Assignment2

October 29, 2023

[1]:
```python
import tarfile
import os

# Define the path to the uploaded file
file_path = 'lingspam_public.tar.gz'

# Define the extraction directory
extraction_dir = 'lingspam_public'

# Extract the tar.gz file
with tarfile.open(file_path, 'r:gz') as tar:
    tar.extractall(path=extraction_dir)

# List the contents of the extracted directory
extracted_files = os.listdir(extraction_dir)
extracted_files
```

[1]:
```
['lingspam_public']
```

[2]:
```python
# Define the path to the 'lemm_stop' folder
lemm_stop_dir = os.path.join(extraction_dir, 'lingspam_public', 'lemm_stop')

# List the contents of the 'lemm_stop' directory
lemm_stop_files = os.listdir(lemm_stop_dir)
lemm_stop_files
```

[2]:
```
['part10',
 'part3',
 'part5',
 'part6',
 'part2',
 'part9',
 'part4',
 'part7',
 'part1',
 'part8']
```

[3]:

```
import email
import re

def load_and_preprocess_emails(folder_path):
    """
    Load and preprocess emails from a given folder path.

    Args:
    folder_path (str): The path to the folder containing email files.

    Returns:
    list of tuples: A list where each tuple contains the preprocessed email␣
 ↪text and its label (0 for ham, 1 for spam).
    """
    emails = []

    # List all files in the folder
    email_files = os.listdir(folder_path)

    for email_file in email_files:
        # Define the path to the email file
        file_path = os.path.join(folder_path, email_file)

        # Read the content of the email file
        with open(file_path, 'r', encoding='latin-1') as f:
            email_content = f.read()

        # Preprocess the email content
        # Convert to lowercase and tokenize
        tokens = re.findall(r'\b\w+\b', email_content.lower())
        preprocessed_email = ' '.join(tokens)

        # Label the email (0 for ham, 1 for spam)
        label = 1 if email_file.startswith('spmsg') else 0

        # Add the preprocessed email and its label to the list
        emails.append((preprocessed_email, label))

    return emails

# Load and preprocess emails from each fold
emails_by_fold = {}
for fold in lemm_stop_files:
    folder_path = os.path.join(lemm_stop_dir, fold)
    emails = load_and_preprocess_emails(folder_path)
    emails_by_fold[fold] = emails
```

```
# Display the number of emails loaded from each fold
{fold: len(emails) for fold, emails in emails_by_fold.items()}
```

[3]:
```
{'part10': 291,
 'part3': 289,
 'part5': 290,
 'part6': 289,
 'part2': 289,
 'part9': 289,
 'part4': 289,
 'part7': 289,
 'part1': 289,
 'part8': 289}
```

[4]:
```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
import numpy as np

# Combine emails from the first 9 folds for training
train_emails = [email for fold, emails in list(emails_by_fold.items())[:-1] for␣
 ↪email in emails]
train_texts, train_labels = zip(*train_emails)

# Use the 10th fold for testing
test_emails = emails_by_fold['part10']
test_texts, test_labels = zip(*test_emails)

# Convert to NumPy arrays for easier manipulation later on
train_labels = np.array(train_labels)
test_labels = np.array(test_labels)

# Create a binary CountVectorizer (for Bernoulli NB)
binary_vectorizer = CountVectorizer(binary=True)
X_train_binary = binary_vectorizer.fit_transform(train_texts)
X_test_binary = binary_vectorizer.transform(test_texts)

# Create a term frequency CountVectorizer (for Multinomial NB)
tf_vectorizer = CountVectorizer(binary=False)
X_train_tf = tf_vectorizer.fit_transform(train_texts)
X_test_tf = tf_vectorizer.transform(test_texts)

# Display the shapes of the term-document matrices
X_train_binary.shape, X_train_tf.shape
```

[4]:
```
((2604, 52180), (2604, 52180))
```

[5]:

```
from scipy.sparse import csr_matrix
from sklearn.feature_selection import SelectKBest, mutual_info_classif

def select_top_features(X, y, top_k):
    """
    Select the top-k features based on mutual information (information gain).

    Args:
    X (csr_matrix): The term-document matrix.
    y (np.array): The array of labels.
    top_k (int): The number of top features to select.

    Returns:
    csr_matrix: The reduced term-document matrix with only the top-k features.
    """
    # Use SelectKBest with mutual_info_classif to select the top-k features
    selector = SelectKBest(score_func=mutual_info_classif, k=top_k)
    X_new = selector.fit_transform(X, y)

    return X_new, selector.get_support(indices=True)

# Select top-10, top-100, and top-1000 features for both binary and TF␣
 ↪representations
X_train_binary_top10, top10_features_binary =␣
 ↪select_top_features(X_train_binary, train_labels, 10)
X_train_binary_top100, top100_features_binary =␣
 ↪select_top_features(X_train_binary, train_labels, 100)
X_train_binary_top1000, top1000_features_binary =␣
 ↪select_top_features(X_train_binary, train_labels, 1000)

X_train_tf_top10, top10_features_tf = select_top_features(X_train_tf,␣
 ↪train_labels, 10)
X_train_tf_top100, top100_features_tf = select_top_features(X_train_tf,␣
 ↪train_labels, 100)
X_train_tf_top1000, top1000_features_tf = select_top_features(X_train_tf,␣
 ↪train_labels, 1000)

# Display the shapes of the reduced term-document matrices
(X_train_binary_top10.shape, X_train_binary_top100.shape,␣
 ↪X_train_binary_top1000.shape,
 X_train_tf_top10.shape, X_train_tf_top100.shape, X_train_tf_top1000.shape)
```

[5]:

```
((2604, 10), (2604, 100), (2604, 1000), (2604, 10), (2604, 100), (2604, 1000))
```

[6]:

```
from sklearn.naive_bayes import BernoulliNB, MultinomialNB
from sklearn.metrics import precision_recall_fscore_support
import time
```

```
# Function to train a classifier and evaluate its performance
def train_and_evaluate_classifier(clf, X_train, y_train, X_test, y_test):
    start_time = time.time()
    clf.fit(X_train, y_train)
    training_time = time.time() - start_time

    start_time = time.time()
    y_pred = clf.predict(X_test)
    evaluation_time = time.time() - start_time

    precision, recall, _, _ = precision_recall_fscore_support(y_test, y_pred,␣
 ↪pos_label=1, average='binary')

    return {
        'precision': precision,
        'recall': recall,
        'training_time': training_time,
        'evaluation_time': evaluation_time
    }

# Train and evaluate Bernoulli Naive Bayes with binary features
bnb_results = {}

# For top-10 features
bnb_clf = BernoulliNB(binarize=None)
bnb_results['binary_top10'] = train_and_evaluate_classifier(bnb_clf,␣
 ↪X_train_binary_top10, train_labels, X_test_binary, test_labels)

# For top-100 features
bnb_clf = BernoulliNB(binarize=None)
bnb_results['binary_top100'] = train_and_evaluate_classifier(bnb_clf,␣
 ↪X_train_binary_top100, train_labels, X_test_binary, test_labels)

# For top-1000 features
bnb_clf = BernoulliNB(binarize=None)
bnb_results['binary_top1000'] = train_and_evaluate_classifier(bnb_clf,␣
 ↪X_train_binary_top1000, train_labels, X_test_binary, test_labels)

bnb_results
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-6-f3457ef5c4d5> in <cell line: 29>()
     27 # For top-10 features
     28 bnb_clf = BernoulliNB(binarize=None)
```

```
---> 29 bnb_results['binary_top10'] = train_and_evaluate_classifier(bnb_clf,␣
 ↪X_train_binary_top10, train_labels, X_test_binary, test_labels)
     30
     31 # For top-100 features

<ipython-input-6-f3457ef5c4d5> in train_and_evaluate_classifier(clf, X_train,␣
 ↪y_train, X_test, y_test)
     10
     11     start_time = time.time()
---> 12     y_pred = clf.predict(X_test)
     13     evaluation_time = time.time() - start_time
     14

/usr/local/lib/python3.10/dist-packages/sklearn/naive_bayes.py in predict(self,␣
 ↪X)
    103         """
    104         check_is_fitted(self)
--> 105         X = self._check_X(X)
    106         jll = self._joint_log_likelihood(X)
    107         return self.classes_[np.argmax(jll, axis=1)]

/usr/local/lib/python3.10/dist-packages/sklearn/naive_bayes.py in _check_X(self␣
 ↪X)
   1193     def _check_X(self, X):
   1194         """Validate X, used only in predict* methods."""
-> 1195         X = super()._check_X(X)
   1196         if self.binarize is not None:
   1197             X = binarize(X, threshold=self.binarize)

/usr/local/lib/python3.10/dist-packages/sklearn/naive_bayes.py in _check_X(self␣
 ↪X)
    577     def _check_X(self, X):
    578         """Validate X, used only in predict* methods."""
--> 579         return self._validate_data(X, accept_sparse="csr", reset=False)
    580
    581     def _check_X_y(self, X, y, reset=True):

/usr/local/lib/python3.10/dist-packages/sklearn/base.py in _validate_data(self,␣
 ↪X, y, reset, validate_separately, **check_params)
    586
    587         if not no_val_X and check_params.get("ensure_2d", True):
--> 588             self._check_n_features(X, reset=reset)
    589
    590         return out

/usr/local/lib/python3.10/dist-packages/sklearn/base.py in␣
 ↪_check_n_features(self, X, reset)
    387
    388         if n_features != self.n_features_in_:
--> 389             raise ValueError(
    390                 f"X has {n_features} features, but {self.__class__.
 ↪__name__} "
    391                 f"is expecting {self.n_features_in_} features as input.

ValueError: X has 52180 features, but BernoulliNB is expecting 10 features as␣
 ↪input.
```

[7]:

```python
import tarfile
import os
import email
import re
from sklearn.feature_extraction.text import CountVectorizer
import numpy as np

# Define the path to the uploaded file
file_path = 'lingspam_public.tar.gz'

# Define the extraction directory
extraction_dir = 'lingspam_public'

# Extract the tar.gz file
with tarfile.open(file_path, 'r:gz') as tar:
    tar.extractall(path=extraction_dir)

# Define the path to the 'lemm_stop' folder
lemm_stop_dir = os.path.join(extraction_dir, 'lingspam_public', 'lemm_stop')

# Function to load and preprocess emails
def load_and_preprocess_emails(folder_path):
    emails = []
    email_files = os.listdir(folder_path)
    for email_file in email_files:
        file_path = os.path.join(folder_path, email_file)
        with open(file_path, 'r', encoding='latin-1') as f:
            email_content = f.read()
        tokens = re.findall(r'\b\w+\b', email_content.lower())
        preprocessed_email = ' '.join(tokens)
        label = 1 if email_file.startswith('spmsg') else 0
        emails.append((preprocessed_email, label))
    return emails

# Load and preprocess emails from each fold
emails_by_fold = {}
for fold in os.listdir(lemm_stop_dir):
    folder_path = os.path.join(lemm_stop_dir, fold)
    emails = load_and_preprocess_emails(folder_path)
    emails_by_fold[fold] = emails

# Combine emails from the first 9 folds for training
train_emails = [email for fold, emails in list(emails_by_fold.items())[:-1] for␣
 ↪email in emails]
train_texts, train_labels = zip(*train_emails)

# Use the 10th fold for testing
test_emails = emails_by_fold['part10']
test_texts, test_labels = zip(*test_emails)

# Convert to NumPy arrays for easier manipulation later on
train_labels = np.array(train_labels)
test_labels = np.array(test_labels)

# Create term-document matrices
binary_vectorizer = CountVectorizer(binary=True)
X_train_binary = binary_vectorizer.fit_transform(train_texts)
X_test_binary = binary_vectorizer.transform(test_texts)

tf_vectorizer = CountVectorizer(binary=False)
X_train_tf = tf_vectorizer.fit_transform(train_texts)
X_test_tf = tf_vectorizer.transform(test_texts)

X_train_binary.shape, X_train_tf.shape
```

[7]:

```
((2604, 52180), (2604, 52180))
```

[8]:

```
from sklearn.feature_selection import SelectKBest, mutual_info_classif

# Function to select top-N features based on mutual information
def select_top_features(X, y, top_k):
    selector = SelectKBest(score_func=mutual_info_classif, k=top_k)
    X_new = selector.fit_transform(X, y)
    return X_new, selector.get_support(indices=True)

# Select top-10, top-100, and top-1000 features for binary features
X_train_binary_top10, top10_features_binary =␣
 ↪select_top_features(X_train_binary, train_labels, 10)
X_train_binary_top100, top100_features_binary =␣
 ↪select_top_features(X_train_binary, train_labels, 100)
X_train_binary_top1000, top1000_features_binary =␣
 ↪select_top_features(X_train_binary, train_labels, 1000)

# Select top-10, top-100, and top-1000 features for TF features
```

```
X_train_tf_top10, top10_features_tf = select_top_features(X_train_tf,␣
 ↪train_labels, 10)
X_train_tf_top100, top100_features_tf = select_top_features(X_train_tf,␣
 ↪train_labels, 100)
X_train_tf_top1000, top1000_features_tf = select_top_features(X_train_tf,␣
 ↪train_labels, 1000)

(X_train_binary_top10.shape, X_train_binary_top100.shape,␣
 ↪X_train_binary_top1000.shape,
 X_train_tf_top10.shape, X_train_tf_top100.shape, X_train_tf_top1000.shape)
```

[8]:
```
((2604, 10), (2604, 100), (2604, 1000), (2604, 10), (2604, 100), (2604, 1000))
```

[9]:
```
from sklearn.feature_selection import chi2

# Function to select top-N features based on chi-squared statistic
def select_top_features_chi2(X, y, top_k):
    selector = SelectKBest(score_func=chi2, k=top_k)
    X_new = selector.fit_transform(X, y)
    return X_new, selector.get_support(indices=True)

# Select top-10, top-100, and top-1000 features for binary features
X_train_binary_top10, top10_features_binary =␣
 ↪select_top_features_chi2(X_train_binary, train_labels, 10)
X_train_binary_top100, top100_features_binary =␣
 ↪select_top_features_chi2(X_train_binary, train_labels, 100)
X_train_binary_top1000, top1000_features_binary =␣
 ↪select_top_features_chi2(X_train_binary, train_labels, 1000)

# Select top-10, top-100, and top-1000 features for TF features
X_train_tf_top10, top10_features_tf = select_top_features_chi2(X_train_tf,␣
 ↪train_labels, 10)
X_train_tf_top100, top100_features_tf = select_top_features_chi2(X_train_tf,␣
 ↪train_labels, 100)
X_train_tf_top1000, top1000_features_tf = select_top_features_chi2(X_train_tf,␣
 ↪train_labels, 1000)

(X_train_binary_top10.shape, X_train_binary_top100.shape,␣
 ↪X_train_binary_top1000.shape,
 X_train_tf_top10.shape, X_train_tf_top100.shape, X_train_tf_top1000.shape)
```

[9]:
```
((2604, 10), (2604, 100), (2604, 1000), (2604, 10), (2604, 100), (2604, 1000))
```

[10]:
```
from sklearn.naive_bayes import BernoulliNB, MultinomialNB
from sklearn.metrics import precision_recall_fscore_support, accuracy_score
import time
```

```python
# Define a function to train and evaluate a classifier
def train_and_evaluate_classifier(clf, X_train, y_train, X_test, y_test, feature_names):
    """
    Train a classifier and evaluate its performance.

    Args:
    clf: Classifier instance.
    X_train, y_train: Training data.
    X_test, y_test: Test data.
    feature_names: List of feature names.

    Returns:
    dict: Evaluation results (accuracy, precision, recall, f1-score, latency, feature names).
    """
    start_time = time.time()
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    latency = time.time() - start_time

    precision, recall, f1, _ = precision_recall_fscore_support(y_test, y_pred, pos_label=1, average='binary')
    accuracy = accuracy_score(y_test, y_pred)

    return {
        'accuracy': accuracy,
        'precision': precision,
        'recall': recall,
        'f1': f1,
        'latency': latency,
        'features': feature_names
    }

# Define a function to get feature names based on indices
def get_feature_names(vectorizer, indices):
    return [vectorizer.get_feature_names_out()[i] for i in indices]

# Train Bernoulli Naive Bayes with binary features
clf_bnb = BernoulliNB()
results_bnb_top10 = train_and_evaluate_classifier(clf_bnb, X_train_binary_top10, train_labels, X_test_binary, test_labels, get_feature_names(binary_vectorizer, top10_features_binary))
results_bnb_top100 = train_and_evaluate_classifier(clf_bnb, X_train_binary_top100, train_labels, X_test_binary, test_labels, get_feature_names(binary_vectorizer, top100_features_binary))
```

```
results_bnb_top1000 = train_and_evaluate_classifier(clf_bnb,␣
 ↪X_train_binary_top1000, train_labels, X_test_binary, test_labels,␣
 ↪get_feature_names(binary_vectorizer, top1000_features_binary))

# Train Multinomial Naive Bayes with binary features
clf_mnb_binary = MultinomialNB()
results_mnb_binary_top10 = train_and_evaluate_classifier(clf_mnb_binary,␣
 ↪X_train_binary_top10, train_labels, X_test_binary, test_labels,␣
 ↪get_feature_names(binary_vectorizer, top10_features_binary))
results_mnb_binary_top100 = train_and_evaluate_classifier(clf_mnb_binary,␣
 ↪X_train_binary_top100, train_labels, X_test_binary, test_labels,␣
 ↪get_feature_names(binary_vectorizer, top100_features_binary))
results_mnb_binary_top1000 = train_and_evaluate_classifier(clf_mnb_binary,␣
 ↪X_train_binary_top1000, train_labels, X_test_binary, test_labels,␣
 ↪get_feature_names(binary_vectorizer, top1000_features_binary))

# Train Multinomial Naive Bayes with TF features
clf_mnb_tf = MultinomialNB()
results_mnb_tf_top10 = train_and_evaluate_classifier(clf_mnb_tf,␣
 ↪X_train_tf_top10, train_labels, X_test_tf, test_labels,␣
 ↪get_feature_names(tf_vectorizer, top10_features_tf))
results_mnb_tf_top100 = train_and_evaluate_classifier(clf_mnb_tf,␣
 ↪X_train_tf_top100, train_labels, X_test_tf, test_labels,␣
 ↪get_feature_names(tf_vectorizer, top100_features_tf))
results_mnb_tf_top1000 = train_and_evaluate_classifier(clf_mnb_tf,␣
 ↪X_train_tf_top1000, train_labels, X_test_tf, test_labels,␣
 ↪get_feature_names(tf_vectorizer, top1000_features_tf))

# Display results for Bernoulli Naive Bayes with binary features
results_bnb_top10, results_bnb_top100, results_bnb_top1000
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-10-846fbd5c323e> in <cell line: 42>()
     40 # Train Bernoulli Naive Bayes with binary features
     41 clf_bnb = BernoulliNB()
---> 42 results_bnb_top10 = train_and_evaluate_classifier(clf_bnb,␣
 ↪X_train_binary_top10, train_labels, X_test_binary, test_labels,␣
 ↪get_feature_names(binary_vectorizer, top10_features_binary))
     43 results_bnb_top100 = train_and_evaluate_classifier(clf_bnb,␣
 ↪X_train_binary_top100, train_labels, X_test_binary, test_labels,␣
 ↪get_feature_names(binary_vectorizer, top100_features_binary))
     44 results_bnb_top1000 = train_and_evaluate_classifier(clf_bnb,␣
 ↪X_train_binary_top1000, train_labels, X_test_binary, test_labels,␣
 ↪get_feature_names(binary_vectorizer, top1000_features_binary))

<ipython-input-10-846fbd5c323e> in train_and_evaluate_classifier(clf, X_train,␣
 ↪y_train, X_test, y_test, feature_names)
```

```
     19     start_time = time.time()
     20     clf.fit(X_train, y_train)
---> 21     y_pred = clf.predict(X_test)
     22     latency = time.time() - start_time
     23 

/usr/local/lib/python3.10/dist-packages/sklearn/naive_bayes.py in predict(self,␣
 ↪X)
    103         """
    104         check_is_fitted(self)
--> 105         X = self._check_X(X)
    106         jll = self._joint_log_likelihood(X)
    107         return self.classes_[np.argmax(jll, axis=1)]

/usr/local/lib/python3.10/dist-packages/sklearn/naive_bayes.py in _check_X(self ␣
 ↪X)
   1193     def _check_X(self, X):
   1194         """Validate X, used only in predict* methods."""
-> 1195         X = super()._check_X(X)
   1196         if self.binarize is not None:
   1197             X = binarize(X, threshold=self.binarize)

/usr/local/lib/python3.10/dist-packages/sklearn/naive_bayes.py in _check_X(self ␣
 ↪X)
    577     def _check_X(self, X):
    578         """Validate X, used only in predict* methods."""
--> 579         return self._validate_data(X, accept_sparse="csr", reset=False)
    580 
    581     def _check_X_y(self, X, y, reset=True):

/usr/local/lib/python3.10/dist-packages/sklearn/base.py in _validate_data(self,␣
 ↪X, y, reset, validate_separately, **check_params)
    586 
    587         if not no_val_X and check_params.get("ensure_2d", True):
--> 588             self._check_n_features(X, reset=reset)
    589 
    590         return out

/usr/local/lib/python3.10/dist-packages/sklearn/base.py in␣
 ↪_check_n_features(self, X, reset)
    387 
    388         if n_features != self.n_features_in_:
--> 389             raise ValueError(
    390                 f"X has {n_features} features, but {self.__class__.
 ↪__name__} "
    391                 f"is expecting {self.n_features_in_} features as input.
```

```
ValueError: X has 52180 features, but BernoulliNB is expecting 10 features as␣
 ↪input.
```

[11]:

```
# Apply the same feature selection transformation to the test set
X_test_binary_top10 = X_test_binary[:, top10_features_binary]
X_test_binary_top100 = X_test_binary[:, top100_features_binary]
X_test_binary_top1000 = X_test_binary[:, top1000_features_binary]

X_test_tf_top10 = X_test_tf[:, top10_features_tf]
X_test_tf_top100 = X_test_tf[:, top100_features_tf]
X_test_tf_top1000 = X_test_tf[:, top1000_features_tf]

# Retrain Bernoulli Naive Bayes with binary features
results_bnb_top10 = train_and_evaluate_classifier(clf_bnb,␣
 ↪X_train_binary_top10, train_labels, X_test_binary_top10, test_labels,␣
 ↪get_feature_names(binary_vectorizer, top10_features_binary))
results_bnb_top100 = train_and_evaluate_classifier(clf_bnb,␣
 ↪X_train_binary_top100, train_labels, X_test_binary_top100, test_labels,␣
 ↪get_feature_names(binary_vectorizer, top100_features_binary))
results_bnb_top1000 = train_and_evaluate_classifier(clf_bnb,␣
 ↪X_train_binary_top1000, train_labels, X_test_binary_top1000, test_labels,␣
 ↪get_feature_names(binary_vectorizer, top1000_features_binary))

# Retrain Multinomial Naive Bayes with binary features
results_mnb_binary_top10 = train_and_evaluate_classifier(clf_mnb_binary,␣
 ↪X_train_binary_top10, train_labels, X_test_binary_top10, test_labels,␣
 ↪get_feature_names(binary_vectorizer, top10_features_binary))
results_mnb_binary_top100 = train_and_evaluate_classifier(clf_mnb_binary,␣
 ↪X_train_binary_top100, train_labels, X_test_binary_top100, test_labels,␣
 ↪get_feature_names(binary_vectorizer, top100_features_binary))
results_mnb_binary_top1000 = train_and_evaluate_classifier(clf_mnb_binary,␣
 ↪X_train_binary_top1000, train_labels, X_test_binary_top1000, test_labels,␣
 ↪get_feature_names(binary_vectorizer, top1000_features_binary))

# Retrain Multinomial Naive Bayes with TF features
results_mnb_tf_top10 = train_and_evaluate_classifier(clf_mnb_tf,␣
 ↪X_train_tf_top10, train_labels, X_test_tf_top10, test_labels,␣
 ↪get_feature_names(tf_vectorizer, top10_features_tf))
results_mnb_tf_top100 = train_and_evaluate_classifier(clf_mnb_tf,␣
 ↪X_train_tf_top100, train_labels, X_test_tf_top100, test_labels,␣
 ↪get_feature_names(tf_vectorizer, top100_features_tf))
results_mnb_tf_top1000 = train_and_evaluate_classifier(clf_mnb_tf,␣
 ↪X_train_tf_top1000, train_labels, X_test_tf_top1000, test_labels,␣
 ↪get_feature_names(tf_vectorizer, top1000_features_tf))

# Display results for Bernoulli Naive Bayes with binary features
```

```
results_bnb_top10, results_bnb_top100, results_bnb_top1000
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-11-1ec254f25d59> in <cell line: 16>()
     14 
     15 # Retrain Multinomial Naive Bayes with binary features
---> 16 results_mnb_binary_top10 = train_and_evaluate_classifier(clf_mnb_binary ␣
  ↪X_train_binary_top10, train_labels, X_test_binary_top10, test_labels,␣
  ↪get_feature_names(binary_vectorizer, top10_features_binary))
     17 results_mnb_binary_top100 =␣
  ↪train_and_evaluate_classifier(clf_mnb_binary, X_train_binary_top100,␣
  ↪train_labels, X_test_binary_top100, test_labels,␣
  ↪get_feature_names(binary_vectorizer, top100_features_binary))
     18 results_mnb_binary_top1000 =␣
  ↪train_and_evaluate_classifier(clf_mnb_binary, X_train_binary_top1000,␣
  ↪train_labels, X_test_binary_top1000, test_labels,␣
  ↪get_feature_names(binary_vectorizer, top1000_features_binary))

NameError: name 'clf_mnb_binary' is not defined
```

[12]:

```
# Define Multinomial Naive Bayes classifiers
clf_mnb_binary = MultinomialNB()
clf_mnb_tf = MultinomialNB()

# Retrain Multinomial Naive Bayes with binary features
results_mnb_binary_top10 = train_and_evaluate_classifier(clf_mnb_binary,␣
 ↪X_train_binary_top10, train_labels, X_test_binary_top10, test_labels,␣
 ↪get_feature_names(binary_vectorizer, top10_features_binary))
results_mnb_binary_top100 = train_and_evaluate_classifier(clf_mnb_binary,␣
 ↪X_train_binary_top100, train_labels, X_test_binary_top100, test_labels,␣
 ↪get_feature_names(binary_vectorizer, top100_features_binary))
results_mnb_binary_top1000 = train_and_evaluate_classifier(clf_mnb_binary,␣
 ↪X_train_binary_top1000, train_labels, X_test_binary_top1000, test_labels,␣
 ↪get_feature_names(binary_vectorizer, top1000_features_binary))

# Retrain Multinomial Naive Bayes with TF features
results_mnb_tf_top10 = train_and_evaluate_classifier(clf_mnb_tf,␣
 ↪X_train_tf_top10, train_labels, X_test_tf_top10, test_labels,␣
 ↪get_feature_names(tf_vectorizer, top10_features_tf))
results_mnb_tf_top100 = train_and_evaluate_classifier(clf_mnb_tf,␣
 ↪X_train_tf_top100, train_labels, X_test_tf_top100, test_labels,␣
 ↪get_feature_names(tf_vectorizer, top100_features_tf))
results_mnb_tf_top1000 = train_and_evaluate_classifier(clf_mnb_tf,␣
 ↪X_train_tf_top1000, train_labels, X_test_tf_top1000, test_labels,␣
 ↪get_feature_names(tf_vectorizer, top1000_features_tf))
```

```
# Display results for Bernoulli Naive Bayes with binary features
results_bnb_top10, results_bnb_top100, results_bnb_top1000
```

```
[12]: ({'accuracy': 0.9106529209621993,
        'precision': 0.7948717948717948,
        'recall': 0.6326530612244898,
        'f1': 0.7045454545454547,
        'latency': 0.004006862640380859,
        'features': ['business',
         'click',
         'free',
         'income',
         'market',
         'million',
         'money',
         'remove',
         'save',
         'sell']},
       {'accuracy': 0.9209621993127147,
        'precision': 0.8823529411764706,
        'recall': 0.6122448979591837,
        'f1': 0.7228915662650602,
        'latency': 0.0036191940307617188,
        'features': ['100',
         'ad',
         'advertise',
         'advertisement',
         'amaze',
         'anywhere',
         'aol',
         'back',
         'best',
         'bonus',
         'bulk',
         'business',
         'buy',
         'cash',
         'cd',
         'check',
         'click',
         'com',
         'company',
         'cost',
         'credit',
         'customer',
         'day',
         'debt',
         'dollar',
         'dream',
         'earn',
         'easy',
         'enter',
         'ever',
         'every',
         'everything',
         'fantastic',
         'financial',
         'free',
         'freedom',
         'fresh',
         'friend',
         'fun',
         'guarantee',
         'hello',
         'here',
         'hour',
         'huge',
         'hundred',
         'income',
         'internet',
         'investment',
         'language',
         'linguistic',
         'live',
         'll',
         'mailing',
         'market',
         'marketing',
         'million',
         'mlm',
         'money',
         'month',
         'net',
         'never',
         'off',
         'offer',
         'online',
         'our',
         'over',
         'overnight',
         'package',
         'pay',
         'product',
         'profit',
         'profitable',
         'purchase',
         'remove',
         'reply',
         'risk',
         'sale',
         'save',
         'secret',
         'security',
         'sell',
         'service',
         'ship',
         'simply',
         'spend',
         'start',
         'step',
         'success',
         'thousand',
         'today',
         'toll',
         'university',
         'wait',
         'want',
         'watch',
         'week',
         'win',
         'yours',
         'yourself',
         'zip']},
       {'accuracy': 0.9450171821305842,
        'precision': 1.0,
        'recall': 0.673469387755102,
        'f1': 0.8048780487804877,
        'latency': 0.007846355438232422,
        'features': ['100',
         '10011',
         '1302',
         '1341',
         '147',
         '149',
         '1618',
         '195',
         '196',
         '199',
         '1995',
         '1998',
         '200',
         '209',
         '24',
         '250',
         '300',
         '3005',
         '31',
         '386',
         '399',
         '3d',
         '400',
         '409',
         '436',
         '470',
         '486',
         '50',
         '500',
         '550',
         '56k',
         '600',
         '700',
         '800',
         '888',
         '937',
         '95',
         '995',
         '9am',
         'absolutely',
         'abstract',
         'abuse',
         'ac',
         'academic',
         'acceptance',
         'access',
         'accountant',
         'accredit',
         'accurately',
         'acquisition',
         'action',
         'ad',
         'add',
         'addressed',
         'addresses',
         'adult',
         'adults',
         'advantage',
         'advertise',
         'advertisement',
         'advertiser',
         'affiliation',
         'afford',
         'affordable',
         'again',
         'against',
         'ahead',
         'aim',
         'air',
         'album',
         'allow',
         'almost',
         'alone',
         'alot',
         'already',
         'alter',
         'alway',
         'always',
         'am',
         'amateur',
         'amaze',
         'amazing',
         'amazingly',
         'amex',
         'among',
         'amount',
         'analysis',
         'anon',
         'anything',
         'anytime',
         'anywhere',
         'aol',
         'approach',
         'april',
         'are',
         'argument',
         'asked',
         'aspect',
         'asset',
         'association',
         'assuming',
         'astonishment',
         'august',
         'authenticate',
         'author',
         'authorization',
         'automatically',
         'average',
         'away',
         'awesome',
         'awhile',
         'back',
         'bad',
         'bank',
         'bankruptcy',
         'bargain',
         'basically',
         'batch',
         'beach',
         'beat',
         'beautiful',
         'before',
         'beg',
         'bel',
         'believe',
         'believer',
         'below',
         'benefits',
         'best',
         'bet',
         'better',
         'between',
         'big',
         'biggest',
         'bill',
         'billboard',
         'billion',
         'bin',
         'bit',
         'biz',
         'blast',
         'blockbuster',
         'blvd',
         'bonus',
         'book',
         'boss',
         'bottom',
         'boy',
         'boyfriend',
         'brand',
         'buddy',
         'bulk',
         'bull',
         'bureaus',
         'business',
         'button',
         'buy',
         'buyer',
         'by',
         'cable',
         'campaign',
         'cancun',
         'capital',
         'capitalfm',
         'car',
         'card',
         'cards',
         'cash',
         'casino',
         'catchy',
         'cd',
         'cds',
         'celebrate',
         'celebrity',
         'cent',
         'centre',
         'cgi',
         'chair',
         'chance',
         'channel',
         'charge',
         'chat',
         'cheap',
         'check',
         'checks',
         'choose',
         'cinema',
         'city',
         'classified',
         'clean',
         'cleanest',
         'clearance',
         'click',
         'client',
         'cloak',
         'clog',
         'clothe',
         'code',
         'cognitive',
         'colby',
         'collector',
         'color',
         'com',
         'comfort',
         'commercialemail',
         'commercialize',
         'committee',
         'company',
         'comparative',
         'competition',
         'competitor',
         'completely',
         'compliance',
         'comply',
         'compuserve',
         'computational',
         'compzone',
         'conceal',
         'conference',
         'confident',
         'confidential',
         'conservative',
         'constraint',
         'construction',
         'consumer',
         'context',
         'continual',
         'contribution',
         'convenience',
         'convince',
         'copyright',
         'corporation',
         'corporations',
         'corpus',
         'correctly',
         'cost',
         'countless',
         'cram',
         'create',
         'credit',
         'creditor',
         'criminal',
         'customer',
         'cut',
         'cyber',
         'daily',
         'dare',
         'datum',
         'dave',
         'david',
         'day',
         'days',
         'de',
         'deadline',
         'debt',
         'decide',
         'delete',
         'deliver',
         'delivery',
         'delphus',
         'department',
         'deposit',
         'description',
         'desire',
         'desirous',
         'desktop',
         'development',
         'dial',
         'dialect',
         'did',
         'didn',
         'diploma',
         'discourse',
         'discover',
         'discuss',
         'discussion',
         'divorce',
         'djs',
         'doesn',
         'dollar',
         'dollars',
         'don',
         'dori',
         'doubt',
         'down',
         'downline',
         'download',
         'downpayment',
         'dramatically',
         'dream',
         'driver',
         'drop',
         'dupe',
         'duplicate',
         'each',
         'earn',
         'earnings',
         'earth',
         'ease',
         'easiest',
         'easily',
         'easy',
         'ed',
         'edu',
         'effective',
         'effort',
         'eliminate',
         'else',
         'emailamendtext',
         'emailer',
         'embark',
         'employee',
         'engine',
         'english',
         'enjoy',
         'enter',
         'enterprise',
         'entertainment',
         'entire',
         'entrepreneur',
         'envelope',
         'error',
         'esq',
         'estate',
         'european',
         'evaluating',
         'even',
         'ever',
         'every',
         'everyday',
         'everyone',
         'everythe',
         'everything',
         'evidence',
         'exactly',
         'exceedingly',
         'excellent',
         'excess',
         'excite',
         'exclusive',
         'exp',
         'expensive',
         'expiration',
         'explode',
         'extra',
         'extractor',
         'extraordinary',
         'extremely',
         'ez',
         'fabulous',
         'fairchild',
         'faith',
         'family',
         'fantastic',
         'fast',
         'faster',
         'fastest',
         'favourite',
         'fax',
         'federal',
         'few',
         'fill',
         'filled',
         'filter',
         'finance',
         'financial',
         'financially',
         'finest',
         'fingertip',
         'fl',
         'flame',
         'flamer',
         'fm',
         'focus',
         'forever',
         'forget',
         'formal',
         'fortunately',
         'fortune',
         'fraction',
         'framework',
         'free',
         'freedom',
         'french',
         'fresh',
         'freshest',
         'friend',
         'friends',
         'frown',
         'fulfill',
         'fun',
         'future',
         'gamble',
         'game',
         'gay',
         'general',
         'generate',
         'genie',
         'german',
         'germany',
         'gift',
         'girdfriend',
         'girl',
         'girlfriend',
         'gold',
         'golden',
         'goodness',
         'goods',
         'gov',
         'grammar',
         'grammatical',
         'great',
         'greatest',
         'grow',
         'grumble',
         'guarantee',
         'guaranteed',
         'hand',
         'happen',
         'happy',
         'hardcore',
         'hello',
         'help',
         'here',
         'hesitate',
         'hi',
         'historical',
         'hit',
         'hobby',
         'hold',
         'holiday',
         'home',
         'homeowner',
         'homosexual',
         'honest',
         'hot',
         'hotline',
         'hotmail',
         'hottest',
         'hour',
         'hours',
         'hr',
         'huge',
         'hundred',
         'hundreds',
         'hurry',
         'husband',
         'id',
         'illegal',
         'imagination',
         'imagine',
         'immediate',
         'immediately',
         'included',
         'income',
         'increase',
         'incredible',
         'industry',
         'inexpensive',
         'inflation',
         'info',
         'infoseek',
         'install',
         'instant',
         'instantly',
         'institute',
         'instruct',
         'instructed',
         'instruction',
         'instructions',
         'insurance',
         'interaction',
         'internet',
         'interpretation',
         'introduction',
         'intrusion',
         'invest',
         'investment',
         'investor',
         'invite',
         'is',
         'isdn',
         'isp',
         'issue',
         'jackson',
         'job',
         'john',
         'join',
         'jump',
         'june',
         'junk',
         'juno',
         'keep',
         'keystroke',
         'kid',
         'kitchen',
         'knock',
         'language',
         'lanse',
         'largest',
         'latest',
         'laugh',
         'launch',
         'law',
         'lawful',
         'leave',
         'legal',
         'legally',
         'legitimate',
         'lesbian',
         'less',
         'let',
         'letter',
         'lexical',
         'lexicon',
         'life',
         'lifetime',
         'likes',
         'limited',
         'line',
         'linguist',
         'linguistic',
         'linguistics',
         'list',
         'lists',
         'literature',
         'little',
         'live',
         'living',
         'll',
         'llc',
         'load',
         'loader',
         'loan',
         'lose',
         'lot',
         'lottery',
         'love',
         'low',
         'luck',
         'lucky',
         'lyco',
         'magazine',
         'mailbox',
         'mailer',
         'mailing',
         'make',
         'making',
         'manual',
         'manufacturer',
         'many',
         'market',
         'marketer',
         'marketing',
         'mastercard',
         'mba',
         'mci',
         'mclaughlin',
         'meg',
         'mega',
         'merciless',
         'message',
         'million',
         'millionaire',
         'millions',
         'miss',
         'mlm',
         'model',
         'modem',
         'modern',
         'moment',
         'money',
         'month',
         'monthly',
         'morphology',
         'mortgage',
         'most',
         'mouse',
         'move',
         'movie',
         'moving',
         'msn',
         'much',
         'muncie',
         'murkowskus',
         'natural',
         'nc',
         'need',
         'net',
         'never',
         'newest',
         'news',
         'newsgroup',
         'newsletter',
         'next',
         'nl',
         'nothing',
         'notification',
         'numbers',
         'obligation',
         'obviously',
         'off',
         'offer',
         'office',
         'offshore',
         'once',
         'online',
         'operate',
         'opportunity',
         'order',
         'ordering',
         'orders',
         'organize',
         'originator',
         'our',
         'ours',
         'over',
         'overflow',
         'overload',
         'overnight',
         'owe',
         'own',
         'owner',
         'pack',
         'package',
         'paid',
         'paper',
         'papers',
         'paradise',
         'particular',
         'partner',
         'pass',
         'password',
         'paste',
         'patient',
         'pattern',
         'pay',
         'payable',
         'pc',
         'pegasus',
         'penny',
         'pentium',
         'per',
         'percent',
         'percentage',
         'perfectly',
         'permanently',
         'persistent',
         'personal',
         'perspective',
         'phillip',
         'phone',
         'phonological',
         'phonology',
         'pick',
         'piece',
         'pile',
         'pipeline',
         'plans',
         'platinum',
         'plus',
         'poorer',
         'pop',
         'porn',
         'postage',
         'postmaster',
         'potential',
         'powerful',
         'pp',
         'practically',
         'pragmatic',
         'premium',
         'prepared',
         'present',
         'presentation',
         'preview',
         'price',
         'print',
         'privacy',
         'prize',
         'proceedings',
         'prodigy',
         'product',
         'products',
         'profanity',
         'professional',
         'professor',
         'profit',
         'profitable',
         'profits',
         'programme',
         'programs',
         'promo',
         'promotion',
         'promotional',
         'prompt',
         'promptly',
         'proof',
         'proposal',
         'protect',
         'protection',
         'proud',
         'prove',
         'proven',
         'provider',
         'publication',
         'publish',
         'purchase',
         'put',
         'qualify',
         'query',
         'quick',
         'quickly',
         'quit',
         'radio',
         'raleigh',
         'ram',
         'rat',
         'rate',
         're',
         'reach',
         'read',
         'real',
         'really',
         'reap',
         'receive',
         'recession',
         'recieve',
         'recipient',
         'record',
         'recruit',
         'reference',
         'referral',
         'refinance',
         'refund',
         'reg',
         'registration',
         'relation',
         'relax',
         'release',
         'released',
         'relevant',
         'remember',
         'removal',
         'remove',
         'removed',
         'rental',
         'reply',
         'report',
         'reports',
         'representation',
         'reprinting',
         'request',
         'requesting',
         'required',
         'resale',
         'research',
         'researcher',
         'resell',
         'reselling',
         'residual',
         'respectability',
         'respectfully',
         'responsive',
         'retail',
         'retire',
         'retirement',
         'return',
         'returns',
         'revenue',
         'reward',
         'richer',
         'right',
         'rights',
         'rip',
         'risk',
         'robbery',
         'robbie',
         'rockland',
         'role',
         'roll',
         'rom',
         'run',
         'rush',
         'sale',
         'sales',
         'satisfaction',
         'satisfy',
         'save',
         'savings',
         'scam',
         'scary',
         'science',
         'search',
         'secret',
         'secrets',
         'secure',
         'security',
         'sell',
         'selle',
         'seller',
         'semantic',
         'semantics',
         'sender',
         'sent',
         'sentence',
         'server',
         'service',
         'services',
         'session',
         'setup',
         'seven',
         'sex',
         'sexiest',
         'sexual',
         'sexually',
         'ship',
         'shock',
         'shop',
         'shot',
         'show',
         'signature',
         'simple',
         'simply',
         'sincerely',
         'sit',
         'site',
         'skeptical',
         'sleep',
         'smart',
         'smtp',
         'society',
         'sociolinguistic',
         'software',
         'solid',
         'someday',
         'someone',
         'soon',
         'sooner',
         'sorry',
         'soundblaster',
         'sources',
         'spam',
         'speak',
         'speaker',
         'specials',
         'speech',
         'speed',
         'spend',
         'spokane',
         'spout',
         'staggering',
         'stamp',
         'stamped',
         'star',
         'start',
         'started',
         'stealth',
         'step',
         'steps',
         'stock',
         'stop',
         'store',
         'storm',
         'strip',
         'structure',
         'student',
         'study',
         'stun',
         'submission',
         'submit',
         'succeed',
         'success',
         'successful',
         'suite',
         'summary',
         'super',
         'supplies',
         'sure',
         'surf',
         'sweepstakes',
         'syntactic',
         'syntax',
         'tarrant',
         'tax',
         'technician',
         'teen',
         'tel',
         'television',
         'tell',
         'testimonial',
         'thank',
         'theme',
         'theoretical',
         'theory',
         'thereafter',
         'thing',
         'thousand',
         'thousands',
         'ticket',
         'tip',
         'tips',
         'today',
         'toll',
         'top',
         'topic',
         'total',
         'totally',
         'totals',
         'toy',
         'trade',
         'trail',
         'translation',
         'trash',
         'tremendous',
         'trial',
         'trouble',
         'true',
         'truly',
         'trust',
         'try',
         'tune',
         'turn',
         'tv',
         'txt',
         'uk',
         'unbelievable',
         'undeliverable',
         'undoubtedly',
         'unemployment',
         'unique',
         'university',
         'unlimit',
         'unlimited',
         'unlist',
         'unproductive',
         'unsolicit',
         'unsubscribe',
         'until',
         'unwant',
         'upgrade',
         'upset',
         'us',
         'utility',
         'vacation',
         'van',
         'vanish',
         'variety',
         'vcs',
         've',
         'vendor',
         'verb',
         'verify',
         'vga',
         'video',
         'visa',
         'visit',
         'vist',
         'vulgarity',
         'wait',
         'wall',
         'want',
         'warning',
         'warranty',
         'waste',
         'watch',
         'wealth',
         'wealthiest',
         'wealthy',
         'webmaster',
         'week',
         'weekend',
         'weekly',
         'weeks',
         'welcome',
         'whatsoever',
         'why',
         'wilburn',
         'win',
         'win95',
         'window',
         'winner',
         'wisely',
         'wish',
         'wonderful',
         'word',
         'works',
         'workshop',
         'worldwide',
         'worth',
         'wrap',
         'wrhel',
         'xxx',
         'yahoo',
         'ye',
         'yes',
         'yours',
         'yourself',
         'zip',
         'zone']})
```

```
[ ]:
```