Setup

Resources

```
# Install packages
!pip install opendatasets
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: opendatasets in /usr/local/lib/python3.9/dist-packages (0.1.22)
Requirement already satisfied: tqdm in /usr/local/lib/python3.9/dist-packages (from opendatasets) (4.65.0)
Requirement already satisfied: click in /usr/local/lib/python3.9/dist-packages (from opendatasets) (8.1.3)
Requirement already satisfied: kaggle in /usr/local/lib/python3.9/dist-packages (from opendatasets) (1.5.13)
Requirement already satisfied: requests in /usr/local/lib/python3.9/dist-packages (from kaggle->opendatasets) (2.27.1)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.9/dist-packages (from kaggle->opendatasets) (1.26.15)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.9/dist-packages (from kaggle->opendatasets) (2.8.2)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.9/dist-packages (from kaggle->opendatasets) (8.0.1)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.9/dist-packages (from kaggle->opendatasets) (1.16.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.9/dist-packages (from kaggle->opendatasets) (2022.12.7)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.9/dist-packages (from python-slugify->kaggle->opendatasets) (1.3)
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.9/dist-packages (from requests->kaggle->opendatasets) (2.0.12)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-packages (from requests->kaggle->opendatasets) (3.4)
```

```
# Imports
import sklearn as sk
import opendatasets as od
import pandas as pd
import nltk
import csv
import json
import math
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB, BernoulliNB
from sklearn.linear_model import LogisticRegression
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, log_loss
```

```
# Downloads

# Dataset
# {"username":"noahagonzo","key":"b7e4d1aedc1148c648f8fcef1ab58905"}
od.download("https://www.kaggle.com/datasets/jabara/freud-detection")

# stopwords
nltk.download('stopwords')
```

```
Skipping, found downloaded files in "./freud-detection" (use force=True to force download)
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
True
```

```
# Setup constants
STOPWORDS = list(set(stopwords.words('english')))
vectorizer = TfidfVectorizer(stop_words = STOPWORDS, binary=True)
```

Import Text Data

```
# Import training data
df = pd.read_csv('freud-detection/Freud_Detection_Train.csv', header=0, encoding='utf-8')
# 30 from Sigmund Freud, 30 from Jane Austen, 30 from Mark Twain, and 30 from Maya Angelou
df.shape
print(df.head())
```

```
                                               quote author  freud
0  One day, in retrospect, the years of struggle ...  Freud      1
1  Being entirely honest with oneself is a good e...  Freud      1
2  Unexpressed emotions will never die. They are ...  Freud      1
3  Most people do not really want freedom, becaus...  Freud      1
4  We are never so defenseless against suffering ...  Freud      1
```

```
# Import test data (just going to combine the two)
df_test = pd.read_csv('freud-detection/Freud_Detection_Test.csv', header=0, encoding='utf-8')
# 10 from Sigmund Freud, 10 from Jane Austen, 10 from Mark Twain, and 10 from Maya Angelou
df_test.shape
print(df_test.head())
```

```
                                               quote author  freud
0            America is a mistake, a giant mistake.  Freud      1
1  The intention that man should be happy is not ...  Freud      1
2  My love is something valuable to me which I ou...  Freud      1
3  Men are more moral than they think and far mor...  Freud      1
4  A man should not strive to eliminate his compl...  Freud      1
```

```
# Combine data
df = pd.concat([df, df_test], join='inner')
```

Preprocess text

```
df['quote'].replace('[\d][\d]+', ' num ', regex=True, inplace=True)
df['quote'].replace('[!@#*][!@#*]+', ' punct ', regex=True, inplace=True)
df['quote'].replace('[A-Z][A-Z]+', ' caps ', regex=True, inplace=True)
```

Divide train/test

```
X = df.quote
y = df.freud

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, train_size=0.8, random_state=1234)
```

Convert to Numeric Data

```
# apply tfidf vectorizer
X_train = vectorizer.fit_transform(X_train)  # fit and transform the train data
X_test = vectorizer.transform(X_test)        # transform only the test data
```

```
# take a peek at the data
# this is a very sparse matrix because most of the 8613 words don't occur in each sms message

print('train size:', X_train.shape)
print(X_train.toarray()[:5])

print('\ntest size:', X_test.shape)
print(X_test.toarray()[:5])
```

```
train size: (128, 683)
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]

test size: (32, 683)
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
```

Naive Bayes

```
# Train
naive_bayes = BernoulliNB()
naive_bayes.fit(X_train, y_train)

# Output
MultinomialNB(alpha = 1.0, class_prior=None, fit_prior=True)
```

▾ MultinomialNB
MultinomialNB()

```
# priors
prior_p = sum(y_train == 1)/len(y_train)
print('prior freud:', prior_p, 'log of prior:', math.log(prior_p))

# the model prior matches the prior calculated above
naive_bayes.class_log_prior_[1]
```

```
prior freud: 0.2421875 log of prior: -1.4180430594344708
-1.4180430594344706
```

```
# what else did it learn from the data?
# the log likelihood of words given the class

naive_bayes.feature_log_prob_
```

```
array([[-3.90197267, -3.90197267, -3.90197267, ..., -3.20882549,
        -3.90197267, -3.90197267],
       [-3.49650756, -3.49650756, -3.49650756, ..., -2.80336038,
        -3.49650756, -2.80336038]])
```

```
# make predictions on the test data
pred = naive_bayes.predict(X_test)

# Confusion matrix
print(confusion_matrix(y_test, pred))

# confusion matrix has this form
#     tp   fp
#     fn   tn
```

```
[[23  0]
 [ 9  0]]
```

```
print('accuracy score: ', accuracy_score(y_test, pred))

print('\nprecision score: ', precision_score(y_test, pred, pos_label=0))

print('\nrecall score: ', recall_score(y_test, pred, pos_label=0))

print('\nf1 score: ', f1_score(y_test, pred, pos_label=0))
```

```
accuracy score:  0.71875

precision score:  0.71875

recall score:  1.0

f1 score:  0.8363636363636363
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test, pred))
```

```
                precision    recall  f1-score   support

             0       0.72      1.00      0.84        23
             1       0.00      0.00      0.00         9

      accuracy                           0.72        32
     macro avg       0.36      0.50      0.42        32
  weighted avg       0.52      0.72      0.60        32

/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in label
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in label
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in label
  _warn_prf(average, modifier, msg_start, len(result))
```

Logistic Regression

```
# train using the same dataset
classifier = LogisticRegression(solver='lbfgs', class_weight='balanced')
classifier.fit(X_train, y_train)
```

```
LogisticRegression(class_weight='balanced')
```

```
# evaluate
pred = classifier.predict(X_test)
print('accuracy score: ', accuracy_score(y_test, pred))
print('precision score: ', precision_score(y_test, pred, pos_label=0))
print('recall score: ', recall_score(y_test, pred, pos_label=0))
print('f1 score: ', f1_score(y_test, pred, pos_label=0))
probs = classifier.predict_proba(X_test)
```

```
accuracy score:  0.71875
precision score:  0.7333333333333333
recall score:  0.9565217391304348
f1 score:  0.8301886792452831
```

Neural Network

```
# train
classifier = MLPClassifier(solver='lbfgs', alpha=1e-5,
                   hidden_layer_sizes=(15, 2), random_state=1)
classifier.fit(X_train, y_train)
```

```
MLPClassifier(alpha=1e-05, hidden_layer_sizes=(15, 2), random_state=1,
              solver='lbfgs')
```

```
# Evaluate
pred = classifier.predict(X_test)
print('accuracy score: ', accuracy_score(y_test, pred))
print('precision score: ', precision_score(y_test, pred, pos_label=0))
print('recall score: ', recall_score(y_test, pred, pos_label=0))
print('f1 score: ', f1_score(y_test, pred, pos_label=0))
```

```
accuracy score:  0.71875
precision score:  0.75
recall score:  0.9130434782608695
f1 score:  0.8235294117647057
```

Final Analysis

The accuracy for all three methods was, strangely, the same: 0.71875. Accuracy is the percentage of correctly classified examples in the test set. Only 71.875% of samples were correctly classified. All of these methods are faily poor at distinguishing between quotes. I don't think that this dataset was large enough. I also think that the quotes were very similar themselves and there wasn't much of a difference between them.

For the methods besides naive-Bayes, the precision was 75%. Precision measures how many observations that were classfied as P, really are P. Of the samples classified as freud, only 75% of the classifications were correct. This is acceptable, but not reliable. Naive-Bayes had an even lower precision at 71.875%.

Naive-Bayes had the highest recall at 100% while the other methods both had a recall of 91.3%. Recall measures how many true P observations were found. Most of these methods reliable identified most freud quotes.

It seems that Naive-bayes sacrificed precision for a greater recall. This results in it having a slightly better f1 score (0.836 vs 0.824). I think this is because it assumes independence and isn't as affected by the similarity in quotes.