful to you

from nltk.corpus import brown

import numpy as np
import pandas as pd
from tqdm.auto import tqdm

#import gensim
#import gensim.corpora as corpora
#from gensim.utils import simple_preprocess
#from gensim.models import CoherenceModel,LdaMulticore, Phrases
#from gensim.models.phrases import Phraser
#from gensim.corpora import Dictionary

import pyLDAvis
import pyLDAvis.sklearn
import pyLDAvis.gensim_models

import spacy
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.decomposition import NMF, TruncatedSVD, LatentDirichletAllocation

from spacy.lang.en.stop_words import STOP_WORDS as stopwords

from collections import Counter, defaultdict

import nltk
nltk.download('wordnet')

# nlp = spacy.load('en_core_web_sm')
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\jackt\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Out[40]: True

Loading the Amazon Kindle Reviews Dataset

In [41]:
```python
# load the data
df = pd.read_csv('/Users/jackt/OneDrive/Desktop/USD/Class 11 Text Mining/Final Project/all_kindle_review .cs
df.head()
```

Out[41]:

	Unnamed: 0.1	Unnamed: 0	asin	helpful	rating	reviewText	reviewTime	reviewerID	reviewerName	summary	unixRevie
0	0	11539	B0033UV8HI	[8, 10]	3	Jace Rankin may be short, but he's nothing to ...	09 2, 2010	A3HHXRELK8BHQG	Ridley	Entertaining But Average	12833
1	1	5957	B002HJV4DE	[1, 1]	5	Great short read. I didn't want to put it dow...	10 8, 2013	A2RGNZ0TRF578I	Holly Butler	Terrific menage scenes!	13811
2	2	9146	B002ZG96I4	[0, 0]	3	I'll start by saying this is the first of four...	04 11, 2014	A3S0H2HV6U1I7F	Merissa	Snapdragon Alley	13971
3	3	7038	B002QHWOEU	[1, 3]	3	Aggie is Angela Lansbury who carries pocketboo...	07 5, 2014	AC4OQW3GZ919J	Cleargrace	very light murder cozy	14045
4	4	1776	B001A06VJ8	[0, 1]	4	I did not expect this type of book to be in li...	12 31, 2012	A3C9V987IQHOQD	Rjostler	Book	13569

Cleaning the Data

In [42]:
```python
# Functions for cleaning and analyzing the dataset
# Some punctuation variations
punctuation = set(punctuation) # speeds up comparison
tw_punct = punctuation - {"#"}

# Stopwords
sw = nltk.corpus.stopwords.words('english')


# Two useful regex
whitespace_pattern = re.compile(r"\s+")
hashtag_pattern = re.compile(r"^#[0-9a-zA-Z]+")

# It's handy to have a full set of emojis
all_language_emojis = set()

for country in emoji.UNICODE_EMOJI :
    for em in emoji.UNICODE_EMOJI[country] :
        all_language_emojis.add(em)

# and now our functions
def descriptive_stats(tokens, num_tokens = 5, verbose=True) :
    """
        Given a list of tokens, print number of tokens, number of unique tokens,
        number of characters, lexical diversity (https://en.wikipedia.org/wiki/Lexical_diversity),
        and num_tokens most common tokens. Return a list with the number of tokens, number
        of unique tokens, lexical diversity, and number of characters.

    """

    # Fill in the correct values here.
    num_tokens = len(tokens)
    num_unique_tokens = len(set(tokens))
    lexical_diversity = num_unique_tokens/num_tokens
    num_characters = len("".join(tokens))

    if verbose :
        print(f"There are {num_tokens} tokens in the data.")
        print(f"There are {num_unique_tokens} unique tokens in the data.")
        print(f"There are {num_characters} characters in the data.")
        print(f"The lexical diversity is {lexical_diversity:.3f} in the data.")

        # print the five most common tokens

    return([num_tokens, num_unique_tokens,
            num_characters,
            lexical_diversity
            ])


def is_emoji(s):
    return(s in all_language_emojis)

def contains_emoji(s):
    s = str(s)
    emojis = [ch for ch in s if is_emoji(ch)]
    return(len(emojis) > 0)


def remove_stop(tokens):
    stopwords = set(nltk.corpus.stopwords.words('english'))
    return [word.lower() for word in tokens if word not in stopwords]

def remove_punctuation(text, punct_set=tw_punct) :
    return("".join([ch for ch in text if ch not in punct_set]))

def tokenize(text) :
    """ Splitting on whitespace rather than the book's tokenize function. That
        function will drop tokens like '#hashtag' or '2A', which we need for Twitter. """
    collapse_whitespace = re.compile(r'\s+')
    return([item.lower() for item in collapse_whitespace.split(text)])

def prepare(text, pipeline) :
    tokens = str(text)

    for transform in pipeline :
        tokens = transform(tokens)

    return(tokens)
```

In [43]:
```python
my_pipeline = [str.lower, remove_punctuation, tokenize, remove_stop]

df["tokens"] = df["reviewText"].apply(prepare,pipeline=my_pipeline)
df["num_tokens"] = df["tokens"].map(len)
```

In [44]:
```python
df.head()
```

Out[44]:

	Unnamed: 0.1	Unnamed: 0	asin	helpful	rating	reviewText	reviewTime	reviewerID	reviewerName	summary	unixRevie
0	0	11539	B0033UV8HI	[8, 10]	3	Jace Rankin may be short, but he's nothing to ...	09 2, 2010	A3HHXRELK8BHQG	Ridley	Entertaining But Average	12833
1	1	5957	B002HJV4DE	[1, 1]	5	Great short read. I didn't want to put it dow...	10 8, 2013	A2RGNZ0TRF578I	Holly Butler	Terrific menage scenes!	13811
2	2	9146	B002ZG96I4	[0, 0]	3	I'll start by saying this is the first of four...	04 11, 2014	A3S0H2HV6U1I7F	Merissa	Snapdragon Alley	13971
3	3	7038	B002QHWOEU	[1, 3]	3	Aggie is Angela Lansbury who carries pocketboo...	07 5, 2014	AC4OQW3GZ919J	Cleargrace	very light murder cozy	14045
4	4	1776	B001A06VJ8	[0, 1]	4	I did not expect this type of book to be in li...	12 31, 2012	A3C9V987IQHOQD	Rjostler	Book	13569

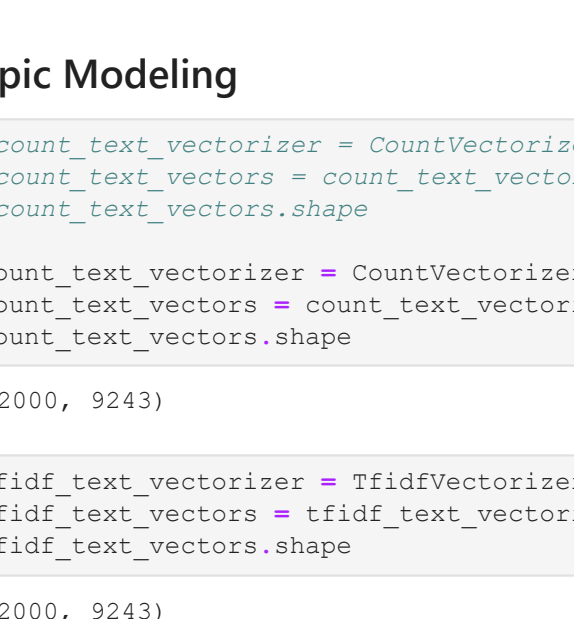
Descriptive Statistics

In [45]:
```python
reviewtext = ''.join(' '.join(elems) for elems in df['tokens'])

full_review_text_amazon = reviewtext.split()

descriptive_stats(reviewtext, verbose=True)
```

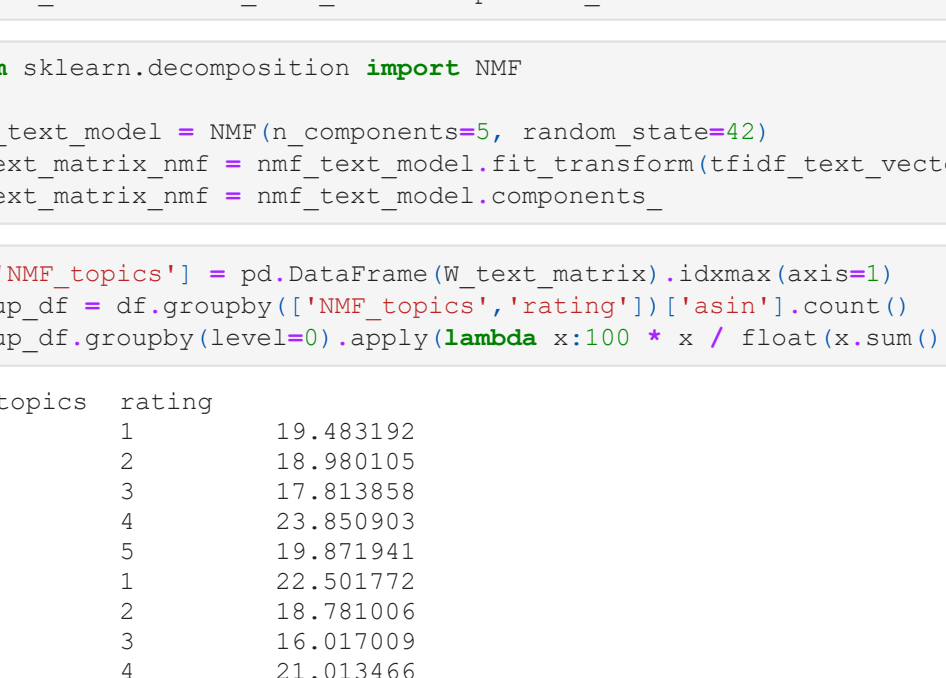
```
There are 4494744 tokens in the data.
There are 38 unique tokens in the data.
There are 4494744 characters in the data.
The lexical diversity is 0.000 in the data.
```

Out[45]: [4494744, 38, 4494744, 8.45431908914056e-06]

In [46]:
```python
y = np.array([6000,6000])
mylabels = ["Postive Sentiment", "Negative Sentiment"]

plt.pie(y, labels = mylabels)
plt.legend()
plt.show()
```



In [47]:
```python
df.rating.hist()
plt.show()
```



Topic Modeling

In [48]:
```python
#count_text_vectorizer = CountVectorizer(stop_words= stopwords.words("english"), min_df=5, max_df=0.7)
#count_text_vectors = count_text_vectorizer.fit_transform(df["reviewText"])
#count_text_vectors.shape

count_text_vectorizer = CountVectorizer(stop_words=set(sw), min_df=5, max_df=0.7)
count_text_vectors = count_text_vectorizer.fit_transform(df["reviewText"])
count_text_vectors.shape
```

Out[48]: (12000, 9243)

In [49]:
```python
tfidf_text_vectorizer = TfidfVectorizer(stop_words=set(sw), min_df=5, max_df=0.7)
tfidf_text_vectors = tfidf_text_vectorizer.fit_transform(df['reviewText'])
tfidf_text_vectors.shape
```

Out[49]: (12000, 9243)

In [50]:
```python
nmf_text_model = NMF(n_components=5, random_state=314)
W_text_matrix = nmf_text_model.fit_transform(tfidf_text_vectors)
H_text_matrix = nmf_text_model.components_
```

In [51]:
```python
from sklearn.decomposition import NMF

nmf_text_model = NMF(n_components=5, random_state=42)
W_text_matrix_nmf = nmf_text_model.fit_transform(tfidf_text_vectors)
H_text_matrix_nmf = nmf_text_model.components_
```

In [52]:
```python
df['NMF_topics'] = pd.DataFrame(W_text_matrix).idxmax(axis=1)
group_df = df.groupby(['NMF_topics','rating'])['asin'].count()
group_df.groupby(level=0).apply(lambda x:100 * x / float(x.sum()))
```

Out[52]:
```
NMF_topics  rating
0           1         19.483192
            2         18.980105
            3         17.813858
            4         23.850903
            5         19.871941
1           1         22.501772
            2         18.781006
            3         16.017009
            4         21.013466
            5         21.686747
2           1         13.080357
            2         18.080357
            3         20.937500
            4         28.348214
            5         19.553571
3           1          6.996820
            2          7.496592
            3         10.722399
            4         29.304861
            5         45.479328
4           1         18.131868
            2         19.230769
            3         17.582418
            4         23.076923
            5         21.978022
Name: asin, dtype: float64
```

In [53]:
```python
group_df.groupby(level=0).apply(lambda x:100 * x / float(x.sum())).plot.bar(figsize=(10,6))
```

Out[53]: <AxesSubplot:xlabel='NMF_topics,rating'>

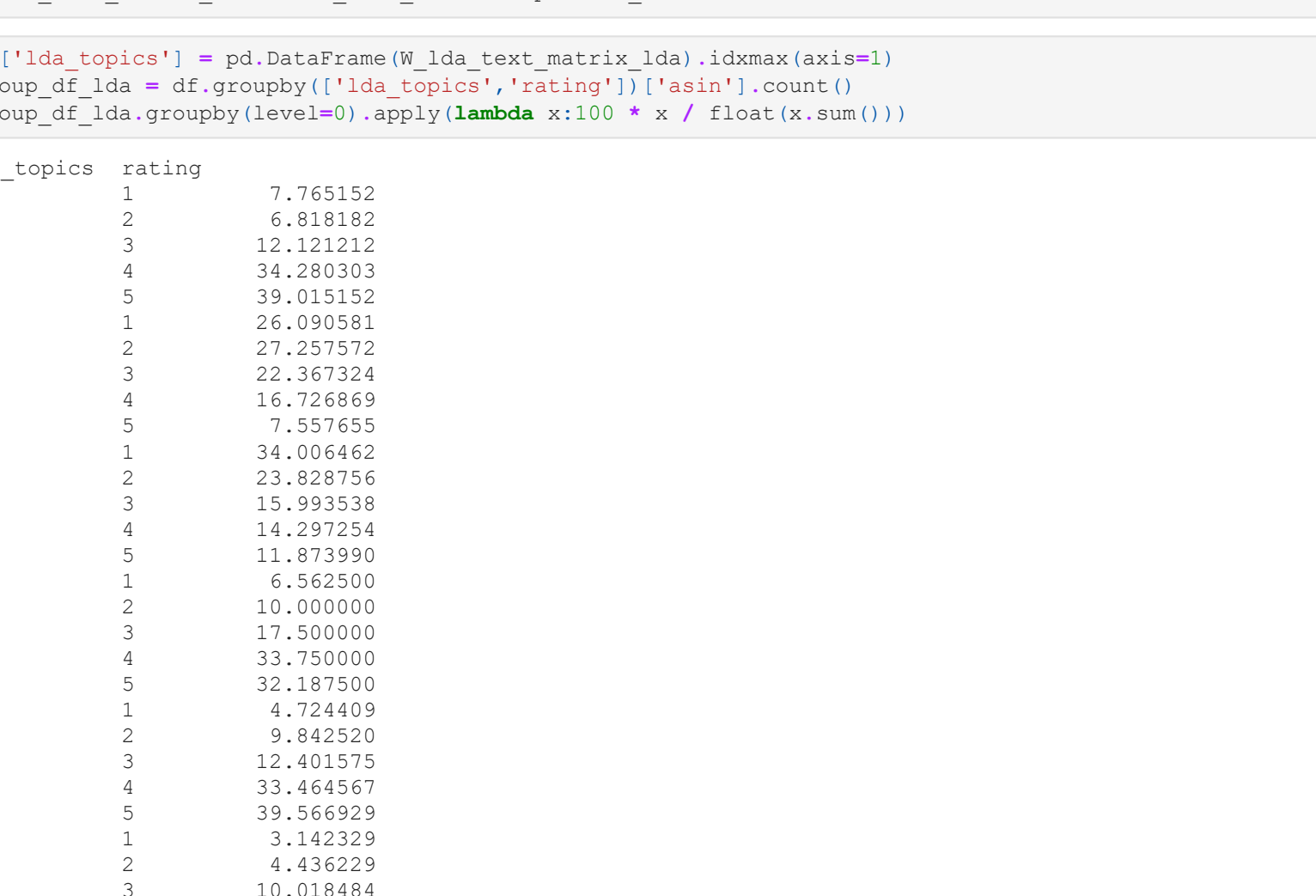
In [54]:
```python
svd_text_model = TruncatedSVD(n_components = 5, random_state=42)
W_svd_text_matrix_svd = svd_text_model.fit_transform(tfidf_text_vectors)
H_svd_text_matrix_svd = svd_text_model.components_
```

In [55]:
```python
df['svd_topics'] = pd.DataFrame(W_svd_text_matrix_svd).idxmax(axis=1)
group_df_svd = df.groupby(['svd_topics','rating'])['asin'].count()
group_df_svd.groupby(level=0).apply(lambda x:100 * x / float(x.sum()))
```

Out[55]:
```
svd_topics  rating
0           1         16.909826
            2         16.857093
            3         16.927404
            4         25.021972
            5         24.283705
1           1         10.000000
            2         11.111111
            3          6.666667
            4         24.444444
            5         47.777778
2           1          1.421801
            2          2.369668
            3          7.582938
            4         26.540284
            5         62.085308
3           1         31.250000
            2         21.875000
            3         12.500000
            4         28.125000
            5          6.250000
4           1         18.685121
            2         20.761246
            3         16.608997
            4         22.837370
            5         21.107266
Name: asin, dtype: float64
```

In [56]:
```python
group_df_svd.groupby(level=0).apply(lambda x:100 * x / float(x.sum())).plot.bar(figsize=(10,6))
```

Out[56]: <AxesSubplot:xlabel='svd_topics,rating'>



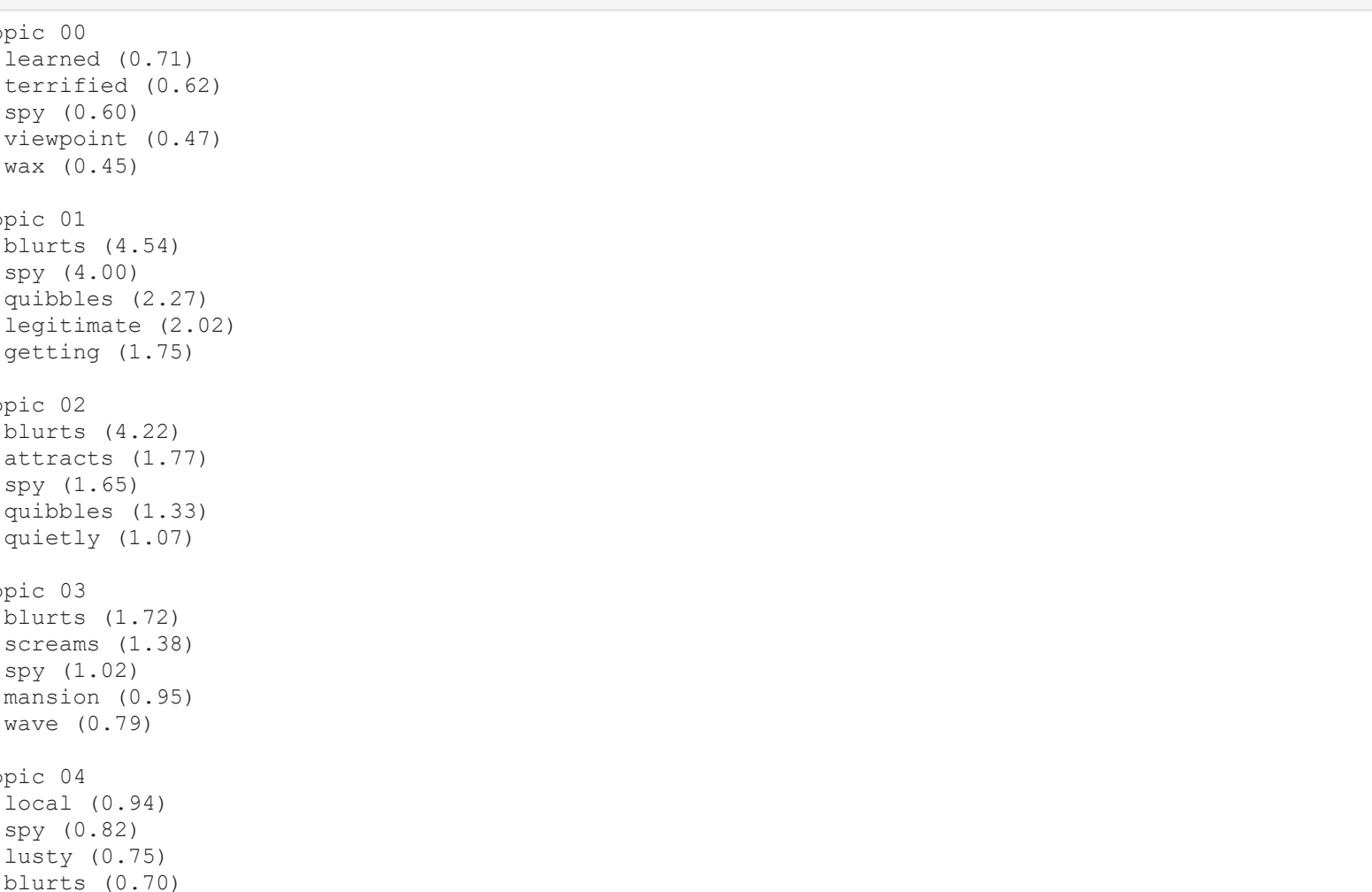
In [57]:
```python
from sklearn.feature_extraction.text import CountVectorizer
count_text_vectorizer = CountVectorizer(stop_words=stopwords, min_df=5,max_df=0.7)

count_text_vectors = count_text_vectorizer.fit_transform(df["reviewText"])

from sklearn.decomposition import LatentDirichletAllocation
lda_text_model = LatentDirichletAllocation(n_components = 10, random_state=42)
W_lda_text_matrix_lda = lda_text_model.fit_transform(count_text_vectors)
H_lda_text_matrix_lda = lda_text_model.components_
```

In [58]:
```python
df['lda_topics'] = pd.DataFrame(W_lda_text_matrix_lda).idxmax(axis=1)
group_df_lda = df.groupby(['lda_topics','rating'])['asin'].count()
group_df_lda.groupby(level=0).apply(lambda x:100 * x / float(x.sum()))
```

Out[58]:
```
lda_topics  rating
0           1          7.765152
            2          6.818182
            3         12.121212
            4         34.280303
            5         39.015152
1           1         26.090581
            2         27.257572
            3         22.367324
            4         16.726869
            5          7.557655
2           1         34.006462
            2         23.828756
            3         15.993538
            4         14.297254
            5         11.873990
3           1          6.562500
            2         10.000000
            3         17.500000
            4         33.750000
            5         32.187500
4           1          4.724409
            2          9.842520
            3         12.401575
            4         33.464567
            5         39.566929
5           1          3.142329
            2          4.436229
            3         10.018484
            4         32.754159
            5         49.648799
6           1         14.365079
            2         13.730159
            3         20.000000
            4         29.285714
            5         22.619048
7           1          2.909091
            2          2.181818
            3         10.181818
            4         42.545455
            5         42.181818
8           1         15.537849
            2         10.358566
            3         13.147410
            4         26.693227
            5         34.262948
9           1         18.967136
            2         23.943662
            3         18.497653
            4         24.037559
            5         14.553991
Name: asin, dtype: float64
```

In [59]:
```python
group_df_lda.groupby(level=0).apply(lambda x:100 * x / float(x.sum())).plot.bar(figsize=(10,6))
```

Out[59]: <AxesSubplot:xlabel='lda_topics,rating'>



In [60]:
```python
def display_topics(model, features, no_top_words=5):
    for topic, words in enumerate(model.components_):
        total = words.sum()
        largest = words.argsort()[::-1] # invert sort order
        print("\nTopic %02d" % topic)
        for i in range(0, no_top_words):
            print("  %s (%2.2f)" % (features[largest[i]], abs(words[largest[i]]*100.0/total)))
```

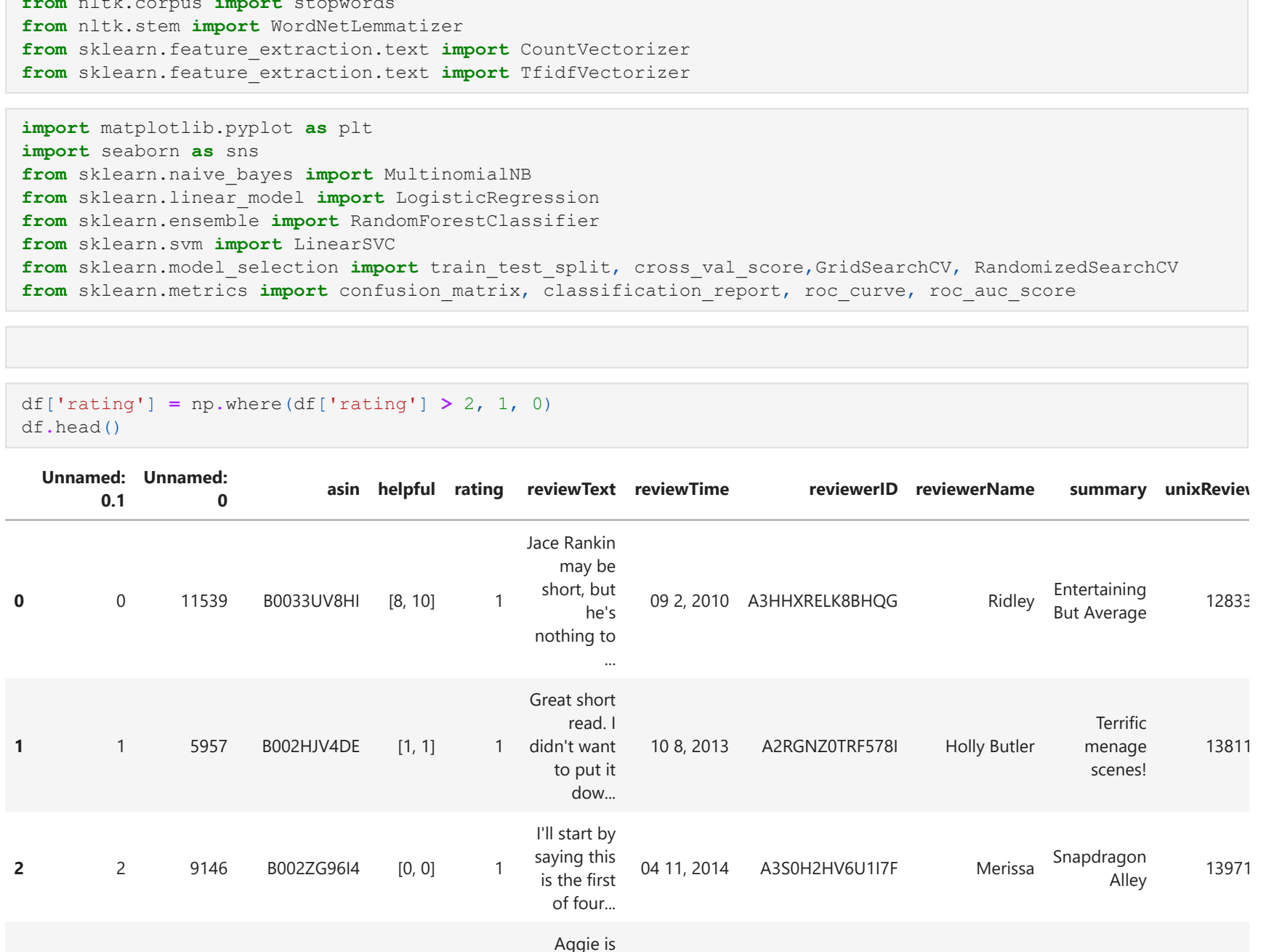
In [61]:
```python
display_topics(lda_text_model, tfidf_text_vectorizer.get_feature_names())
```

```
Topic 00
  learned (0.71)
  terrified (0.62)
  spy (0.60)
  viewpoint (0.47)
  wax (0.45)

Topic 01
  blurts (4.54)
  spy (4.00)
  quibbles (2.27)
  legitimate (2.02)
  getting (1.75)

Topic 02
  blurts (4.22)
  attracts (1.77)
  spy (1.65)
  quibbles (1.33)
  quietly (1.07)

Topic 03
  blurts (1.72)
  screams (1.38)
  spy (1.02)
  mansion (0.95)
  wave (0.79)

Topic 04
  local (0.94)
  spy (0.82)
  lusty (0.75)
  blurts (0.70)
  fake (0.61)

Topic 05
  quibbles (5.12)
  blurts (4.63)
  board (2.29)
  spy (2.01)
  screams (1.98)

Topic 06
  local (1.55)
  spy (1.47)
  secondary (1.40)
  legitimate (0.88)
  holt (0.80)

Topic 07
  screams (1.18)
  blurts (1.00)
  quibbles (0.90)
  ipad (0.73)
  moody (0.69)

Topic 08
  kat (3.41)
  quibbles (1.29)
  8217 (1.07)
  legitimate (0.88)
  always (0.70)

Topic 09
  spy (1.99)
  blurts (1.72)
  legitimate (1.28)
  cereal (1.17)
  central (0.99)
```

In [62]:
```python
lda_display = pyLDAvis.sklearn.prepare(lda_text_model, count_text_vectors, count_text_vectorizer, sort_topic
```

In [63]:
```python
pyLDAvis.display(lda_display)
```

Out[63]:



Classification Model

In [64]:
```python
import string
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [65]:
```python
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import LinearSVC
from sklearn.model_selection import train_test_split, cross_val_score,GridSearchCV, RandomizedSearchCV
from sklearn.metrics import confusion_matrix, classification_report, roc_curve, roc_auc_score
```

In [ ]:

In [66]:
```python
df['rating'] = np.where(df['rating'] > 2, 1, 0)
df.head()
```

Out[66]:

	Unnamed: 0.1	Unnamed: 0	asin	helpful	rating	reviewText	reviewTime	reviewerID	reviewerName	summary	unixRevie
0	0	11539	B0033UV8HI	[8, 10]	1	Jace Rankin may be short, but he's nothing to ...	09 2, 2010	A3HHXRELK8BHQG	Ridley	Entertaining But Average	12833
1	1	5957	B002HJV4DE	[1, 1]	1	Great short read. I didn't want to put it dow...	10 8, 2013	A2RGNZ0TRF578I	Holly Butler	Terrific menage scenes!	13811
2	2	9146	B002ZG96I4	[0, 0]	1	I'll start by saying this is the first of four...	04 11, 2014	A3S0H2HV6U1I7F	Merissa	Snapdragon Alley	13971
3	3	7038	B002QHWOEU	[1, 3]	1	Aggie is Angela Lansbury who carries pocketboo...	07 5, 2014	AC4OQW3GZ919J	Cleargrace	very light murder cozy	14045
4	4	1776	B001A06VJ8	[0, 1]	1	I did not expect this type of book to be in li...	12 31, 2012	A3C9V987IQHOQD	Rjostler	Book	13569

In [67]:
```python
def make_xy(data, vec, n):
    """Takes in a dataframe with text and labels and returns a vocabulary of some sort
        depending on the vectorizer used.

        Arguments:

        data - the input dataframe containing the text and the labels

        vec - the chosen vectorizer to use

        n - the number of samples per class
        """
    temp = pd.DataFrame()
    for rating in range(2):
        temp = pd.concat([temp, data[data.rating == rating].sample(n, random_state = 42)], ignore_index = Tr

    #vectorizing the vocabulary
    X = vec.fit_transform(temp.reviewText)
    y = temp.rating
    return X, y
```

In [68]:
```python
# The Count vectorizer to make our X and y's:

count = CountVectorizer()
X, y = make_xy(df, count, 2000)
```

In [69]:
```python
# Now needing to split the dataset into the train and test sets:

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .25, random_state = 42, stratify = y)

naive_bayes_model1 = MultinomialNB()
naive_bayes_model1.fit(X_train, y_train)
print('Training Accuracy: {:.2f}'.format(naive_bayes_model1.score(X_train, y_train)))
print('Testing Accuracy: {:.2f}'.format(naive_bayes_model1.score(X_test, y_test)))
```

```
Training Accuracy: 0.92
Testing Accuracy: 0.81
```

In [71]:
```python
# Now Looking at the Naive Bayes multinomial model to look at best score, best alpham and best min df:

best_alpha = 0
best_min_df = 0
best_score = 0

# Alpha/mind_df:

alphas = [.1, 1, 5, 10, 50]
min_dfs = [1e-5, 1e-4, 1e-3, 1e-2, 1e-1]

for alpha in alphas:
    for m_df in min_dfs:
        tfidf = TfidfVectorizer(min_df = m_df)
        X, y = make_xy(df, tfidf, 2000)
        naive_bayes_model2 = MultinomialNB(alpha = alpha)
        score = np.mean(cross_val_score(naive_bayes_model2, X, y, scoring = 'accuracy', cv = 3))
        if score > best_score:
            best_score = score
            best_alpha = alpha
            best_min_df = m_df

print('Naive Bayes Best_score: {:.2f}'.format(best_score))

print('Naive Bayes Best_alpha: {:.2f}'.format(best_alpha))

print('Naive Bayes Best_min_df: {:.5f}'.format(best_min_df))
```

```
Naive Bayes Best_score: 0.82
Naive Bayes Best_alpha: 1.00
Naive Bayes Best_min_df: 0.00100
```

In [72]:
```python
# Now to create a logisitc Regression model since we covered both topics during the class:

logistic_regression_model2 = LogisticRegression()
logistic_regression_model2.fit(X_train, y_train)
logistic_regression_model2_pred = logistic_regression_model2.predict(X_test)
logistic_regression_model2_prob = logistic_regression_model2.predict_proba(X_test)[:,1]

print('Training Accuracy for TfidfVectorizer: {:.2f}'.format(logistic_regression_model2.score(X_train, y_tra

print('Testing Accuracy with TfidfVectorizer: {:.2f}'.format(logistic_regression_model2.score(X_test, y_test

print('\n Confusion matrix:')

print(confusion_matrix(y_test, logistic_regression_model2_pred))

print('\n Classification Report for Logistic Regression Model: ')

print(classification_report(y_test, logistic_regression_model2_pred))
```

```
Training Accuracy for TfidfVectorizer: 1.00
Testing Accuracy with TfidfVectorizer: 0.82

 Confusion matrix:
[[403  97]
 [ 87 413]]

 Classification Report for Logistic Regression Model:
              precision    recall  f1-score   support

           0       0.82      0.81      0.81       500
           1       0.81      0.83      0.82       500

    accuracy                           0.82      1000
   macro avg       0.82      0.82      0.82      1000
weighted avg       0.82      0.82      0.82      1000
```

In [73]:
```python
# Repeating the above steps to identify our best parameters:

param_best = np.logspace(-5,6, 8)
best_score = []
score = 0
best_c = 0

# looking at identifying our best parameters:

for value in param_best:
    svm_model1 = LinearSVC(C = value)
    score = np.mean(cross_val_score(naive_bayes_model2, X_train, y_train, scoring = 'accuracy', cv = 5))
    if score > best_score:
        best_score = score
        best_c = value

# Next steps are to rerun the training based on the best parameters found in param_best with our SVM model:
try:
    svm_model1 = LinearSVC(best_c)
    svm_model1.fit(X_train, y_train)
    svm_model1_pred = svm_model1.predict(X_test)

except:
    svm_model1 = LinearSVC()
    svm_model1.fit(X_train, y_train)
    svm_model1_pred = svm_model1.predict(X_test)

print('Training Accuracy for our TfidfVectorizer: {:.2f}'.format(svm_model1.score(X_train, y_train)))

print('Testing Accuracy with TfidfVectorizer: {:.2f}'.format(svm_model1.score(X_test, y_test)))

print('\n Confusion Matrix: ')

print(confusion_matrix(y_test, svm_model1_pred))

print('\n Classification Report')

print(classification_report(y_test, svm_model1_pred))
```

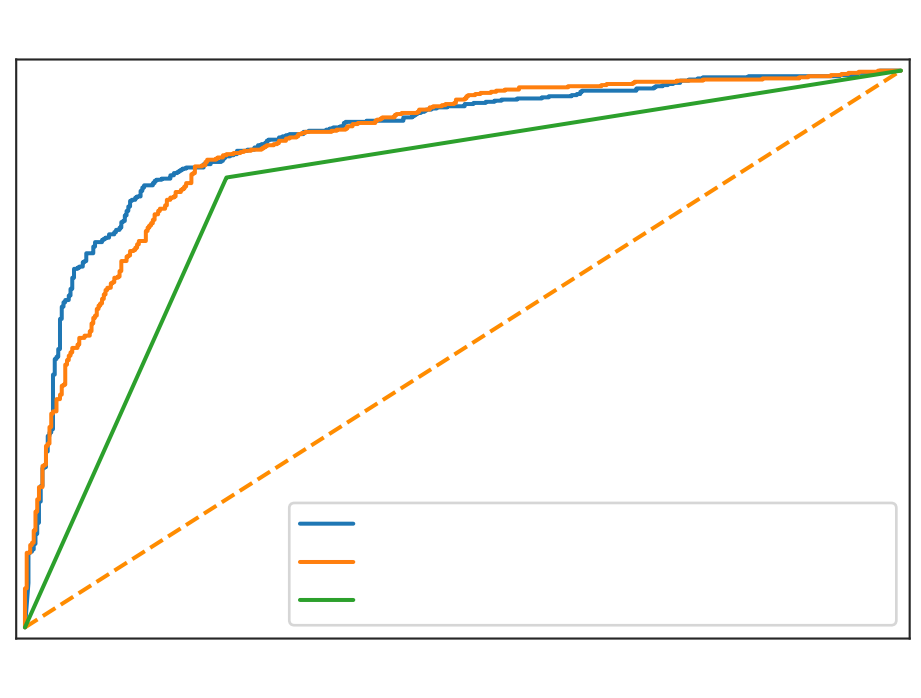
```
Training Accuracy for our TfidfVectorizer: 1.00
Testing Accuracy with TfidfVectorizer: 0.79

 Confusion Matrix:
[[385 115]
 [ 96 404]]

 Classification Report
              precision    recall  f1-score   support

           0       0.80      0.77      0.78       500
           1       0.78      0.81      0.79       500

    accuracy                           0.79      1000
   macro avg       0.79      0.79      0.79      1000
weighted avg       0.79      0.79      0.79      1000
```

In [74]:

```
# Working through our third iteration of the Naive Bayes Model:

naive_bayes_model3 = MultinomialNB(alpha = 5)
naive_bayes_model3.fit(X_train, y_train)
naive_bayes_model3_pred = naive_bayes_model3.predict(X_test)
naive_bayes_model3_prob = naive_bayes_model3.predict_proba(X_test)[:,1]
```

In [75]:

```
# Plotting our results:

sns.set_style('white')
fig, ax = plt.subplots()
ax.plot([0,1], [0,1], linestyle = '--', color = 'darkorange')

probs = [naive_bayes_model3_prob, logistic_regression_model2_prob, svm_model1_pred]
labels = ['Naive Bayes Best', 'Logistic Regression Best', 'SVC Best']
for idx in range(len(probs)):
    fpr, tpr, thresholds = roc_curve(y_test, probs[idx])
    ax.plot(fpr, tpr, label = (labels[idx] + ' AUC score = %.2f' % roc_auc_score(y_test, probs[idx])))

plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curves')
ax.legend(loc = 'lower right')
ax.margins(x = 0.01, y =0.02)
plt.show()
```



The above graphs in our notebook seem to have lost their legends when we were editing. None of us have been able to figure out why we were running into this error. We wanted to make not of that. The blue line should be our Naive Bayes model 3, orange should be our Logistic Regression model, and the green will be our SVM model 1.

In [ ]: