```
!pip install openai
!pip install tweepy
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting openai
  Downloading openai-0.26.5.tar.gz (55 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 55.5/55.5 KB 1.5 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Installing backend dependencies ... done
  Preparing metadata (pyproject.toml) ... done
Requirement already satisfied: requests>=2.20 in /usr/local/lib/python3.8/dist-packages (from openai) (2.25.1)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.8/dist-packages (from openai) (3.8.4)
Requirement already satisfied: tqdm in /usr/local/lib/python3.8/dist-packages (from openai) (4.64.1)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.8/dist-packages (from requests>=2.20->openai) (1.24.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.8/dist-packages (from requests>=2.20->openai) (2022.12.7)
Requirement already satisfied: chardet<5,>=3.0.2 in /usr/local/lib/python3.8/dist-packages (from requests>=2.20->openai) (4.0.0)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.8/dist-packages (from requests>=2.20->openai) (2.10)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.8/dist-packages (from aiohttp->openai) (1.8.2)
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in /usr/local/lib/python3.8/dist-packages (from aiohttp->openai) (4.0.2)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.8/dist-packages (from aiohttp->openai) (6.0.4)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.8/dist-packages (from aiohttp->openai) (1.3.3)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.8/dist-packages (from aiohttp->openai) (22.2.0)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.8/dist-packages (from aiohttp->openai) (1.3.1)
Requirement already satisfied: charset-normalizer<4.0,>=2.0 in /usr/local/lib/python3.8/dist-packages (from aiohttp->openai) (3.0.1)
Building wheels for collected packages: openai
  Building wheel for openai (pyproject.toml) ... done
  Created wheel for openai: filename=openai-0.26.5-py3-none-any.whl size=67620 sha256=c745a604abd73b722c40a714da4a4c56301481d13d3a46de6
  Stored in directory: /root/.cache/pip/wheels/a7/47/99/8273a59fbd59c303e8ff175416d5c1c9c03a2e83ebf7525a99
Successfully built openai
Installing collected packages: openai
Successfully installed openai-0.26.5
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: tweepy in /usr/local/lib/python3.8/dist-packages (3.10.0)
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.8/dist-packages (from tweepy) (1.15.0)
Requirement already satisfied: requests[socks]>=2.11.1 in /usr/local/lib/python3.8/dist-packages (from tweepy) (2.25.1)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.8/dist-packages (from tweepy) (1.3.1)
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.8/dist-packages (from requests-oauthlib>=0.7.0->tweepy) (3.2.2
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.8/dist-packages (from requests[socks]>=2.11.1->tweepy) (
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.8/dist-packages (from requests[socks]>=2.11.1->tweepy) (202
Requirement already satisfied: chardet<5,>=3.0.2 in /usr/local/lib/python3.8/dist-packages (from requests[socks]>=2.11.1->tweepy) (4.0.
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.8/dist-packages (from requests[socks]>=2.11.1->tweepy) (2.10)
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in /usr/local/lib/python3.8/dist-packages (from requests[socks]>=2.11.1->tweepy)
```

```
import tweepy
import pandas as pd
import openai
import time
import os
```

Api Secret key

```
import os

with open("api_keys_new.py", 'r', encoding='utf-8') as api_file:
    api_keys = api_file.readlines()

for key in api_keys:
    if "bearer_token" in key:
        twitter_api_key = key.split('"')
    elif "openai_token" in key:
        openai_api_key = key.split('"')
```

Scrapping tweet data

```
def get_tweets_by_handle(handle: str, api_key: str):
    client = tweepy.Client(bearer_token=twitter_api_key, wait_on_rate_limit=True)
    user_id = client.get_user(username=handle).data.id
    tweets_data = client.get_users_tweets(id=user_id, max_results=100,
```

```
                                        tweet_fields=['id', 'text', 'created_at', 'context_annotations']).data
    tweets = {
        'handle': [],
        'tweet': [],
        'id': [],

    }

    for tweet in tweets_data:
        tweets['handle'].append(handle)
        tweets['tweet'].append(tweet.text)
        tweets['id'].append(tweet.id)
    tweets = pd.DataFrame(tweets)
    return tweets
```

Label tweet with OpenAI

```
def check_deal_from_tweet(tweet: str, api_key: str) -> bool:
    openai.api_key = api_key
    prompt_text = tweet + ' contains any deal information. True or False?'
    response = openai.Completion.create(model="text-ada-001",
                                        prompt=prompt_text,
                                        temperature=0, n=1,
                                        max_tokens=7).choices[0].text
    time.sleep(5)
    if 'true' in response.lower():
        return True
    else:
        return False
```

```
#handle = 'RedFlagDeals'
##hasDeal = []
#tweets = get_tweets_by_handle(handle, twitter_api_key)

#for id, row in tweets.iterrows():
    ##hasDeal.append(check_deal_from_tweet(tweet=row['tweet'], api_key=openai_api_key))

#tweets['has_deal'] = hasDeal
```

export tweet data to csv

```
#tweets.to_csv('tweet_data.csv', index=False)
```

Preprocessing

```
import numpy as np
import re
import nltk
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
```

```
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
```

```
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
True
```

```
def process_data(data):
    documents = []
    stemmer = WordNetLemmatizer()

    for sen in range(0, len(data)):
        # Remove all the special characters
        document = re.sub(r'\W', ' ', str(data[sen]))
        # remove all single characters
        document = re.sub(r'\s+[a-zA-Z]\s+', ' ', document)
        # Remove single characters from the start
        document = re.sub(r'\^[a-zA-Z]\s+', ' ', document)
        # Substituting multiple spaces with single space
        document = re.sub(r'\s+', ' ', document, flags=re.I)
        # Removing prefixed 'b'
        document = re.sub(r'^b\s+', '', document)
        # Converting to Lowercase
        document = document.lower()
        # Lemmatization
        document = document.split()
        document = [stemmer.lemmatize(word) for word in document]
        document = ' '.join(document)
        documents.append(document)
    return documents
```

```
df = pd.read_csv('tweet_data.csv')
df.head()
```

	handle	tweet	id	has_deal
0	RedFlagDeals	The seasonal clearance event is on now at @hud...	1624423014182928385	False
1	RedFlagDeals	Get Recipe For Disaster for free this week at ...	1624211618635587586	False

```
y = df['has_deal']
X = df['tweet']
```

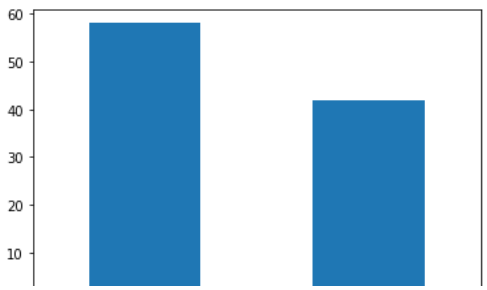
```
text_arr = process_data(X)
```

```
tfidfconverter = TfidfVectorizer(max_features=1500, stop_words=stopwords.words('english'))
X = tfidfconverter.fit_transform(text_arr).toarray()
```

EDA

```
df['has_deal'].value_counts().plot(kind='bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f70c1e96820>
```



```
print("There are %i corpus" %(len(text_arr)))
print("Percentage of corpus with deal:", (len(df[df['has_deal']==True])))
print("Percentage of words without deal", (len(df[df['has_deal']==False])))
```

```
There are 100 corpus
Percentage of corpus with deal: 42
Percentage of words without deal 58
```

```
from wordcloud import WordCloud, STOPWORDS

stopwords = set(STOPWORDS)
wordcloud = WordCloud(width = 800, height = 400,
            background_color ='white',
            stopwords = stopwords,
            min_font_size = 10).generate(' '.join(text_arr))

plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```

Machine learning

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15, random_state=20)
```

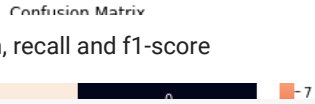
```
classifier = RandomForestClassifier(n_estimators=1000, random_state=0)
classifier.fit(X_train, y_train)
```

```
RandomForestClassifier(n_estimators=1000, random_state=0)
```

Confusion matrix

```
y_pred = classifier.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
```

```
ax= plt.subplot()
sns.heatmap(cm, annot=True, fmt='g', ax=ax);
# labels, title and ticks
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.xaxis.set_ticklabels(['False','True']); ax.yaxis.set_ticklabels(['False','True']);
ax.set_title('Confusion Matrix');
```



Calculate precision, recall and f1-score

```
print("Report model\n")
print(classification_report(y_pred, y_test))
```

```
     Report model

                  precision    recall  f1-score   support

           False       1.00      0.75      0.86        12
            True       0.50      1.00      0.67         3

        accuracy                           0.80        15
       macro avg       0.75      0.88      0.76        15
    weighted avg       0.90      0.80      0.82        15
```

✓ 0s completed at 6:11 PM