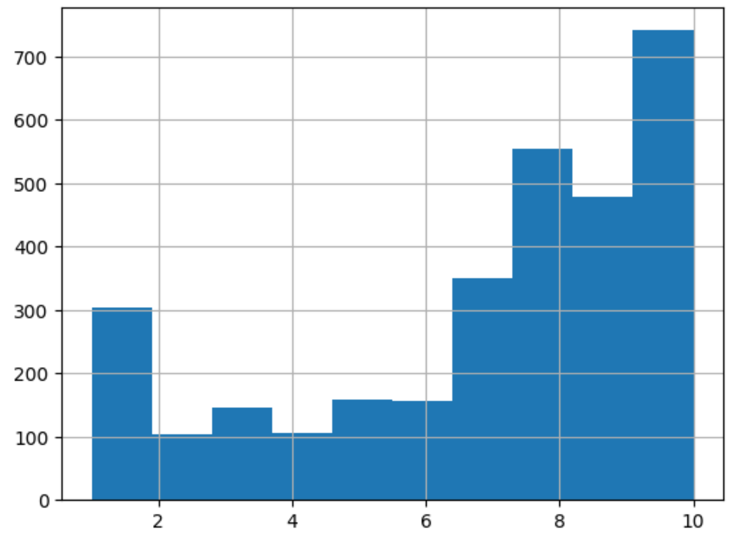
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import cross_val_score
from sklearn.svm import LinearSVC
from sklearn.metrics import accuracy_score
from sklearn.pipeline import Pipeline
from sklearn.metrics import f1_score
import pickle
```

```
df = pd.read_csv("train (1).csv")
df2 = df.copy()
```

```
#EDA
df.dtypes
df.rating.hist()
# most ratings are 6 and above
```

```
<Axes: >
```



```
def rating(rating):
    if rating == 1:
        return 'negative'
    elif rating == 2:
        return 'negative'
    elif rating == 3:
        return 'negative'
    elif rating == 4:
        return 'negative'
    else:
        return 'positive'
df2.rating = df2.rating.apply(rating)
```

```
X = df2.loc[:, ['benefits_review','side_effects_review','comments_review']]
y = df2.rating
X, y = df2.comments_review.fillna(' '), df2.rating
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,stratify=y)
```

```
 X_train_docs = [doc for doc in X_train]
```

```
import spacy
import re
from spacy.lang.en.stop_words import STOP_WORDS
en_nlp = spacy.load('en_core_web_sm', disable=['parser', 'ner'])
pattern = re.compile('(?u)\\b\\w\\w+\\b')
```

```
import spacy
nlp = spacy.load("en_core_web_sm")
```

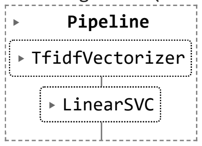
using a spacy tokenizer similar to in-class

```
def custom_tokenizer(document):
  doc_spacy = en_nlp(document)
  lemmas = [token.lemma_ for token in doc_spacy]
  pattern = re.compile('(?u)\\b\\w\\w+\\b')
  return [token for token in lemmas if token not in STOP_WORDS and pattern.match(token)]
```

```
pipeline = Pipeline([
  ('vect', TfidfVectorizer(tokenizer = custom_tokenizer )),
  ('cls',LinearSVC())
])
```

```
pipeline.fit(X_train_docs, y_train)
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/feature_extraction/text.py:528: UserWarning: The pa
  warnings.warn(
```



```
cross_val_score(pipeline, X_train_docs, y_train, cv=5).mean()
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/feature_extraction/text.py:528: UserWarning: The parameter 'token_pattern' will not be us
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/feature_extraction/text.py:528: UserWarning: The parameter 'token_pattern' will not be us
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/feature_extraction/text.py:528: UserWarning: The parameter 'token_pattern' will not be us
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/feature_extraction/text.py:528: UserWarning: The parameter 'token_pattern' will not be us
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/feature_extraction/text.py:528: UserWarning: The parameter 'token_pattern' will not be us
  warnings.warn(
0.7812613744000171
```

```
predicted= pipeline.predict([doc for doc in X_test])
```

```
accuracy_score(y_test, predicted)
```

```
0.7763440860215054
```

```
f1_score(y_test, predicted, pos_label='positive')
```

```
0.868520859671302
```

```
df_sample = pd.read_csv("test (1).csv")
```

```
df_sample['review'] = df_sample['benefits_review'].astype(str) + df_sample['side_effects_review'].astype(str) + df_sample['comments_review'].
df_sample.rating.shape
```

```
(1036,)
```

```
X_sample = df_sample.review
```

```
def rating(rating):
    if rating == 1:
        return 'negative'
    elif rating == 2:
        return 'negative'
    elif rating == 3:
        return 'negative'
    elif rating == 4:
        return 'negative'
    else:
        return 'positive'
df_sample.rating = df_sample.rating.apply(rating)
```

```
y_sample= df_sample.rating
y_sample.shape
```

```
(1036,)
```

```
predicted_sample = pipeline.predict([doc for doc in X_sample])
```

```
accuracy_score(y_sample, predicted_sample)
```

```
0.88996138996139
```

```
f1_score(y_sample, predicted_sample, pos_label='positive')
```

```
0.9417773237997957
```

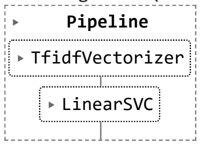
Using a basic Spacy tokenizer

```
def spacy_tokenizer(doc):
    tokens = nlp(doc)
    return [token.text for token in tokens if not token.is_stop and not token.is_punct and not token.is_space]
```

```
pipeline2 = Pipeline([
  ('vect', TfidfVectorizer(tokenizer = spacy_tokenizer)),
  ('cls',LinearSVC())
])
```

```
pipeline2.fit(X_train_docs, y_train)
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/feature_extraction/text.py:528: UserWarning: The par
  warnings.warn(
```



```
cross_val_score(pipeline2, X_train_docs, y_train, cv=5).mean()
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/feature_extraction/text.py:528: UserWarning: The parameter 'token_pattern' will not be us
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/feature_extraction/text.py:528: UserWarning: The parameter 'token_pattern' will not be us
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/feature_extraction/text.py:528: UserWarning: The parameter 'token_pattern' will not be us
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/feature_extraction/text.py:528: UserWarning: The parameter 'token_pattern' will not be us
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/feature_extraction/text.py:528: UserWarning: The parameter 'token_pattern' will not be us
  warnings.warn(
0.7840231585445026
```

```
predicted2= pipeline2.predict([doc for doc in X_test])
```

```
accuracy_score(y_test, predicted2)
```

```
0.7720430107526882
```

```
f1_score(y_test, predicted2, pos_label='positive')
```

```
0.8663303909205549
```

```
predicted2_sample = pipeline2.predict([doc for doc in X_sample])
```

```
accuracy_score(y_sample, predicted2_sample)
```

```
0.9015444015444015
```

```
f1_score(y_sample, predicted2_sample, pos_label = 'positive')
```

```
0.948223350253807
```

Using a spacy tokenizer with POS

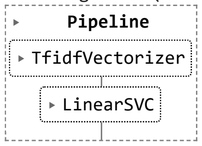
```
nlp = spacy.load("en_core_web_sm")

def custom_tokenizer2(document):
    doc_spacy = nlp(document)
    return [token.lemma_ for token in doc_spacy if token.pos_ in ['ADJ', 'NOUN']]
```

```
pipeline3 = Pipeline([
  ('vect', TfidfVectorizer(tokenizer = custom_tokenizer2)),
  ('cls',LinearSVC())
])
```

```
pipeline3.fit(X_train_docs, y_train)
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/feature_extraction/text.py:528: UserWarning: The pa
  warnings.warn(
```



```
cross_val_score(pipeline3, X_train_docs, y_train, cv=5).mean()
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/feature_extraction/text.py:528: UserWarning: The parameter 'token_pattern' will not be us
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/feature_extraction/text.py:528: UserWarning: The parameter 'token_pattern' will not be us
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/feature_extraction/text.py:528: UserWarning: The parameter 'token_pattern' will not be us
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/feature_extraction/text.py:528: UserWarning: The parameter 'token_pattern' will not be us
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/feature_extraction/text.py:528: UserWarning: The parameter 'token_pattern' will not be us
  warnings.warn(
0.7867923925884144
```

```
predicted3= pipeline3.predict([doc for doc in X_test])
```

```
accuracy_score(y_test, predicted3)
```

```
0.7623655913978494
```

```
f1_score(y_test, predicted3, pos_label='positive')
```

```
0.8603916614024004
```

```
predicted_sample3 = pipeline3.predict([doc for doc in X_sample])
```

```
accuracy_score(y_sample, predicted_sample3)
```

```
0.8841698841698842
```

```
f1_score(y_sample, predicted_sample3, pos_label='positive')
```

```
0.9385245901639344
```

Using both POS & STOP_words

```
def custom_tokenizer3(text):
    doc = nlp(text)
    return [token.lemma_ for token in doc if not token.is_stop and token.pos_ in ['NOUN', 'ADJ', 'VERB', 'ADV']]
```

```
pipeline4 = Pipeline([
  ('vect', TfidfVectorizer(tokenizer = custom_tokenizer3)),
  ('cls',LinearSVC())
])
```

```
pipeline4.fit(X_train_docs, y_train)
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/feature_extraction/text.py:528: UserWarning: The pai
  warnings.warn(
▸ Pipeline
▸ TfidfVectorizer
▸ LinearSVC
```

```
cross_val_score(pipeline4, X_train_docs, y_train, cv=5).mean()
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/feature_extraction/text.py:528: UserWarning: The parameter 'token_pattern' will not be us
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/feature_extraction/text.py:528: UserWarning: The parameter 'token_pattern' will not be us
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/feature_extraction/text.py:528: UserWarning: The parameter 'token_pattern' will not be us
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/feature_extraction/text.py:528: UserWarning: The parameter 'token_pattern' will not be us
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/feature_extraction/text.py:528: UserWarning: The parameter 'token_pattern' will not be us
  warnings.warn(
0.7923415033897042
```

```
predicted4 = pipeline4.predict([doc for doc in X_test])
```

```
accuracy_score(y_test, predicted4)
```

```
0.7806451612903226
```

```
f1_score(y_test, predicted4, pos_label = 'positive')
```

```
0.8710493046776232
```

```
predicted_sample4 = pipeline4.predict([doc for doc in X_sample])
```

```
accuracy_score(y_sample, predicted_sample4)
```

```
0.8841698841698842
```

```
f1_score(y_sample, predicted_sample4, pos_label='positive')
```

```
0.9385245901639344
```

Using lexicons, stop words, and POS

```
!pip install afinn
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting afinn
  Downloading afinn-0.1.tar.gz (52 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 52.6/52.6 kB 2.6 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: afinn
  Building wheel for afinn (setup.py) ... done
  Created wheel for afinn: filename=afinn-0.1-py3-none-any.whl size=53448 sha256=a7a721eb742fe5c70646aaa7279e648750cad9224314666d8575242
  Stored in directory: /root/.cache/pip/wheels/79/91/ee/8374d9bc8c6c0896a2db75afdfd63d43653902407a0e76cd94
Successfully built afinn
Installing collected packages: afinn
Successfully installed afinn-0.1
```

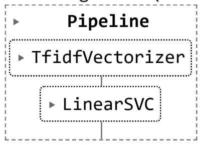
```
from afinn import Afinn
afinn = Afinn()
```

```
def custom_tokenizer5(doc):
    doc = nlp(doc)
    tokens = [token.lemma_.lower() for token in doc if token.pos_ in ["ADJ", "VERB", "NOUN"] and token.lemma_.lower() not in STOP_WORDS]
    sentiment = afinn.score(" ".join(tokens))
    return tokens
```

```
pipeline5 = Pipeline([
  ('vect', TfidfVectorizer(tokenizer = custom_tokenizer5)),
  ('cls',LinearSVC())
])
```

```
pipeline5.fit(X_train_docs, y_train)
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/feature_extraction/text.py:528: UserWarning: The par
  warnings.warn(
```



```
cross_val_score(pipeline5, X_train_docs, y_train, cv=5).mean()
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/feature_extraction/text.py:528: UserWarning: The parameter 'token_pattern' will not be us
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/feature_extraction/text.py:528: UserWarning: The parameter 'token_pattern' will not be us
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/feature_extraction/text.py:528: UserWarning: The parameter 'token_pattern' will not be us
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/feature_extraction/text.py:528: UserWarning: The parameter 'token_pattern' will not be us
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/feature_extraction/text.py:528: UserWarning: The parameter 'token_pattern' will not be us
  warnings.warn(
0.7891061184959718
```

```
predicted5 = pipeline5.predict([doc for doc in X_test])
```

```
accuracy_score(y_test, predicted5)
```

```
0.7698924731182796
```

```
f1_score(y_test, predicted5, pos_label = 'positive')
```

```
0.8638676844783716
```

```
predicted_sample5 = pipeline5.predict([doc for doc in X_sample])
```

```
accuracy_score(y_sample, predicted_sample5)
```

```
0.8754826254826255
```

```
f1_score(y_sample, predicted_sample5, pos_label='positive')
```

```
0.9336078229541944
```

Transforming Pipeline2 into a pickle file for Streamlit

```
# assume pipeline is defined and trained
with open('ML_Streamlit_Pipeline.pkl', 'wb') as f:
    pickle.dump(pipeline2, f)
```