```
! pip install kaggle
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.10/dist-packages (1.5.16)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.10/dist-packages (from kaggle) (1.16.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from kaggle) (2024.2.2)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.8.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.31.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from kaggle) (4.66.2)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.10/dist-packages (from kaggle) (8.0.4)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.0.7)
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages (from kaggle) (6.1.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-packages (from bleach->kaggle) (0.5.1)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.10/dist-packages (from python-slugify->kagg
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->kaggl
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle) (3.6)
```

```
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
```

```
!kaggle datasets download -d kazanova/sentiment140
```

```
Downloading sentiment140.zip to /content
 88% 71.0M/80.9M [00:00<00:00, 265MB/s]
100% 80.9M/80.9M [00:00<00:00, 258MB/s]
```

```
from zipfile import ZipFile
ds='/content/sentiment140.zip'
with ZipFile(ds,'r') as e:
  e.extractall()
  print("Done")
```

```
Done
```

```
import numpy as np
import pandas as pd
import re
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

```
import nltk
nltk.download('stopwords')
print(stopwords.words('english'))
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'you
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
from zipfile import ZipFile
ds='/content/sentiment140.zip'
with ZipFile(ds,'r') as e:
  e.extractall()
  print("Done")
```

```
Done
```

```
import numpy as np
import pandas as pd
import re
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

```
import nltk
nltk.download('stopwords')
print(stopwords.words('english'))
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'you
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
td=pd.read_csv('/content/training.1600000.processed.noemoticon.csv', encoding="ISO-8859-1")
```

```
td.shape
```

```
(1599999, 6)
```

```
td.head()
```

	0	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_	@switchfo http://twitpic.com/2y1 - Awww, that's a bumme You shoulda got Dav Carr of Third Day to it.
0	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update h Facebook by
1	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many tim for the ball. Mar
			Mon Apr			

```
cn=['target', 'id', 'date', 'flag', 'user', 'text']
td=pd.read_csv('/content/training.1600000.processed.noemoticon.csv',names=cn, encoding="ISO-8859-1")
```

```
td.shape
```

```
(1600000, 6)
```

```
td.head()
```

	target	id	date	flag	user	text
0	0	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zl - Awww, t...
1	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by ...

```
td.isnull().sum()
```

```
target    0
id        0
date      0
flag      0
user      0
text      0
dtype: int64
```

```
td['target'].value_counts()
```

```
0    800000
4    800000
Name: target, dtype: int64
```

```
td.replace({'target': {4:1}},inplace=True)
td['target'].value_counts()
```

```
0    800000
1    800000
Name: target, dtype: int64
```

```
ps=PorterStemmer()
```

```
def stemming(sc):
    sc = re.sub('[^a-zA-Z]', ' ', sc)
    sc = sc.lower()
    sc = sc.split()
    sc = [ps.stem(word) for word in sc if word not in stopwords.words('english')]
    sc = ' '.join(sc)
    return sc
```

```
#td['stemmed_content'] = td['text'].apply(stemming)
```

```
import pickle
```

```
td['sc']=pickle.load(open('/content/stemmed_model.sav', 'rb'))
```

```
print(td['sc'])
```

```
0          switchfoot http twitpic com zl awww bummer sho...
1          upset updat facebook text might cri result sch...
2          kenichan dive mani time ball manag save rest g...
3                            whole bodi feel itchi like fire
4                              nationwideclass behav mad see
                                 ...
1599995                           woke school best feel ever
1599996    thewdb com cool hear old walt interview http b...
1599997                          readi mojo makeov ask detail
1599998    happi th birthday boo alll time tupac amaru sh...
1599999    happi charitytuesday thenspcc sparkschar speak...
Name: sc, Length: 1600000, dtype: object
```

```
print(td['target'])
```

```
0          0
1          0
2          0
3          0
4          0
          ..
1599995    1
1599996    1
1599997    1
1599998    1
1599999    1
Name: target, Length: 1600000, dtype: int64
```

```
x=td['sc'].values
```

```
y=td['target'].values
```

```
print(x)
```

```
['switchfoot http twitpic com zl awww bummer shoulda got david carr third day'
 'upset updat facebook text might cri result school today also blah'
 'kenichan dive mani time ball manag save rest go bound' ...
 'readi mojo makeov ask detail'
 'happi th birthday boo alll time tupac amaru shakur'
 'happi charitytuesday thenspcc sparkschar speakinguph h']
```

```
print(y)
```

```
[0 0 0 ... 1 1 1]
```

```
x_train,x_test,y_train,y_test=train_test_split(x, y, test_size=0.3, stratify=y, random_state=2)
```

```
print(x.shape, x_train.shape, x_test.shape)
```

```
(1600000,) (1120000,) (480000,)
```

```
v=TfidfVectorizer()
```

```
x_train=v.fit_transform(x_train)
```

```
x_test=v.transform(x_test)
```

```
print(x_train)
```

```
  (0, 374778)	0.31949818170660904
  (0, 328004)	0.5314491135690684
  (0, 43969)	0.4748653389926461
  (0, 306319)	0.4371164314704369
  (0, 121348)	0.44599875194306554
  (1, 360908)	0.20232270128356875
  (1, 45367)	0.24454930105105102
  (1, 32222)	0.2885646935506797
  (1, 400139)	0.5008486447803013
  (1, 36162)	0.21760975021723114
  (1, 376966)	0.16230856596272633
  (1, 224362)	0.24716194206021685
  (1, 2941)	0.3329517305729671
  (1, 28528)	0.1699742942038265
  (1, 377777)	0.5390797727604116
  (2, 209149)	0.36506885244060977
  (2, 217307)	0.35862123416780045
  (2, 331146)	0.45772430309259293
  (2, 192775)	0.7270515839609045
  (3, 54144)	0.3405985100366321
  (3, 381278)	0.3044716859774938
  (3, 114004)	0.3165169852577187
  (3, 344001)	0.3565976148233519
  (3, 366377)	0.4018889828996008
  (3, 139076)	0.25759117714273316
  :	:
  (1119996, 327788)	0.2626904633863284
  (1119996, 133766)	0.14165568151908714
  (1119996, 400064)	0.17675763929108915
  (1119996, 366962)	0.1645582384006955
  (1119996, 121102)	0.2083143336548096
  (1119996, 166244)	0.2582309496394417
  (1119997, 364520)	0.7210223776952688
  (1119997, 176159)	0.6575712947692484
  (1119997, 366962)	0.21846446658016408
  (1119998, 384964)	0.23515032977930925
  (1119998, 26391)	0.22101418705941295
  (1119998, 255402)	0.23115328947076547
  (1119998, 313788)	0.49883995385230084
  (1119998, 405750)	0.597655663753657
  (1119998, 252433)	0.34998949156158465
  (1119998, 56066)	0.20742679758052754
  (1119998, 306613)	0.14119239340870796
  (1119998, 155263)	0.14334431819822027
  (1119998, 36162)	0.17432788141763644
  (1119999, 88171)	0.5404842471107606
  (1119999, 124023)	0.61479006915859
  (1119999, 397207)	0.36018431710547344
  (1119999, 28754)	0.24841651195142356
  (1119999, 399014)	0.31666662934784157
  (1119999, 221882)	0.19541926631660167
```

```
model = LogisticRegression(max_iter=1000)


model.fit(x_train, y_train)


y_train_pred = model.predict(x_train)


y_test_pred = model.predict(x_test)


training_accuracy = accuracy_score(y_train, y_train_pred)


testing_accuracy = accuracy_score(y_test, y_test_pred)


print(f'Training Accuracy: {training_accuracy}')
print(f'Testing Accuracy: {testing_accuracy}')
```

```
    Training Accuracy: 0.8106169642857143
    Testing Accuracy: 0.7776708333333333
```

```
from sklearn.metrics import precision_score, recall_score, f1_score

precision = precision_score(y_test, y_test_pred)
recall = recall_score(y_test, y_test_pred)
f1 = f1_score(y_test, y_test_pred)


print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1 Score: {f1:.4f}")
```

```
    Precision: 0.7671
    Recall: 0.7974
    F1 Score: 0.7820
```

```
def predict_sentiment(text):

    text = stemming(text)

    text_vector = v.transform([text])

    sentiment = model.predict(text_vector)

    return 'Positive' if sentiment == 1 else 'Negative'


new_sentence = "Movie is Good to watch";

print(f"The sentiment of the sentence is: {predict_sentiment(new_sentence)}")
```

```
    The sentiment of the sentence is: Positive
```

```
from sklearn.naive_bayes import MultinomialNB
m2 = MultinomialNB()
m2.fit(x_train, y_train)
x_train_pred = m2.predict(x_train)
trd_acc = accuracy_score(y_train, x_train_pred)
print('Accuracy on Training Data:', trd_acc)
x_test_pred = m2.predict(x_test)
tsd_acc = accuracy_score(y_test, x_test_pred)
print('Accuracy on Test Data:', tsd_acc)
```

```
    Accuracy on Training Data: 0.8216964285714285
    Accuracy on Test Data: 0.7554416666666667
```

```
from sklearn.metrics import precision_score, recall_score, f1_score
precision = precision_score(y_test, x_test_pred)
recall = recall_score(y_test, x_test_pred)
f1 = f1_score(y_test, x_test_pred)

print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1 Score: {f1}')
```

```
    Precision: 0.7709082713943561
    Recall: 0.7268958333333333
    F1 Score: 0.7482554074861998
```

```
from sklearn.linear_model import SGDClassifier
sgd_clf = SGDClassifier(random_state=2)
sgd_clf.fit(x_train, y_train)
x_train_pred = sgd_clf.predict(x_train)
trd_acc = accuracy_score(y_train, x_train_pred)
print('Accuracy on Training Data:', trd_acc)


x_test_pred = sgd_clf.predict(x_test)
tsd_acc = accuracy_score(y_test, x_test_pred)
print('Accuracy on Test Data:', tsd_acc)

precision = precision_score(y_test, x_test_pred)
recall = recall_score(y_test, x_test_pred)
f1 = f1_score(y_test, x_test_pred)
```

```
    Accuracy on Training Data: 0.75661875
    Accuracy on Test Data: 0.7561770833333333
```

```
print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1 Score: {f1}')
```

```
    Precision: 0.7271165716692094
    Recall: 0.8201541666666666
    F1 Score: 0.7708381876959286
```

```
from sklearn.svm import SVC
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score
svc_clf = SVC(random_state=2)
svc_clf.fit(x_train, y_train)
```