

Togyer Zoltán – JT0PU6

Haladó Programozás beadandó

2025 – Haladó Programozás

Tartalomjegyzék

| | |
|---------------------------------------|---|
| Tartalomjegyzék..... | 2 |
| Bevezetés..... | 2 |
| Github..... | 3 |
| Technikai Információk..... | 3 |
| Adatforrás | 4 |
| Szöveg előfeldolgozása | 4 |
| Címkék számmá alakítása | 4 |
| Adatok felosztása..... | 5 |
| Vektorizálás / TF-IDF | 5 |
| A modell tanítása (Naive Bayes) | 5 |
| A modell értékelési módszere | 5 |
| Konfúziós mátrix..... | 6 |
| Saját Üzenet..... | 7 |
| SPAM mentés..... | 7 |
| Összegzés..... | 8 |
| AI használat | 8 |

Bevezetés

A projekt célja egy olyan gépi tanulási modell készítése volt, amely képes automatikusan felismerni a spam (kéretlen) üzeneteket. Fontos megjegyezni, hogy a kód jelenleg csak angol nyelvű üzeneteket és e-maileket képes feldolgozni, mivel a modell angol nyelvű spamüzeneteken lett betanítva.

Ez egy klasszikus szövegfeldolgozási (Natural Language Processing, NLP) probléma, ahol az algoritmusnak meg kell tanulnia a különbséget az átlagos, hétköznapi üzenetek és a kéretlen, reklám- vagy csaló jellegű üzenetek között.

A spam felismerése valós alkalmazásokban is rendkívül fontos, például az e-mail szolgáltatók (Gmail, Outlook stb.) is használnak hasonló modelleket. Ebben a projektben egy egyszerű, de hatékony megoldást valósítottam meg a **Naive Bayes osztályozó** segítségével, amely jól működik szöveges adatok feldolgozásánál.

2025 – Haladó Programozás

Github

https://github.com/tzoli07/Spam_Detector_HaladoP.git

Technikai Információk

A projekt **Python nyelven** készült, mivel ez a legelterjedtebb nyelv gépi tanulás és adatfeldolgozás területén. Illetve a gyakorlati órákon is ezt használjuk.

A fejlesztő környezet a **Visual Studio Code** volt.

Az alábbi Python könyvtárakat kellett használni az eredményes munkához:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import string
import nltk
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

Az alábbiakban taglalom, hogy melyik miért szükséges:

- **pandas** – táblázatos adatok kezelésére, azaz az üzenetek betöltésére és előfeldolgozására szolgál.
- **numpy** – matematikai műveletekhez, tömbök kezeléséhez hasznos, de itt inkább háttérben segít.
- **matplotlib.pyplot** – az eredmények vizuális megjelenítéséhez, például a konfúziós mátrix kirajzolásához.
- **string** – a szövegből az írásjelek eltávolításához használtam.
- **nltk (Natural Language Toolkit)** – természetes nyelvfeldolgozó könyvtár, innen használtam a stopwords listát, hogy kiszűrjem a felesleges szavakat („and”, „is”, „the”, stb.).
- **sklearn.model_selection.train_test_split** – az adathalmazt tanító és teszt részre osztja.
- **sklearn.feature_extraction.text.TfidfVectorizer** – a szöveget numerikus vektorrá alakítja, hogy a modell tudja értelmezni.
- **sklearn.naive_bayes.MultinomialNB** – maga a Naive Bayes osztályozó algoritmus, amivel a modellt tanítom.

2025 – Haladó Programozás

- **sklearn.metrics** – a modell teljesítményének méréséhez (pontosság, riport, mátrix).

Adatforrás

A projekt egy nyilvánosan elérhető adatforrást használ, amely SMS-üzeneteket tartalmaz, és mindegyikhez meg van adva, hogy **spam** vagy **nem spam (ham)**,

```
url = "https://raw.githubusercontent.com/justmarkham/pycon-2016-tutorial/master/data/sms.tsv"
df = pd.read_csv(url, sep='\t', header=None, names=['label', 'message'])
```

Ezzel körülbelül **5500 soros** adatbázist töltök be, ahol:

- „**label**” oszlop: tartalmazza az osztályozást („ham” = nem spam, „spam” = spam)
- „**message**” oszlop: maga az üzenetnek a szövege

Szöveg előfeldolgozása

A gépi tanulásos modellek csak **számokkal tudnak dolgozni**, ezért előbb a nyers szöveget elő kell készíteni. Ehhez **egy clean_text()** függvényt írtam:

```
def clean_text(text):
    text = text.lower()
    text = ''.join([c for c in text if c not in string.punctuation])
    words = text.split()
    words = [w for w in words if w not in stopwords.words('english')]
    return ' '.join(words)
```

A függvény az alábbi feladatokat hatja végbe:

- **Case sensitive:** a modell ne különböztesse meg a „Free” és „free” szót.
- **Írásjelek eltávolítása:** pontok, felkiáltójelek stb. nem hordoznak jelentést.
- **Stopwordök eltávolítása:** ezek gyakori, de értelmetlen szavak („the”, „and”, „in”).
- **Tisztított szöveg visszaadása:** így a modell csak a lényeges szavakat tanulja meg.

A megtisztított üzeneteket elmentődnek egy új oszlopba:

```
df['cleaned'] = df['message'].apply(clean_text)
```

Címkék számmá alakítása

A modell numerikus bemenetet vár, ezért a szöveges címkéket (spam/ham) 0 és 1 értékekre kell alakítani, így a 0 = nem spam, 1 = spam:

```
df['label_num'] = df['label'].map({'ham': 0, 'spam': 1})
```

Adatok felosztása

A teljes adatbázist **tanító** és **teszt** halmazra kell osztani. Ez azért kell, hogy a modell olyan adatokkal is ki legyen próbálva, amelyeket **nem látott tanítás közben** így lehet tesztelni a későbbiekben, hogy valóban működik. Ezt a kód részlet azt jelenti, hogy **80%** adat szükséges a tanításhoz és **20%** adat a teszteléshez. A „`random_state=42`” biztosítja, hogy minden ugyanaz az eredmény szülessen (reprodukálhatóság).

```
X_train, X_test, y_train, y_test = train_test_split(  
    df['cleaned'], df['label_num'], test_size=0.2, random_state=42  
)
```

Vektorizálás / TF-IDF

A gépi tanulási modell nem tud közvetlenül szöveggel dolgozni, ezért **a szöveget számokká kell alakítani**. Ezt a TF-IDF módszerrel kell megtenni:

```
vectorizer = TfidfVectorizer(max_features=3000)  
X_train_vec = vectorizer.fit_transform(X_train)  
X_test_vec = vectorizer.transform(X_test)
```

- **TF (Term Frequency)**: egy szó hányszor szerepel az adott üzenetben.
- **IDF (Inverse Document Frequency)**: mennyire ritka a szó a teljes adatbázisban. Így a ritka, de fontos szavak (pl. „winner”, „free”, „offer”) nagyobb súlyt kapnak.

A „`max_features=3000`” azt jelenti, hogy legfeljebb 3000 legfontosabb szó kerül be a modellbe.

A modell tanítása (Naive Bayes)

A tanításhoz a **Multinomial Naive Bayes** modellt használtam. Ez egy gyors és hatékony modell, különösen szövegalapú feladatokhoz. A modell „megtanulja”, hogy mely szavak jellemzőek a spam üzenetekre (pl. free, click, win) és melyek a normál üzenetekre (pl. meeting, call, tomorrow).

```
model = MultinomialNB()  
model.fit(X_train_vec, y_train)
```

A modell értékelési módszere

A tesztadatokat a modell ki értékeli utána pedig megjelenik, hogy mennyire volt pontos. A klasszifikációs riport részletesen mutatja:

- **Precision (pontosság)**: mennyire pontos, ha spam-et jelöl.
- **Recall (érzékenység)**: mennyit talált meg a tényleges spamek közül.
- **F1-score**: a kettő kombinált mérőszáma.

A modell 97–98%-os pontossággal dolgozik ami nagyon jó eredmény.

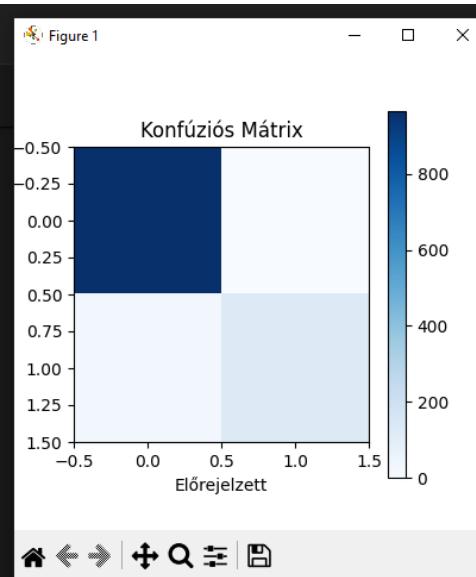
2025 – Haladó Programozás

Futás közben az alábbi kimenetet kapjuk, ahol megjelenik a Konfúziós mátrix is, mint grafikus elem.

```
45
46 # --- 6. Modell tanítása ---
47 print("Modell tanítása...")
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\spam-detector> run .\spam_detector.py
o PS D:\spam-detector> app .\spam_detector.py^C
o PS D:\spam-detector> python spam_detector.py
[nltk_data] Downloading package stopwords to
[nltk_data]   C:\Users\x\AppData\Roaming\nltk_data...
[nltk_data] Package stopwords is already up-to-date!
Adatok betöltése...
Szövegek tisztítása...
Modell tanítása...

Modell pontossága: 97.85%

Klasszifikációs riport:
precision    recall  f1-score   support
          0       0.98     1.00     0.99      966
          1       1.00     0.84     0.91      149
   accuracy                           0.98      1115
  macro avg       0.99     0.92     0.95      1115
weighted avg       0.98     0.98     0.98      1115
```

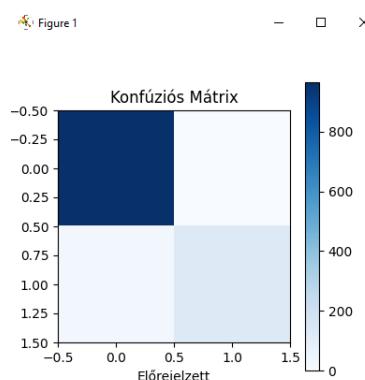


Konfúziós mátrix

A mátrix vizuálisan mutatja, hol hibázott a modell, Minél sötétebb a bal felső és jobb alsó mező, annál jobb a modell.

```
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(4,4))
plt.imshow(cm, cmap='Blues')
plt.title('Konfúziós Mátrix')
plt.xlabel('Előrejelzett')
plt.ylabel('Valódi')
plt.colorbar()
plt.show()
```

| Valós / Előrejelzett | Nem Spam | Spam |
|----------------------|------------------------|--------------------|
| Nem Spam | Helyesen nem spam (TN) | Tévesen spam (FP) |
| Spam | Tévesen nem spam (FN) | Helyesen spam (TP) |



2025 – Haladó Programozás

Saját Üzenet

A program végén interaktív rész található, ahol a felhasználó saját üzenetet írhat be.

A modell azonnal visszajelzést ad.

SPAM mentés

A továbbfejlesztett verzióban bevezetésre került egy **spam archiváló funkció**, amely a spamként azonosított e-maileket és üzeneteket automatikusan elmenti egy külön mappába.

Működés lépései:

1. Többsoros bemenet támogatása:

A felhasználó akár egy teljes email szövegét is beírhatja, ami több soros, vagy fájlból is van lehetőség az üzenet betöltésére.

2. Spam mappa létrehozás:

A kód ellenőrzi, hogy létezik-e „spam_emails” mappa, ha nem akkor a kód létrehozza:

```
spam_folder = "spam_emails"
if not os.path.exists(spam_folder):
    os.makedirs(spam_folder)
```

3. Spam mentése fájlba

Ha a modell azonosít, a szöveg automatikusan elmentésre kerül. A kimenet jelzi az eredményt és a fájl mentési helyét.

```
if pred == 1:
    print("Ez az e-mail: SPAM")
    timestamp = datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
    safe_name = f"spam_{timestamp}.txt"
    filepath = os.path.join(spam_folder, safe_name)
    with open(filepath, "w", encoding="utf-8") as f:
        f.write(f"Detected: {datetime.now().isoformat()}\n\n")
        f.write("Original email:\n")
        f.write(email_text + "\n\n")
        f.write("Cleaned text:\n")
        f.write(cleaned + "\n")
    print(f" A spam e-mail elmentve ide: {filepath}")
else:
    print("Ez az e-mail: NEM SPAM")
```

4. Tesztelés spam emailel

```
-- Saját e-mail tesztelése --
Használat:
- Fájl beolvasása: file:<fájlnév> (pl. file:email1.txt)
- Többsoros beillesztés / gépelés: paste vagy gépelд be a levelet, majd új sorban írj egyetlen pontot: . és ENTER-rel fejezd be
- Kilépés: exit

Adj meg 'file:<fájlnév>' vagy kezd el beírni/illesszed a levelet (vagy 'exit'): From: promotions@freeprizes.example.com
Ha szeretnéd folytatni a levelet, folytasd a gépelést; ha vége: írj egy sort, ami csak '.' (pont) és ENTER.
Subject: §§ CONGRATULATIONS! You have won a FREE iPhone! §§

Click here to claim your prize:
http://free-prize.example.com/claim?id=ABC123
.

Ez az e-mail: SPAM
A spam e-mail elmentve ide: spam_emails\spam_2025-10-23_17-18-17.txt
```

2025 – Haladó Programozás

Összegzés

A projekt sikeresen megvalósított egy **gépi tanulásos spam detektort**, amely közel **98%-os pontossággal** képes megkülönböztetni a kéretlen és a normál üzeneteket.

Fő lépések:

- Szöveg előfeldolgozás (kisbetűsítés, stopword-eltávolítás)
- TF-IDF vektorizálás
- Naive Bayes osztályozás
- Teljesítményelemzés (pontosság, riport, konfúziós mátrix)
- Spam archiválás fájlba

A modell gyors, hatékony, és jól bemutatható oktatási környezetben. A kód bővíthető, például grafikus felüallettel (Tkinter) vagy webes alkalmazással (Streamlit).

AI használat

A Python kód egyes részeinek kidolgozásához, a ChatGPT (OpenAI GPT-5 modell) nyújtott segítséget.