Assignment NO.3 Solutions

Data Mining | Fall 1401 | Dr.Rahmani

Student name : **Amin Fathi**

Student id : **400722102**

ابتدا پکیج‌های لازم را import می کنیم.

```
import pandas as pd
import numpy as np
import json
import nltk
import re
import csv
import matplotlib.pyplot as plt
import seaborn as sns
from tqdm import tqdm
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn import metrics

%matplotlib inline
pd.set_option('display.max_colwidth', 300)
```

Dataset دریافتی شـامل یک فایل json و یک فایل csv اسـت. در فایل json، جفت‌های خصـوصـیت-کلید داریم. ابتدا فایل json را به csv تبدیل می کنیم. اینگونه تبدیل آن به DataFrame راحت تر است. همچنین عناصر آن به خوبی جدا می شود.

```
df = pd.read_json(r'/content/drive/MyDrive/University/DM/HW3/movie_synopsis.json')
df.to_csv('/content/drive/MyDrive/University/DM/HW3/movie_synopsis.csv')
```

پس از تبدیل به csv، هر دو فایل مجموعه داده را لود و به DataFrame تبدیل می کنیم.

```
movie_synopsis = pd.read_csv(r"/content/drive/MyDrive/University/DM/HW3/movie_synopsis.csv")
movie_info = pd.read_csv(r"/content/drive/MyDrive/University/DM/HW3/movie_info.csv")
```

در ادامه جدول حاصل از هر دو مجموعه داده ی دریافتی نشان داده می شود.

movie_synopsis

	_key	imdbID	plot_synopsis	local_id	plot_source
0	1	tt0114709	A boy called Andy Davis (voice: John Morris) uses his toys to act out a bank robbery. The bank is a cardboard box, the robber is Mr. Potato Head (voice: Don Rickles) assisted by Slinky Dog (voice: Jim Varney), and the bystanders include Bo Peep (voice: Annie Potts) and her sheep. The day is save...	1	imdb
1	2	tt0113189	The story opens in 1986, in the Cold War Soviet Union. British secret agent James Bond (Pierce Brosnan) and his fellow 00 agent, Alec Trevelyan (Sean Bean), have infiltrated a secret Soviet chemical weapons production facility with the intention of destroying it. After finding their target, a la...	2	imdb
2	3	tt0113101	The film begins with Ted the Bellhop (Tim Roth) in a room filled with hotel memorabilia, talking to Sam the Bellhop (Marc Lawrence) about what a bellhop is. It's New Year's Eve."The Missing Ingredient" begins with a bunch of women (including Madonna and Lili Taylor) converging in the Honeymoon S...	3	imdb
3	4	tt0113161	Chilli Palmer (John Travolta) is a loan shark living in Miami, Florida. While sitting in a restaurant on an unusually cold day, he's approached by the cantankerous and egotistical Ray 'Bones' Barboni (Dennis Farina). Bones mocks Chilli's name in reference to the cold weather and leaves with his ...	4	imdb
4	5	tt0112722	After giving a guest lecture on criminal psychology at a local university, Dr. Helen Hudson, a respected field expert on serial killers, is cornered in the lecture hall's restroom by one of her previous subjects, Daryll Lee Cullum, who has escaped from prison. He kills a police officer and bruta...	5	imdb
...	...	...	...	...	...
1676	1678	NaN	The film opens on two human forms, which soon reveal themselves to be that of a young man and a frail old woman. They recline in a silence broken only by whispers and indistinguishable noises. The young man is the son (Alexei Ananishnov) who is taking care of his exhausted sick mother (Gudrun Ge...	1678	wiki
1677	1679	tt0120594	Alan (Jared Harris) is a schoolteacher in London who also moonlights as a jazz disc jockey for a hospital PA system. One night after work, he goes to a bar and sees Beatrice (Asia Argento) a beautiful woman who is arguing with two men. Alan is immediately captivated by Beatrice and begins to pu...	1679	wiki
1678	1680	tt0120148	The film follows Helen Quilley (Gwyneth Paltrow), a young Englishwoman living in London who has just been fired from her public relations job. The plot splits into two parallel universes, based on the two paths her life could take depending on whether she catches a London Underground train or no...	1680	imdb
1679	1681	tt0111804	NaN	1681	wiki
1680	1682	tt0102855	NaN	1682	wiki

1681 rows × 5 columns

```
movie_info
```

	locale_id	title	id_imdb	genre_imdb
0	1	Toy Story (1995)	tt0114709	Animation|Adventure|Comedy|Family|Fantasy
1	2	GoldenEye (1995)	tt0113189	Action|Adventure|Thriller
2	3	Four Rooms (1995)	tt0113101	Comedy
3	4	Get Shorty (1995)	tt0113161	Comedy|Crime|Thriller
4	5	Copycat (1995)	tt0112722	Drama|Mystery|Thriller
...	...	...	...	...
1677	1678	Mat' i syn (1997)	NaN	NaN
1678	1679	B. Monkey (1998)	tt0120594	Crime|Drama|Romance|Thriller
1679	1680	Sliding Doors (1998)	tt0120148	Comedy|Drama|Fantasy|Romance
1680	1681	You So Crazy (1994)	tt0111804	Documentary|Comedy
1681	1682	Scream of Stone (Schrei aus Stein) (1991)	tt0102855	Drama

1682 rows × 4 columns

مجموعه داده‌ی movie_synopsis شامل یک خلاصه از فیلم در ستون plot_synopsis و همچنین منبع آن در ستون plot_source است. برای هر فیلم یک id در ستون imdbID در نظر گرفته شده است. در مجموعه داده‌ی movie_info نیز نام فیلم، ژانر، id و locale_id به عنوان ستون‌ها قرار دارد.

در ادامه ستون local_id را برای هر دو مجموعه هم نام کرده تا بتوانیم دو مجموعه داده را روی این ستون merge کنیم. پس از merge نیز ستون‌های تکراری و اضافی را حذف می کنیم.

```
movie_synopsis = movie_synopsis.rename(columns={'local_id':'locale_id'})
movies = pd.merge(movie_synopsis, movie_info[['locale_id','title', 'id_imdb', 'genre_imdb']], on = 'locale_id')
movies = movies.drop(['id_imdb', '_key', 'plot_source', 'imdbID'], axis=1)
```

پس از merge کردن دو مجموعه داده و تبدیل آن ها به یک DataFrame به نام movies، مراحل دیگر پیش پردازش را انجام می دهیم. یکی از مهم ترین مراحل حذف missing valueها است. با توجه به بزرگ بودن مجموعه داده، نمونه هایی که شامل missing value می شوند را حذف می کنیم.

```
movies.dropna(axis = 0 ,inplace = True)
movies = movies.reset_index()
```

اگر به movies نگاه شود، متوجه می شویم که ستون ژانرها نیاز به یک پیش پردازش داشته و باید عناصر آن را جدا کرده و به لیست تبدیل کنیم. اینگونه انجام پردازش بر روی آن ها ساده تر و بهتر خواهد بود. برای جدا کردن عناصری که با | جدا شده اند، از تابع split استفاده می کنیم. این تابع عناصر را بر حسب | جدا کرده و به لیست تبدیل می کند.

```
genres = []
for i in movies['genre_imdb']:
  genres.append(i.split("|"))
movies['genre_imdb'] = genres
```

movies در انتها به صورت زیر حاصل می شود.

```
movies
```

	index	plot_synopsis	locale_id	title	genre_imdb
0	0	A boy called Andy Davis (voice: John Morris) uses his toys to act out a bank robbery. The bank is a cardboard box, the robber is Mr. Potato Head (voice: Don Rickles) assisted by Slinky Dog (voice: Jim Varney), and the bystanders include Bo Peep (voice: Annie Potts) and her sheep. The day is save...	1	Toy Story (1995)	[Animation, Adventure, Comedy, Family, Fantasy]
1	1	The story opens in 1986, in the Cold War Soviet Union. British secret agent James Bond (Pierce Brosnan) and his fellow 00 agent, Alec Trevelyan (Sean Bean), have infiltrated a secret Soviet chemical weapons production facility with the intention of destroying it. After finding their target, a la...	2	GoldenEye (1995)	[Action, Adventure, Thriller]
2	2	The film begins with Ted the Bellhop (Tim Roth) in a room filled with hotel memorabilia, talking to Sam the Bellhop (Marc Lawrence) about what a bellhop is. It's New Year's Eve."The Missing Ingredient" begins with a bunch of women (including Madonna and Lili Taylor) converging in the Honeymoon S...	3	Four Rooms (1995)	[Comedy]
3	3	Chilli Palmer (John Travolta) is a loan shark living in Miami, Florida. While sitting in a restaurant on an unusually cold day, he's approached by the cantankerous and egotistical Ray 'Bones' Barboni (Dennis Farina). Bones mocks Chilli's name in reference to the cold weather and leaves with his ...	4	Get Shorty (1995)	[Comedy, Crime, Thriller]
4	4	After giving a guest lecture on criminal psychology at a local university, Dr. Helen Hudson, a respected field expert on serial killers, is cornered in the lecture hall's restroom by one of her previous subjects, Daryll Lee Cullum, who has escaped from prison. He kills a police officer and bruta...	5	Copycat (1995)	[Drama, Mystery, Thriller]
...	...	...	...	...	...
1404	1671	There's this detective dude who's hired to follow some guy's wife because she keeps getting herself into trouble. She suffers from severe headaches and loss of memory. She never seems to know what's going on. it turns out she's got a split personality. This split personality is the one getting h...	1673	Mirage (1995)	[Action, Thriller]
1405	1674	Jeremy Collier is a returning Vietnam War hero whose experiences leave him unable to adjust to the quiet realities of small-town life. Bob Collier, Jeremy's father, expects his son to go back to his life as it was, without understanding the problems of PTSD. Jeremy's mother, Maurine, treats him ...	1676	War at Home, The (1996)	[Drama]
1406	1675	Angel celebrates the birth of his daughter by taking his first hit of crack cocaine. With the hesitant support of his wife, Monika, he joins a friend of his to deal drugs for a short time—enough time to get out of debt and buy some nice things for the family. Three years later, Angel is still de...	1677	Sweet Nothing (1995)	[Drama]
1407	1677	Alan (Jared Harris) is a schoolteacher in London who also moonlights as a jazz disc jockey for a hospital PA system. One night after work, he goes to a bar and sees Beatrice (Asia Argento) a beautiful woman who is arguing with two men. Alan is immediately captivated by Beatrice and begins to pu...	1679	B. Monkey (1998)	[Crime, Drama, Romance, Thriller]
1408	1678	The film follows Helen Quilley (Gwyneth Paltrow), a young Englishwoman living in London who has just been fired from her public relations job. The plot splits into two parallel universes, based on the two paths her life could take depending on whether she catches a London Underground train or no...	1680	Sliding Doors (1998)	[Comedy, Drama, Fantasy, Romance]

1409 rows × 5 columns

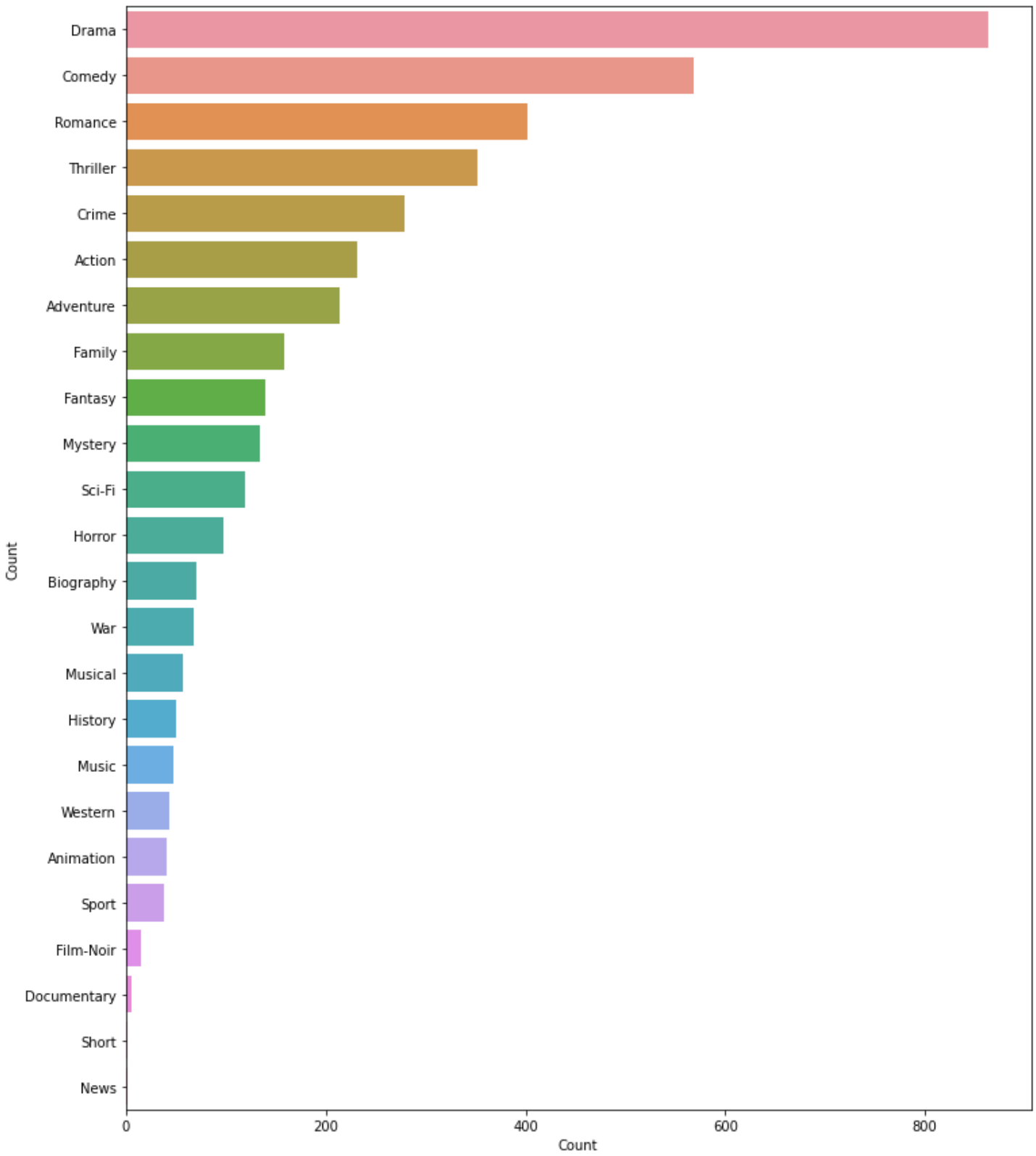
در ادامه تعداد ژانرها را به دســت آورده و از تابع nltk برای ایجاد فرهنگ لغت ژانرها و تعداد آن‌ها در مجموعه داده اســتفاده می شود.

```
# get all genre tags in a list
all_genres = sum(genres,[])
all_genres = nltk.FreqDist(all_genres)
all_genres_df = pd.DataFrame({'Genre': list(all_genres.keys()), 'Count': list(all_genres.values())})
```

سپس نموداری از توزیع فراوانی ژانرها و مقایسه ی فراوانی آن ها در فیلم های مختلف رسم می شود.

```
g = all_genres_df.nlargest(columns="Count", n = 50)
plt.figure(figsize=(12,15))
ax = sns.barplot(data=g, x= "Count", y = "Genre")
ax.set(ylabel = 'Count')
plt.show()
```

طبق نمودار زیر ژانر Dram پرتکرارترین ژانر میان بقیه ی ژانرها اســت. بعد از آن بقیه ی ژانرها نیز بر حســب پرتکرار به پایین لیست شده است.



تابع clean_text به پاکسازی متن در چند مرحله می پردازد. مثلاً حذف علائم نگارشی یا تبدیل تمام حروف به حروف کوچک و ... .

```
# function for text cleaning
def clean_text(text):
    # remove backslash-apostrophe
    text = re.sub("\'", "", text)
    # remove everything alphabets
    text = re.sub("[^a-zA-Z]"," ",text)
    # remove whitespaces
    text = ' '.join(text.split())
    # convert text to lowercase
    text = text.lower()

    return text
```

این تابع بر روی ستون plot_synopsis اعمال می شود. همچنین نتیجه ی اعمال این تابع برای سه نمونه نشان داده شده است. همانطور که مشاهده می شود، علائم نگارشی حذف شده و حروف بزرگ به کوچک تبدیل شده و کل متن با حرف کوچک است.

```
movies['clean_plot'] = movies['plot_synopsis'].apply(lambda x: clean_text(x))
movies[['plot_synopsis', 'clean_plot']].sample(3)
```

	plot_synopsis	clean_plot
1173	Delta of Venus is the sultry adventure of Elena Martin (Audie England), a young female American writer in Paris during the dawn of WWII. All of the stories Elena writes involves her as the star of her own erotic adventures which are secretly contracted by her lover, Lawrence Walters (Costas Mand...	delta of venus is the sultry adventure of elena martin audie england a young female american writer in paris during the dawn of wwii all of the stories elena writes involves her as the star of her own erotic adventures which are secretly contracted by her lover lawrence walters costas mandylor f...
473	In 1953, 10-year-old Larry Flynt is selling moonshine in Kentucky. Twenty years later, Flynt (Woody Harrelson) and his younger brother, Jimmy (Brett Harrelson) run the Hustler Go-Go club in Cincinnati. With profits down, Flynt decides to publish a newsletter for the club, the first Hustler magaz...	in year old larry flynt is selling moonshine in kentucky twenty years later flynt woody harrelson and his younger brother jimmy brett harrelson run the hustler go go club in cincinnati with profits down flynt decides to publish a newsletter for the club the first hustler magazine with nude pictu...
102	The Truth About Cats and Dogs is the story of two youngish women with good hearts and insecurity, who live in the same building. One is an intelligent brunette veterinarian of average looks, with her own radio show about caring for pets; the other is a blonde beginner fashion model of modest int...	the truth about cats and dogs is the story of two youngish women with good hearts and insecurity who live in the same building one is an intelligent brunette veterinarian of average looks with her own radio show about caring for pets the other is a blonde beginner fashion model of modest intelle...

از تابع دیگری به نام freq_word نیز استفاده می شود که تعداد تکرار کلمات را مشخص می کند.

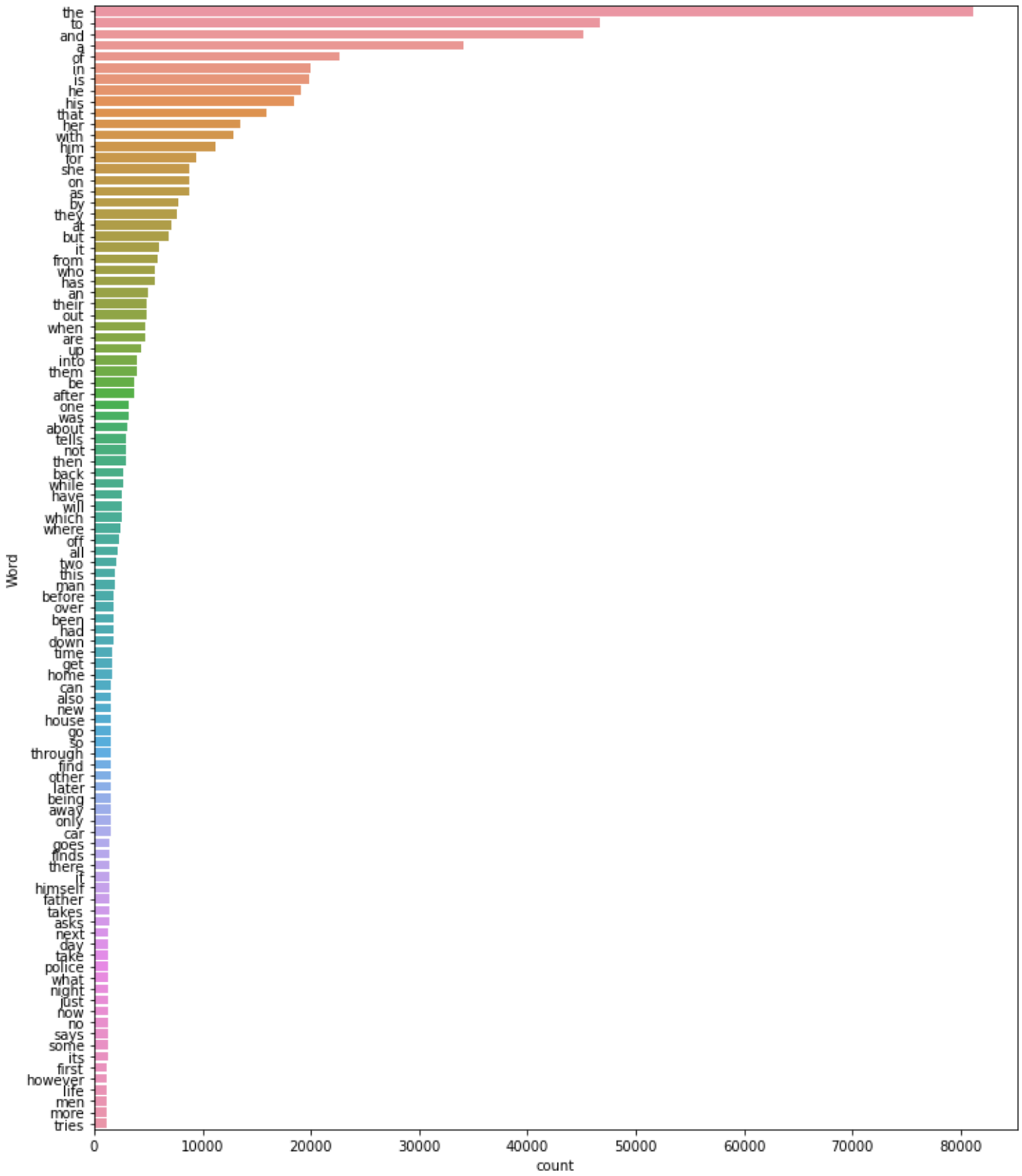
```
def freq_words(x, terms = 30):
  all_words = ' '.join([text for text in x])
  all_words = all_words.split()

  fdist = nltk.FreqDist(all_words)
  words_df = pd.DataFrame({'word':list(fdist.keys()), 'count':list(fdist.values())})

  # selecting top 20 most frequent words
  d = words_df.nlargest(columns="count", n = terms)
  plt.figure(figsize=(12,15))
  ax = sns.barplot(data=d, x= "count", y = "word")
  ax.set(ylabel = 'Word')
  plt.show()
```

نمودار زیر با استفاده از تابع freq_xord پرتکرارترین کلمات را نشان می دهد.

```
# print 100 most frequent words
freq_words(movies['clean_plot'], 100)
```



پرتکرارترین کلمات، stop wordها هستند. این کلمات در مقایسه با سایر کلمات درون متن معنای کمتری دارند. به همین دلیل بهتر است آن ها را از درون متن حذف کنیم. برای این کار ابتدا لیست stop wordها را با استفاده از کتابخانه nltk دانلود می کنیم.

```
nltk.download('stopwords')
```

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

سپس stop word را از داخل متن با استفاده از تابع remove_stopwords حذف می کنیم.

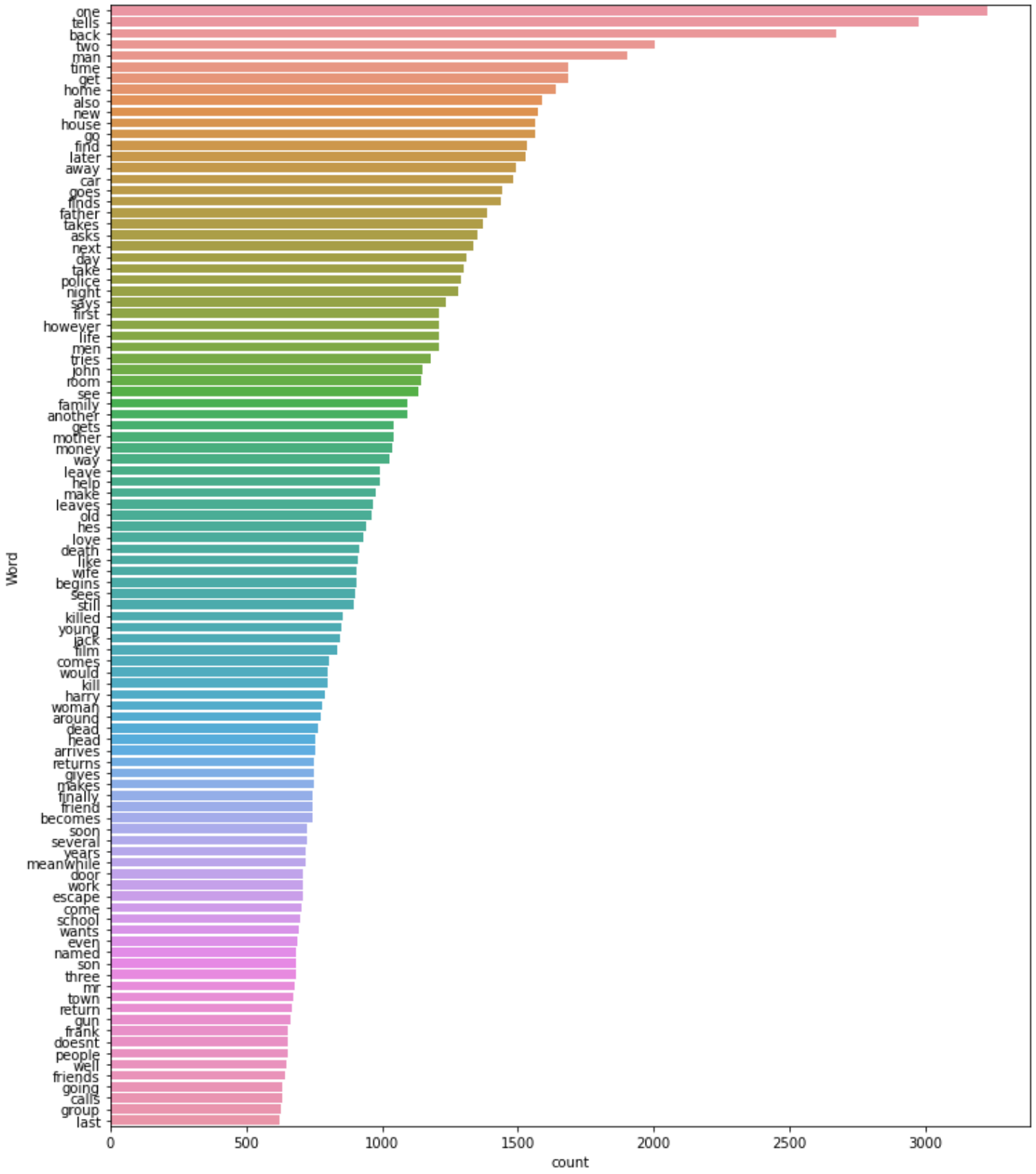
```
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))

# function to remove stopwords
def remove_stopwords(text):
    no_stopword_text = [w for w in text.split() if not w in stop_words]
    return ' '.join(no_stopword_text)

movies['clean_plot'] = movies['clean_plot'].apply(lambda x: remove_stopwords(x))
```

پس از حذف stop wordها، مجدداً کلمات پرتکرار را در نمودار نشان می دهیم. مشاهده می شود که این سری کلمات مهم تری در دسته ی کلمات پرتکرار قرار دارند.

```
freq_words(movies['clean_plot'], 100)
```



در ادامه ابتدا ژانرها را به بردارهای باینری تبدیل کرده و با توجه به اینکه ۲۴ ژانر مختلف داریم، این ها را به عنوان برچسب در y ذخیره می کنیم.

```
from sklearn.preprocessing import MultiLabelBinarizer

multilabel_binarizer = MultiLabelBinarizer()
multilabel_binarizer.fit(movies['genre_imdb'])

# transform target variable
y = multilabel_binarizer.transform(movies['genre_imdb'])
```

در این جا با استفاده از TF-IDF استخراج ویژگی انجام می دهیم. TF-IDF بر تعداد تکرار کلمات توجه دارد. البته نه تنها وقوع یک کلمه در یک سند، بلکه در کل پیکره را در نظر می گیرد. از ۱۰۰۰۰ کلمه رایج در داده‌ها به عنوان ویژگی های خود استفاده می کنیم. می توان هر عدد دیگری در این جا برای این پارامتر مشخص کرد.

```
tfidf_vectorizer = TfidfVectorizer(max_df=0.8, max_features=10000)
```

داده ها را به دو مجموعه ی train و validation تقسیم می کنیم. به این صورت که ۸۰٪ داده ها در مجموعه ی train و باقی آن در مجموعه ی validation قرار می گیرد.

```
# split dataset into training and validation set
xtrain, xval, ytrain, yval = train_test_split(movies['clean_plot'], y, test_size=0.2, random_state=9)
```

در ادامه TF-IDF را بر روی این دو مجموعه اعمال می کنیم تا ویژگی ها را استخراج کند.

```
# create TF-IDF features
xtrain_tfidf = tfidf_vectorizer.fit_transform(xtrain)
xval_tfidf = tfidf_vectorizer.transform(xval)
```

در ادامه می خواهیم ابتدا الگوریتم LogisticRegression و سپس یکی از الگوریتم های خوشه بندی را اعمال کنیم. الگوریتم LogisticRegression را با رویکرد OneVsRestClassifier اعمال می کنیم. این رویکرد به این صورت عمل می کند که ابتدا یکی از ژانرها را در یک کلاس و بقیه را در کلاس دیگر قرار می دهد و طبقه بندی را به صورت باینری انجام میدهد. سپس دوباره یکی دیگر از ژانرها در یک کلاس و بقیه در کلاس دیگر. این کار ۲۴ بار تکرار می شود. زیرا ۲۴ ژانر مختلف داریم.

```
from sklearn.linear_model import LogisticRegression
from sklearn.multiclass import OneVsRestClassifier
from sklearn.metrics import f1_score
lr = LogisticRegression()
clf = OneVsRestClassifier(lr)
# fit model on train data
clf.fit(xtrain_tfidf, ytrain)
```

```
OneVsRestClassifier(estimator=LogisticRegression())
```

پس از آموزش مدل بر روی داده های train، آن را بر روی داده های validation و با استفاده از معیار ارزیابی F1-score ارزیابی میکنیم.

```
] # make predictions for validation set
  y_pred = clf.predict(xval_tfidf)

  # evaluate performance
  f1_score(yval, y_pred, average="micro")
```

```
0.3275705186533212
```

```
# predict probabilities
y_pred_prob = clf.predict_proba(xval_tfidf)

t = 0.3 # threshold value
y_pred_new = (y_pred_prob >= t).astype(int)

# evaluate performance
f1_score(yval, y_pred_new, average="micro")
```

```
0.5094217024041586
```

برای ارزیابی دو آستانه ی 0.5 و 0.3 در نظر گرفته می شود. آستانه به این معنا که احتمالا بزرگ تر یا مساوی 0.5 به 1 و بقیه صفر تبدیل شده اند (برای 0.3 نیز به همین منوال).

همانطور که مشاهده می شود هنگامی که آستانه از 0.5 به 0.3 تغییر می یابد، مقدار F1-score افزایش می یابد. بنابراین میتوان این نتیجه را گرفت که مقدار آستانه 0.3 بهتر است.

سیستم پیش بینی ژانر فیلم باید بتواند یک طرح فیلم را به شکل خام به عنوان ورودی دریافت کند و برچسب های ژانر خود را به عنوان خروجی ارائه دهد. در ادامه تابع استنتاج را بر روی چند نمونه از مجموعه اعتبارسنجی آزمایش میکنیم.

```
def infer_tags(q):
    q = clean_text(q)
    q = remove_stopwords(q)
    q_vec = tfidf_vectorizer.transform([q])
    q_pred = clf.predict(q_vec)
    return multilabel_binarizer.inverse_transform(q_pred)
```

```
for i in range(5):
    k = xval.sample(1).index[0]
    print("Movie: ", movies['title'][k], "\nPredicted genre: ", infer_tags(xval[k])), print("Actual genre: ",movies['genre_imdb'][k], "\n")
```

```
Movie:  How to Make an American Quilt (1995)
Predicted genre:  [('Drama',)]
Actual genre:  ['Comedy', 'Drama', 'Romance']

Movie:  Boys, Les (1997)
Predicted genre:  [('Comedy', 'Drama')]
Actual genre:  ['Comedy', 'Drama', 'Sport']

Movie:  Family Thing, A (1996)
Predicted genre:  [('Drama',)]
Actual genre:  ['Comedy', 'Drama']

Movie:  Mina Tannenbaum (1994)
Predicted genre:  [('Drama',)]
Actual genre:  ['Drama']

Movie:  Yankee Zulu (1994)
Predicted genre:  [('Drama',)]
Actual genre:  ['Comedy', 'Drama']
```

در ادامه می خواهیم الگوریتم DBScan که یکی از الگوریتم های خوشه بندی است را بر روی مجموعه داده اعمال کنیم. این الگوریتم دو پارامتر Eps (شعاع همسایگی) و MinPts (تعداد نقاط موجود در شعاع همسایگی) را دارد و به شدت به تغییر مقادیر این پارامترها حساس است. برای Eps یک لیست و برای MinPts یک رنج درنظر گرفته شده است. سپس بر روی مقادیر مختلف این دو پارامتر الگوریتم اجرا شده و معیار ارزیابی silhouette_score حاصل می شود. همچنین تعداد خوشه ها نیز نشان داده می شود.

```
from sklearn.cluster import DBSCAN
from sklearn.metrics import silhouette_score
for eps in [0.125,0.25,0.5,1,2,3,4,5]:
    for min_samples in range(1,50):
        db = DBSCAN(eps=eps, min_samples=min_samples)
        y = db.fit_predict(xtrain_tfidf)
        if len(set(db.labels_))>4 and len(set(db.labels_))<249:
            print (str(eps) + " " + str(min_samples) + ": "+ str(silhouette_score(xtrain_tfidf, db.labels_, metric='euclidean')))
```

```
0.125 2: 0.004837730460302407
0.25 2: 0.004837730460302407
0.5 2: 0.003769860089987392
1 2: 0.0072634027982073105
1 3: -0.0032491386164149923
```

```
db = DBSCAN(eps=1, min_samples=2).fit(xtrain_tfidf)

core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
dbscanlabels = db.labels_

# Number of clusters in labels, ignoring noise if present.
n_clusters_ = len(set(dbscanlabels)) - (1 if -1 in dbscanlabels else 0)

print('Estimated number of clusters: %d' % n_clusters_)
print("Silhouette Coefficient: %0.3f"
      % metrics.silhouette_score(xtrain_tfidf, dbscanlabels))
```

```
Estimated number of clusters: 41
Silhouette Coefficient: 0.007
```

```
db = DBSCAN(eps=1, min_samples=3).fit(xtrain_tfidf)

core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
dbscanlabels = db.labels_

# Number of clusters in labels, ignoring noise if present.
n_clusters_ = len(set(dbscanlabels)) - (1 if -1 in dbscanlabels else 0)

print('Estimated number of clusters: %d' % n_clusters_)
print("Silhouette Coefficient: %0.3f"
      % metrics.silhouette_score(xtrain_tfidf, dbscanlabels))
```

```
Estimated number of clusters: 6
Silhouette Coefficient: -0.003
```

```
db = DBSCAN(eps=0.5, min_samples=2).fit(xtrain_tfidf)

core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
dbscanlabels = db.labels_

# Number of clusters in labels, ignoring noise if present.
n_clusters_ = len(set(dbscanlabels)) - (1 if -1 in dbscanlabels else 0)

print('Estimated number of clusters: %d' % n_clusters_)
print("Silhouette Coefficient: %0.3f"
      % metrics.silhouette_score(xtrain_tfidf, dbscanlabels))
```

```
Estimated number of clusters: 11
Silhouette Coefficient: 0.004
```

```
db = DBSCAN(eps=1.25, min_samples=2).fit(xtrain_tfidf)

core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
dbscanlabels = db.labels_

# Number of clusters in labels, ignoring noise if present.
n_clusters_ = len(set(dbscanlabels)) - (1 if -1 in dbscanlabels else 0)

print('Estimated number of clusters: %d' % n_clusters_)
print("Silhouette Coefficient: %0.3f"
      % metrics.silhouette_score(xtrain_tfidf, dbscanlabels))
```

```
Estimated number of clusters: 88
Silhouette Coefficient: 0.021
```

آنچه از تغییر پارامترها و خروجی ها حاصل می شود در جدول زیر نشان داده شده است.

Eps	MinPts	number of clusters	Silhouette Coefficient
1	2	41	0.007
0.25	2	9	0.005
1.25	2	88	0.021
1	3	6	-0.003

معیار ارزیابی Silhouette Coefficient به این صورت است که مقدار فاصله ی عنصر با میانگین فاصله ی عناصر نزدیک ترین خوشه به آن خوشه و همچنین فاصله ی عنصر از عناصر درون خوشه را در نظر می گیرد (فرمول آن در اسلایدهای درسی بیان شده) و هر چه این مقدار به 1 نزدیک تر باشد بهتر است. در جدول مقادیر حاصل نشان داده شده و می توان مقایسه کرد و متوجه شد که کدام Eps و MinPts برای این مجموعه داده بهتر است.

پاسخ تمرین های ۱ تا ۴ در بالا به صورت کامل همراه با کد بیان شده است.

سوال ۱:

Lemmatization: یک تکنیک عادی‌سازی متن است که در پردازش زبان طبیعی شده و هر نوع کلمه را به حالت ریشه اصلی آن تغییر می‌دهد Lemmatization .وظیفه‌ی گروه‌بندی اشکال مختلف عطف کلمات را به شکل ریشه‌ای دارد که معنای یکسانی

دارند Lemmatization .معمولاً شامل استفاده از واژگان و تجزیه‌وتحلیل صرفی کلمات، حذف پایان‌های عطفی و برگرداندن فرم فرهنگ لغت یک کلمه (لم) است. به عبارتی می‌توان گفت که Lemmatization عبارت است از گروه‌بندی شکل‌های مختلف یک کلمه باهم. در پرس‌وجوهای جستجو، واژه‌سازی به کاربران نهایی اجازه می‌دهد تا هر نسخه‌ای از یک کلمه‌ی پایه را جستجو کنند و نتایج مرتبط را دریافت کنند. به‌عنوان مثال، رایانه می‌تواند کلماتی را که ریشه یکسانی ندارند، اما دارای همان معنای عطفی هستند، در کنار هم قرار دهد. گروه‌بندی کلمه "خوب" با کلماتی مانند "بهتر" و "بهترین" نمونه‌ای از واژه‌سازی است.

Lemmatizationیکی از بهترین راه‌ها برای کمک به چت‌بات‌ها برای درک بهتر سوالات مشتریان است. از آنجایی که این شامل تجزیه‌وتحلیل مورفولوژیکی کلمات است، ربات چت می‌تواند درک بهتری از معنای کلی جمله‌ای که در حال اصطلاح‌سازی است به دست آورد. همچنین Lemmatization برای فعال‌کردن ربات‌ها برای صحبت و مکالمه استفاده می‌شود. این موضوع باعث می‌شود واژه‌سازی بخش نسبتاً مهمی از درک زبان طبیعی و پردازش زبان طبیعی در هوش مصنوعی باشد.

Stemming: فرآیند کاهش یک کلمه به ریشه کلمه آن است که به پسوندها و پیشوندها یا به ریشه کلمات معروف به لم می‌چسبد. ریشه در درک زبان طبیعی و پردازش زبان طبیعی مهم است. الگوریتم‌های مختلفی جهت انجام عمل Stemming وجود دارد. در زبان انگلیسی الگوریتم Porter بسیار معروف است. این الگوریتم طبق یک سری قاعده‌ی منظم) مثلاً حذف s در آخر کلمات جمع (می‌تواند ریشه‌ی کلمات را با دقت خوبی به دست آورد. همچنین در زبان فارسی، الگوریتم کاظم تقوی، این کار را با دقت بالایی (برای کلمات فارسی) انجام می‌دهد.

به‌صورت کلی می‌توان اینگونه بیان کرد که Stemming فرایندی است که چند کاراکتر آخر یک کلمه را منشا می‌گیرد یا حذف می‌کند، که اغلب منجر به معانی و املای نادرست می‌شود. Lemmatization، زمینه را درنظر گرفته و کلمه را به شکل پایه معنی‌دار خود تبدیل می‌کند. به‌عنوان مثال، برای Lemmatization، مجموعه کلمات {فرماندهان، فرماندهی، فرمانده، فرمانده‌ای، فرمانده‌ی، فرماند} را "فرمانده" نتیجه می‌دهد. برای Stemming، رفتن به رفت، گفتن به گفت، آمدن به آمد، خوردن به خورد که در تمام این‌ها، -َن در آخر کلمات حذف شده است.

سوال ۲:

Continuous Bag of words (CBOW): روش کار CBOW به این صورت است که تمایل دارد احتمال یک کلمه را در یک زمینه پیش‌بینی کند. یک زمینه ممکن است یک کلمه یا گروهی از کلمات باشد. به این صورت عمل می‌کند که مثلاً سه کلمه قبل و سه کلمه‌ی بعد از کلمه‌ی مورد نظر را به شبکه داده و شبکه کلمه‌ی مورد نظر را در خروجی به ما می‌دهد. از شبکه‌های عصبی دو لایه می‌توان در این جا استفاده کرد.

Bag-Of-Word (BOW): راهی برای استخراج ویژگی‌ها از متن برای استفاده در مدل‌سازی، مانند الگوریتم‌های یادگیری ماشین است. این رویکرد بسیار ساده و انعطاف‌پذیر است و می‌تواند به روش‌های بی‌شماری برای استخراج ویژگی‌ها از اسناد استفاده شود. BOW نمایشی از متن است که وقوع کلمات را در یک سند توصیف می‌کند و شامل دو چیز است. ۱- واژگانی از کلمات شناخته‌شده. ۲- معیار حضور کلمات شناخته‌شده. این مدل تنها به این موضوع می‌پردازد که آیا کلمات شناخته شده در سند وجود دارند یا نه و اگر وجود دارند در کجای سند هستند.

سوال ۳:

روش K-means یکی از روش‌های پرکاربرد در خوشه‌بندی است. اما این روش نسبت به نویز و outlier به شدت حساس است. همچنین، داده‌هایی که شکل غیرمحدب دارند را به خوبی خوشه‌بندی نمی‌کند. به همین دلیل از روش DBScan در این جا استفاده شده که این دو مشکل k-means را برطرف کرده و بر روی این مجموعه داده به خوبی عمل می‌کند.