

# למידת מכונה - תקציר פרויקט + קישור לפרויקט

מגשים:

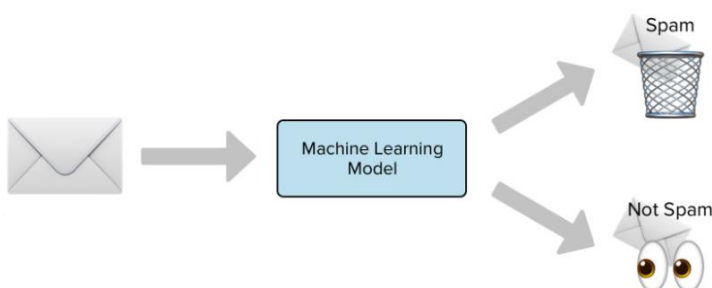
איריס רביץ 320466238

קישור לגיטהאב:

<https://github.com/EvgenTen/SpamDetection>

השאלה שאני מתכוונת לענות עליה:

בהינתן הודעה חדשה, תוכל התוכנית לזהות אותה כהודעת ספאם או הודעה רגילה?



תיאור של המאגר הנבחר:

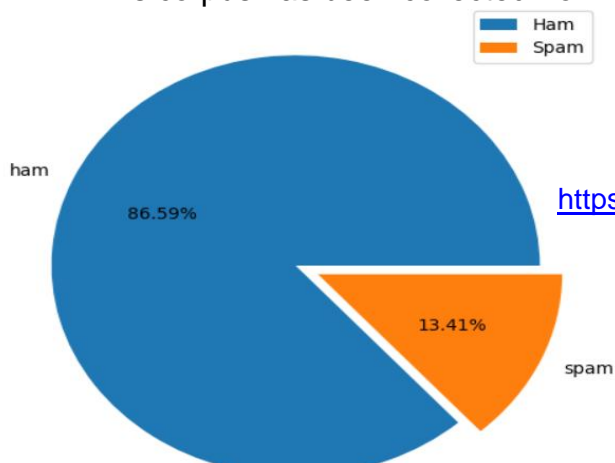
Context:

The SMS Spam Collection is a set of SMS tagged messages that have been collected for SMS Spam research. It contains one set of SMS messages in English of 5,574 messages, tagged according being ham (legitimate) or spam.

Content:

The files contain one message per line. Each line is composed by two columns: v1 contains the label (ham or spam) and v2 contains the raw text.

This corpus has been collected from free or free for research sources on the Internet.



קישור למאגר המידע:

<https://archive.ics.uci.edu/ml/datasets/sms+spam+collection>

## -Preprocess

The first step (and the harder part) was to find a way to convert the `spam.csv` file to data that algorithms understand.

### Data Preprocessing, Feature Vector and Models Used:

1. The Dataset had two columns: 1st containing the class of data ham or spam and 2nd containing a string which is the text message.

2. All English Stopwords were imported from NLTK and were removed if found in the sentences.

3. I used a basic Count vectorizer from Sklearn library in order to tokenize and vectorize the string of text. The Count Vectorizer provides a simple way to both tokenize a collection of text documents and build a vocabulary of known words, but also to encode new documents using that vocabulary.

It can be used as follows:

a) Create an instance of the CountVectorizer class.

b) Call the `fit()` function in order to learn a vocabulary from document.

c) Call the `transform()` function on the document as needed to encode each as a vector.

An encoded vector is returned with a length of the entire vocabulary and an integer count for the number of times each word appeared in the document.

4. I got our numeric or real feature vector from string of text messages.

5. Next perform the test train split in the ratio of 70:30, we select the samples randomly.

6. I feed the `X_train`, `X_test`, `y_train`, `y_test` to different ML models namely, AdaBoost, Decision tree classifier, K-nearest neighbors, Support vector machines, (regular)MultinomialNB

### **The model: MultinomialNB + TfidfTransformer**

```
print(ham)
```

	Class	Text	...	numClass	Count
0	ham	Go until jurong point, crazy.. Available only ...	...	0	111
1	ham	Ok lar... Joking wif u oni...	...	0	29
3	ham	U dun say so early hor... U c already then say...	...	0	49
4	ham	Nah I don't think he goes to usf, he lives aro...	...	0	61
6	ham	Even my brother is not like to speak with me. ...	...	0	77

```
print(spam)
```

	Class	Text	...	numClass	Count
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	...	1	155
5	spam	FreeMsg Hey there darling it's been 3 week's n...	...	1	148
8	spam	WINNER!! As a valued network customer you have...	...	1	158
9	spam	Had your mobile 11 months or more? U R entitle...	...	1	154
11	spam	SIX chances to win CASH! From 100 to 20,000 po...	...	1	136
...	...	...	...	...	...

```
vectorizer.fit_transform(data_read.Text)
```

```
(0, 8030) 1
(0, 4350) 1
(0, 5920) 1
(0, 2327) 1
(0, 1303) 1
(0, 5537) 1
(0, 4087) 1
(0, 1751) 1
(0, 3634) 1
```

...

```
(5570, 4161) 1
(5570, 903) 1
(5570, 1546) 1
(5571, 7756) 1
(5571, 5244) 1
(5571, 4225) 2
(5571, 7885) 1
(5571, 6505) 1
```

```
data_read.numClass
```

```
0      0
1      0
2      1
3      0
4      0
```

...

```
5567    1
5568    0
5569    0
5570    0
5571    0
```

**The model: MultinomialNB + TfidfTransformer:**

### Data Preprocessing, Feature Vector and Models Used:

1. no change
2. no change
3. I added the TFIDF algorithm:

TFIDF, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in searches of information retrieval, text mining, and user modeling. The tf-idf value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general.

```
tfidf_transformer = TfidfTransformer().fit(x)
dummy_transformed = tfidf_transformer.transform(x)
print(dummy_transformed)
```

```
(5570, 1546) 0.3402048888248921
(5570, 1438) 0.1429585509124154
(5570, 1084) 0.11225268140936365
(5570, 903)  0.3247623397615813
(5571, 7885) 0.42752913176432156
(5571, 7756) 0.14849350328973984
(5571, 6505) 0.5565029307246045
(5571, 5244) 0.39009002726386227
(5571, 4225) 0.5773238083586979
```

```
(0, 8489) 0.22080132794235655
(0, 8267) 0.18238655630689804
(0, 8030) 0.22998520738984352
(0, 7645) 0.15566431601878158
(0, 5920) 0.2553151503985779
(0, 5537) 0.15618023117358304
(0, 4476) 0.2757654045621182
(0, 4350) 0.3264252905795869
```

Now, lets check IDF for 'you', the most frequently repeated word in the message against 'hey', a least repeated word:

```
you: 2.2548286210328206
hey: 4.907189916274442
```

As we can see, words with lower frequency are weighed higher than words with higher frequency in the dataset.

4. no change

5. I feed the X\_train, X\_test, y\_train, y\_test to ML model namely MultinomialNB.

### תהליך הלמידה-

תכנות הראשוני הוא להשתמש באלגוריתמי למידת מכונה שלמדנו בשיעור. לחקור את הנתונים בעזרת המודלים:

AdaBoost, Decision tree classifier, K nearest neighbours, Support vector machines, (regular) MultinomialNB

עבודה התחלתית עם הנתונים הייתה קשה, לא ידעתי איך להאמיר טקסט לנתונים שאלגוריתם יוכל להבין. לאחר החיפושים ולימוד הנושא הצלחתי לקודד משפטים לווקטורים ומשם התקדמתי לחקירת נתונים ומציאת תוצאות ביניים. כל המודלים הראו תוצאות טובות. אז בחרתי באחת המודלים MultinomialNB, רציתי לשפר את האלגוריתם ולהגיע לתוצאות יותר מדויקות. לכן הפעלתי אלגוריתם נוסף שנקרא TFIDF. הוא נתון מספרי שנועד לשקף עד כמה מילה חשובה בטקסט. אלגוריתם שיפר את התוצאה באחוז אחד והיו גם הרצות שלא הראו שינוי. המטרה הסופית הייתה לבנות מודל חדש שממין הודעה שנכנסת מבחוץ כמשפט ספאם או כמשפט רגיל. ולענות על השאלה שנשאלה בתחילת המחקר: בהינתן הודעה חדשה, תוכל התוכנית לזהות אותה כהודעת ספאם או הודעה רגילה? נתקעתי בבעיה נוספת, לפני כן למדתי איך לקודד קובץ שלם לווקטורים וכאן העבודה עם משפט בודד ובתוצאה לקבל וקטורים שהמודל יוכל לקרוא. לאחר החיפושים הצלחתי לממש גם את זה. והמודל היה מוכן לקלוט משפט חדש לקידוד. נשאר היה להכניס אותם למודל שמבוסס על אלגוריתמים MultinomialNB ו TFIDF ולקבל תוצאה סופית. האם הודעה החדשה ספאם או לא.

### אלגוריתמים שהשתמשתי בהם:

AdaBoost\*

Decision tree classifier\*

K nearest neighbours\*

Support vector machines\*

(regular) MultinomialNB\*

SGD\*

## אלגוריתמים שהשתמשתי במודל:

MultinomialNB\*

TFIDF\*

## תוצאות -

בכל אלגוריתם חילקתי את הדאטהסט ל-70 אחוז training ו-30 אחוז testing.

באלגוריתמים **KNN** הרצתי כמה גירסאות- אחד של שכן אחד, שלושה, חמישה ושבעה. הרצתי את האלגוריתם כמה הרצות, כל פעם שיניתי את הדאטה של test והtrain, ובדקתי איזה מספר שכנים נותן את התוצאה הטובה ביותר. התוצאות היו שונות לפי מדד הצלחה **Accuracy**, כל הווריאציות נתנו תוצאות בסביבות ה-90~95 אחוזי דיוק, במקום הראשון (תמיד) היה כאשר מספר השכנים הוא אחד, במקום השני היו שלושה שכנים, במקום השלישי חמישה שכנים, ובמקום האחרון שבעה שכנים. לפי המדד **F1 Score**:

That is, a good F1 score means that you have low false positives and low false negatives, so you're correctly identifying real threats and you are not disturbed by false alarms. An F1 score is considered perfect when it's 1, while the model is a total failure when it's 0

התוצאות היו שונות, כל הווריאציות נתנו תוצאות בסביבות ה-0.43~0.75 אחוזי דיוק, (המודל מושלם כאשר מקבלים 1 מוחלט וכישלון מוחלט כאשר 0) במקום הראשון השכן הראשון עם דיוק הכי קרוב ל-1. במקום השני היו שלושה שכנים, במקום השלישי חמישה שכנים, ובמקום האחרון שבעה שכנים עם דיוק הכי קרוב ל-0. אלגוריתם מראה דיוק הכי גבוה בהרצת גירסאה של שכן אחד. תוצאות:

(מתוך הקובץ knn.py)

```
KNN1 :  
Accuracy in %:  
94.67703349282297
```

```
F1 Score:  
0.7588075880758809
```

```
KNN3 :  
Accuracy in %:  
92.16507177033493
```

```
F1 Score:  
0.5969230769230769
```

```
KNN5 :  
Accuracy in %:  
91.02870813397129  
  
F1 Score:  
0.5098039215686275
```

```
KNN7 :  
Accuracy in %:  
90.19138755980862  
  
F1 Score:  
0.4383561643835616
```

האלגוריתם **SVM** תוצאה: האלגוריתם מראה דיוק גבוה בשני מדדים.

```
SVM:  
  
Accuracy in %:  
98.20574162679426  
  
F1 Score:  
0.9308755760368664
```

(מתוך הקובץ svm.py).

האלגוריתם **Adaboost** תוצאה: האלגוריתם מראה דיוק גבוה בשני מדדים.

```
Adaboost:  
  
Accuracy in %:  
98.08612440191388  
  
F1 Score:  
0.9130434782608695
```

(מתוך הקובץ adaboost.py).

האלגוריתם **DecisionTreeClassifier**: תוצאה: האלגוריתם מראה דיוק גבוה בשני מדדים.

```
DecisionTreeClassifier:
```

```
Accuracy in %:
```

```
96.88995215311004
```

```
F1 Score:
```

```
0.8864628820960699
```

(מתוך הקובץ decisionTree.py).

האלגוריתם **MultinomialNB**: (רגיל) תוצאה: האלגוריתם מראה דיוק גבוה בשני מדדים.

```
regular_MultinomialNB:
```

```
Accuracy in %:
```

```
97.54784688995215
```

```
F1 Score:
```

```
0.9154639175257732
```

(מתוך הקובץ regularmultinomialNB.py).

האלגוריתם **SGD**: תוצאה: האלגוריתם מראה דיוק גבוה בשני מדדים.  
(מתוך הקובץ sgd.py).

```
SGD:
```

```
Accuracy in %:
```

```
98.44497607655502
```

```
F1 Score:
```

```
0.9304812834224598
```



האלגוריתם **TFIDF + MultinomialNB** תוצאה: האלגוריתם מראה דיוק גבוה בשני מדדים. בהשוואה לאלגוריתם הרגיל יש שיפור קטן. (מתוך הקובץ model.py).

```
Multi-NB:

Accuracy in %:
98.74401913875597

F1 Score:
0.952808988764045
```

### השוואה בין כל האלגוריתמים:

כמו כן עשיתי השוואה בין כל האלגוריתמים השונים, כדי לראות איזה אלגוריתם הוא המדויק ביותר.

במקום הראשון עם ממוצע גבוה של בערך 98.74 אחוזים - **TFIDF + MultinomialNB**.  
במקום השני עם ממוצע גבוה של בערך 98.44 אחוזים - **SGD**.  
במקום השלישי עם ממוצע לא רחוק ממנו של בערך 98.08-98.2 אחוזים - **Adaboost** ו **SVM**.  
במקום הרביעי והמכובד - **MultinomialNB**, עם ממוצע של בין 97.54 אחוז דיוק.  
ובמקום האחרון והחביב - **DecisionTreeClassifier** עם ממוצע בין 96.88 אחוזי דיוק.  
(האחוזים יכולים להשתנות מריצה לריצה בחצי אחוז לכל היותר, אבל לרוב זאת החלוקה)

## טסט על הודעה חדשה-

רציתי לבחון עד כמה המודל שלי טוב גם על הודעה נכנסת חדשה.  
טבלאות של מילים הכי נפוצים במאגר המחולקים לפי סוג:

Top 10 Spam words are :

call	346
free	217
txt	156
ur	144
u	144
mobile	123
text	121
stop	114
claim	113
reply	104

Top 10 Ham words are :

u	974
gt	318
lt	316
get	301
go	246
ok	246
got	242
ur	237
know	234
like	231

טבלאות של מילים הכי נפוצים במאגר המחולקים לפי סוג:

Top 10 Spam words are :

call	346
free	217
txt	156
ur	144
u	144
mobile	123
text	121
stop	114
claim	113
reply	104

Top 10 Ham words are :

u	974
gt	318
lt	316
get	301
go	246
ok	246
got	242
ur	237
know	234
like	231

השתמשתי בטבלאות כדי לראות עד כמה המודל עובד נכון. למשפט החדש הכנסתי מילה מתוך הטבלאות וזה מה שקיבלתי: (מתוך הקובץ model.py).

1SMS- הודעת ספאם ו-2SMS- הודעה רגילה  
גם אתם יכולים לנסות את המודל ולהכניס הודעה חדשה בקונסול וללחוץ enter.

למשל:

```
please write a new sentence using words from the top spam words or regular words:  
stop free  
SMS1 is spam .. SMS2 is ham .. new sentence is spam
```

הכנסתי הודעה חדשה: קיבלתי שהיא ספאם. המודל עובד עם דיוק של 98 אחוז שזה די טוב.

```
Testing specific messages:
```

```
SMS1 = '[URGENT!] Your Mobile No 398174814449 was awarded a vacation'
```

```
SMS2 = 'Hello my friend, how are you?'
```

```
SMS1 is spam .. SMS2 is ham
```