

KNN:

برای نشان دادن نتایج این قسمت از روش کسینوس که سرعتش بیشتر و اکثراً دقتش کمی بهتر از فاصله اقلیدسی شده استفاده می‌کنیم:

K = 1:

```
KNN
/usr/bin/python3.6 /home/esra/Desktop/MIR_Phase2/KNN.py
Confusion Matrix:
571.000  53.000  56.000  57.000
50.000  604.000  49.000  53.000
66.000  52.000  544.000  99.000
63.000  41.000  101.000  541.000
precision: [0.7747625508819539, 0.798941798941799, 0.7148488830486203, 0.725201072386059]   mean precision: 0.7534385763146081
recall: [0.7613333333333333, 0.8053333333333333, 0.7253333333333334, 0.7213333333333334]   mean recall: 0.7533333333333333
f1: [0.7679892400806994, 0.8021248339973439, 0.7200529450694904, 0.7232620320855615]   mean f1: 0.7533572628082738
accuracy: 0.7533333333333333

Process finished with exit code 0
```

K = 3:

```
KNN
/usr/bin/python3.6 /home/esra/Desktop/MIR_Phase2/KNN.py
Confusion Matrix:
598.000  38.000  54.000  51.000
47.000  632.000  50.000  47.000
63.000  47.000  559.000  94.000
42.000  33.000  87.000  558.000
precision: [0.8070175438596491, 0.8144329896907216, 0.7326343381389253, 0.775]   mean precision: 0.782271217922324
recall: [0.7973333333333333, 0.8426666666666667, 0.7453333333333333, 0.744]   mean recall: 0.7823333333333333
f1: [0.8021462105969148, 0.8283093053735257, 0.7389292795769993, 0.7591836734693876]   mean f1: 0.7821421172542068
accuracy: 0.7823333333333333

Process finished with exit code 0
```

K = 5:

```
Confusion Matrix:
300.000  32.000  32.000  39.000
20.000  303.000  16.000  24.000
39.000  23.000  277.000  39.000
24.000  16.000  54.000  262.000
precision: [0.7444168734491315, 0.8347107438016529, 0.7328042328042328, 0.7359550561797753]   mean precision: 0.761971726558698
recall: [0.783289817232376, 0.8101604278074866, 0.7308707124010554, 0.7197802197802198]   mean recall: 0.7610252943052844
f1: [0.7633587786259542, 0.8222523744911805, 0.7318361955085865, 0.7277777777777777]   mean f1: 0.7613062816008748
accuracy: 0.7613333333333333
```

بنابراین به نظر می‌رسد K=3 بهترین مقداردهی باشد.

Naïve Bayes: با امتحان کردن مقادیر ۱، ۵، ۱۰، ۲۰، و ۵۰، بهترین مقدار alpha عددی حدود ۱۰ بود که به ازای $\alpha=10.4$ الگوریتم برای این مجموعه داده به ما دقت ۸۹.۱ درصد می‌دهد.

Nltk:

(1 KNN با $K=3$ ، Cosine:

(برای کمتر شدن زمان اجرا این قسمت را با نصف داده‌ها انجام دادم)

Nothing:

```
KNN x
Confusion Matrix:
296.000  19.000  23.000  30.000
 24.000 312.000  23.000  32.000
 41.000  25.000 275.000  44.000
 22.000  18.000  58.000 258.000
precision: [0.8043478260869565, 0.7979539641943734, 0.7142857142857143, 0.7247191011235955]   mean precision: 0.76032665142266
recall: [0.7728459530026109, 0.8342245989304813, 0.725936675461742, 0.7087912087912088]   mean recall: 0.7603638570676189
f1: [0.7882822902796273, 0.8156862745098039, 0.7198952879581153, 0.7166666666666667]   mean f1: 0.7601326298535533
accuracy: 0.7606666666666667

Process finished with exit code 0
```

Stopword Removal:

```
/usr/bin/python3.6 /home/esra/Desktop/MIR_Phase2/KNN.py
Confusion Matrix:
299.000  23.000  26.000  28.000
 30.000 305.000  23.000  22.000
 27.000  30.000 280.000  46.000
 27.000  16.000  50.000 268.000
precision: [0.7952127659574468, 0.8026315789473685, 0.7310704960835509, 0.7423822714681441]   mean precision: 0.7678242781141276
recall: [0.7806788511749347, 0.8155080213903744, 0.7387862796833773, 0.7362637362637363]   mean recall: 0.7678092221281057
f1: [0.7878787878787878, 0.8090185676392574, 0.7349081364829397, 0.7393103448275862]   mean f1: 0.7677789592071428
accuracy: 0.768

Process finished with exit code 0
```

Lemmatization:

```
Run: KNN
/usr/bin/python3.6 /home/esra/Desktop/MIR_Phase2/KNN.py
Confusion Matrix:
301.000  28.000  27.000  29.000
26.000  302.000  17.000  23.000
25.000  19.000  282.000  44.000
31.000  25.000  53.000  268.000

precision: [0.7818181818181819, 0.8206521739130435, 0.7621621621621621, 0.7108753315649867]    mean precision: 0.7688769623645935
recall: [0.7859007832898173, 0.8074866310160428, 0.7440633245382586, 0.7362637362637363]    mean recall: 0.7684286187769638
f1: [0.7838541666666667, 0.8140161725067384, 0.753004005340454, 0.7233468286099864]    mean f1: 0.768552932809614
accuracy: 0.7686666666666667

Process finished with exit code 0
```

Stemming:

```
Run: KNN
/usr/bin/python3.6 /home/esra/Desktop/MIR_Phase2/KNN.py
Confusion Matrix:
299.000  27.000  30.000  30.000
24.000  303.000  20.000  18.000
29.000  22.000  280.000  47.000
31.000  22.000  49.000  269.000

precision: [0.7746113989637305, 0.8301369863013699, 0.7407407407407407, 0.7250673854447439]    mean precision: 0.7676391278626462
recall: [0.7806788511749347, 0.8101604278074866, 0.7387862796833773, 0.739010989010989]    mean recall: 0.767159136919197
f1: [0.7776332899869962, 0.8200270635994588, 0.7397622192866579, 0.7319727891156463]    mean f1: 0.7673488404971899
accuracy: 0.7673333333333333

Process finished with exit code 0
```

البته اگر به جای SnowballStemmer از LancasterStemmer استفاده کنیم که با شدت بیشتری کلمات را استم می‌کند، دقت پایین می‌آید:

```
Run: KNN
fanciness_decoder: List[bool] = [False, False, True]
/usr/bin/python3.6 /home/esra/Desktop/MIR_Phase2/KNN.py
Confusion Matrix:
308.000  31.000  35.000  32.000
27.000  292.000  22.000  20.000
25.000  28.000  274.000  43.000
23.000  23.000  48.000  269.000

precision: [0.7586206896551724, 0.8088642659279779, 0.7405405405405405, 0.7410468319559229]    mean precision: 0.7622680820199035
recall: [0.804177545691906, 0.7807486631016043, 0.7229551451187335, 0.739010989010989]    mean recall: 0.7617230857308082
f1: [0.7807351077313055, 0.7945578231292517, 0.7316421895861148, 0.7400275103163687]    mean f1: 0.7617406576907602
accuracy: 0.762

Process finished with exit code 0
```

مشاهده می‌کنیم که با پیش‌پردازش متن با کمک nltk دقت در حد چند دهم درصد زیاد می‌شود

naïve Bayes: (2

Nothing:

```

Confusion Matrix:
668.000  11.000  31.000  31.000
29.000  730.000  11.000  11.000
29.000   4.000  644.000  72.000
24.000   5.000  64.000  636.000
precision: [0.9014844804318488, 0.9346991037131882, 0.8598130841121495, 0.8724279835390947]    mean precision: 0.8921061629490703
recall: [0.8906666666666667, 0.9733333333333334, 0.8586666666666667, 0.848]    mean recall: 0.8926666666666667
f1: [0.8960429242119383, 0.953625081645983, 0.8592394929953302, 0.8600405679513186]    mean f1: 0.8922370167011425
accuracy: 0.8926666666666667

Process finished with exit code 0
|

```

Stopword Removal:

```

/usr/bin/python3.6 /home/esra/Desktop/MIR_Phase2/naive_bayes.py
Confusion Matrix:
667.000  12.000  27.000  31.000
27.000  728.000   7.000   6.000
33.000   5.000  651.000  74.000
23.000   5.000  65.000  639.000
precision: [0.9050203527815468, 0.9479166666666666, 0.8532110091743119, 0.8729508196721312]    mean precision: 0.8947747120736642
recall: [0.8893333333333333, 0.9706666666666667, 0.868, 0.852]    mean recall: 0.8949999999999999
f1: [0.8971082716879623, 0.9591567852437418, 0.8605419695968275, 0.8623481781376517]    mean f1: 0.8947888011665458
accuracy: 0.895

Process finished with exit code 0
|

```

Lemmatization:

```

Confusion Matrix:
673.000  12.000  30.000  32.000
28.000  730.000  11.000   9.000
26.000   5.000  646.000  69.000
23.000   3.000  63.000  640.000
precision: [0.9009370816599732, 0.9383033419023136, 0.8659517426273459, 0.877914951989026]    mean precision: 0.8957767795446646
recall: [0.8973333333333333, 0.9733333333333334, 0.8613333333333333, 0.8533333333333334]    mean recall: 0.8963333333333334
f1: [0.8991315965263862, 0.9554973821989527, 0.8636363636363635, 0.8654496281271129]    mean f1: 0.8959287426222038
accuracy: 0.8963333333333333

Process finished with exit code 0

```

Stemming:

```

Confusion Matrix:
672.000  11.000  30.000  34.000
 28.000 731.000  11.000  10.000
 26.000   5.000 643.000  73.000
 24.000   3.000 66.000 633.000
precision: [0.8995983935742972, 0.9371794871794872, 0.8607764390896921, 0.871900826446281] mean precision: 0.8923637865724393
recall: [0.896, 0.9746666666666667, 0.8573333333333333, 0.844] mean recall: 0.8929999999999999
f1: [0.8977955911823646, 0.9555555555555556, 0.8590514362057449, 0.8577235772357723] mean f1: 0.8925315400448592
accuracy: 0.893

Process finished with exit code 0

```

اینجا هم اگر به جای SnowballStemmer از LancasterStemmer استفاده کنیم، دقت پایین می‌آید:

```

Confusion Matrix:
675.000  10.000  34.000  32.000
 30.000 729.000  15.000  13.000
 26.000   6.000 633.000  70.000
 19.000   5.000 68.000 635.000
precision: [0.8988015978695073, 0.9263024142312579, 0.8612244897959184, 0.8734525447042641] mean precision: 0.889945261650237
recall: [0.9, 0.972, 0.844, 0.8466666666666667] mean recall: 0.8906666666666666
f1: [0.8994003997335109, 0.9486011711125569, 0.8525252525252526, 0.8598510494245092] mean f1: 0.8900944681989574
accuracy: 0.8906666666666667

Process finished with exit code 0

```

اینجا هم مشاهده می‌کنیم که با پیش‌پردازش متن با کمک nltk دقت در حد دهم درصد زیاد می‌شود.

:SVM

C = 1

```

▶ clf = svm.SVC(kernel='linear')

print("I got here")
clf.fit(X_train, y_train)
print("I got here 2")
y_pred = clf.predict(X_test)

acc = 0
for i in range(len(y_test)):
    if y_test[i] == y_pred[i]:
        acc += 1

print(acc/ len(y_test))

```

```

↳ I got here
I got here 2
0.8133333333333334

```

C = 5.5

```

▶ clf = svm.SVC(kernel='linear', C=5.5)

print("I got here")
clf.fit(X_train, y_train)
print("I got here 2")
y_pred = clf.predict(X_test)

acc = 0
for i in range(len(y_test)):
    if y_test[i] == y_pred[i]:
        acc += 1

print(acc/ len(y_test))

```

```

↳ I got here
I got here 2
0.8066666666666666

```

C = 10

```
clf = svm.SVC(kernel='linear', C=10)

print("I got here")
clf.fit(X_train, y_train)
print("I got here 2")
y_pred = clf.predict(X_test)

acc = 0
for i in range(len(y_test)):
    if y_test[i] == y_pred[i]:
        acc += 1

print(acc/ len(y_test))
```

I got here
I got here 2
0.8066666666666666

بنابراین به نظر می‌رسد $C=5$ خوبی باشد و fit کردن مدل بیش از آن فایده‌ی قابل‌توجهی نداشته باشد.

Random Forest:

به نظر می‌رسد وقتی Max_Depth می‌گذاریم دقت بدتر می‌شود ولی وقتی تعداد درختان را زیاد می‌کنیم دقت به مقدار اندکی زیاد می‌شود:

$\text{max_depth} = \text{None}$; $\text{N_estimators} = 100$

```
Run: random_forest x
/usr/bin/python3.6 /home/esra/Desktop/MIR_Phase2/random_forest.py
I got here
I got here 2
0.8126666666666666

Process finished with exit code 0
```

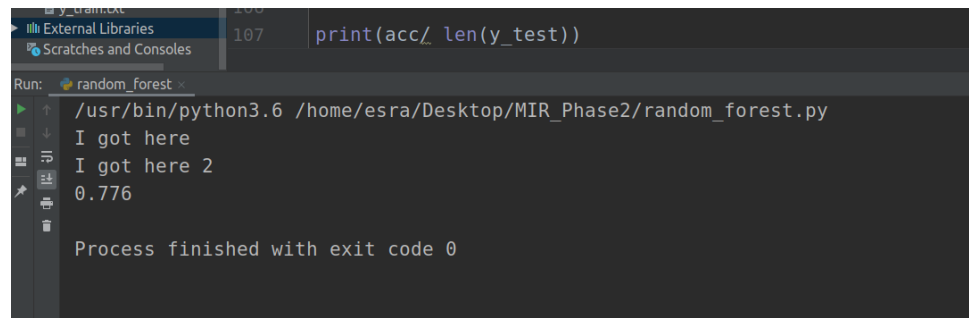
$\text{max_depth} = \text{None}$; $\text{N_estimators} = 500$

```
Run: random_forest x
/usr/bin/python3.6 /home/esra/
I got here
I got here 2
0.818

Process finished with exit code 0
```

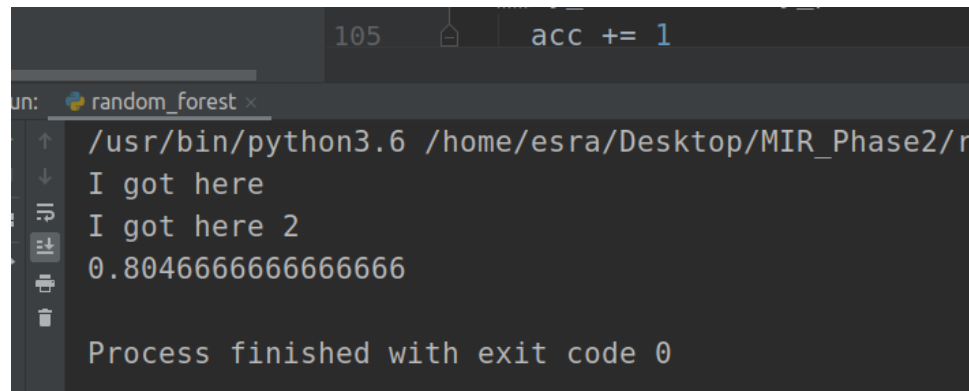
Return an object that produces a sequence of integers from start (inclusive) to stop (exclusive) by step. range(i, j) produces i, i+1, i+2, ..., j-1. start defaults to 0, and stop is omitted! range(4) produces 0, 1, 2, 3. These are exactly the valid indices for a list of 4 elements. When step is given, it specifies the increment (or decrement).
Documentation is copied from: [range](#)
Python 3.6 >

max_depth = 12: ,N_estimators = 500



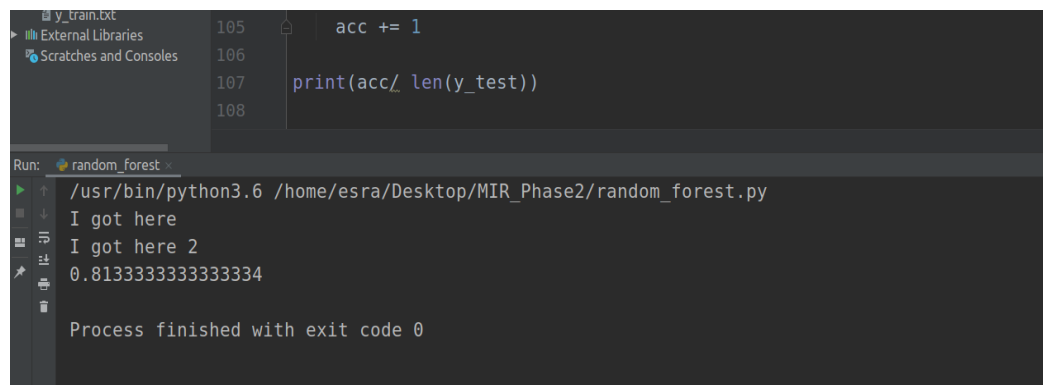
```
106 print(acc/ len(y_test))
107
Run: random_forest x
/usr/bin/python3.6 /home/esra/Desktop/MIR_Phase2/random_forest.py
I got here
I got here 2
0.776
Process finished with exit code 0
```

max_depth = 64: ,N_estimators = 550



```
105 acc += 1
106
Run: random_forest x
/usr/bin/python3.6 /home/esra/Desktop/MIR_Phase2/r
I got here
I got here 2
0.8046666666666666
Process finished with exit code 0
```

max_depth = 100: ,N_estimators = 500



```
105 acc += 1
106
107 print(acc/ len(y_test))
108
Run: random_forest x
/usr/bin/python3.6 /home/esra/Desktop/MIR_Phase2/random_forest.py
I got here
I got here 2
0.8133333333333334
Process finished with exit code 0
```

max_depth = None: ,N_estimators = 750


```
random_forest
/usr/bin/python3.6 /home/esra/Desktop/MIR_Phase2/random_forest.py
I got here
I got here 2
0.8133333333333334

Process finished with exit code 0
```

در کل بهتر است max_depth نگذاریم و n_estimators هم اگر ۵۰۰ باشد کافی است و بیشتر از آن افزایش دقت چندانی به ما نمی‌دهد.

Kmeans: (نمودار اول حاصل از خوشه‌بندی پیاده‌سازی‌شده و نمودار دوم حاصل از خوشه‌بندی Kmeans موجود در sklearn است.)

3

