Import required packages

```
! pip install hazm
import hazm
! pip install autocorrect
from autocorrect import Speller
! pip install clean-text
from cleantext import clean
! pip install word2vec
import word2vec

import pandas as pd
import numpy as np
import nltk
from string import punctuation
import re
import nltk
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import Dense,Embedding,LSTM,SpatialDropout1D
from tensorflow.keras.models import Sequential
from sklearn.model_selection import train_test_split
from sklearn.svm import LinearSVC ,SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
```

```
  Created wheel for nltk: filename=nltk-3.3-py3-none-any.whl size=1394485 sha256=7202
  Stored in directory: /root/.cache/pip/wheels/9b/fd/0c/d92302c876e5de87ebd7fc0979d82
  Building wheel for libwapiti (setup.py) ... done
  Created wheel for libwapiti: filename=libwapiti-0.2.1-cp37-cp37m-linux_x86_64.whl s
  Stored in directory: /root/.cache/pip/wheels/ab/b2/5b/0fe4b8f5c0e65341e8ea7bb3f4a6e
Successfully built nltk libwapiti
Installing collected packages: nltk, libwapiti, hazm
  Attempting uninstall: nltk
    Found existing installation: nltk 3.7
    Uninstalling nltk-3.7:
      Successfully uninstalled nltk-3.7
Successfully installed hazm-0.7.0 libwapiti-0.2.1 nltk-3.3
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/p
Collecting autocorrect
  Downloading autocorrect-2.6.1.tar.gz (622 kB)
     |████████████████████████████████| 622 kB 32.5 MB/s
Building wheels for collected packages: autocorrect
  Building wheel for autocorrect (setup.py) ... done
  Created wheel for autocorrect: filename=autocorrect-2.6.1-py3-none-any.whl size=622
  Stored in directory: /root/.cache/pip/wheels/54/d4/37/8244101ad50b0f7d9bffd93ce58ed
Successfully built autocorrect
Installing collected packages: autocorrect
Successfully installed autocorrect-2.6.1
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/p
Collecting clean-text
  Downloading clean_text-0.6.0-py3-none-any.whl (11 kB)
Collecting ftfy<7.0,>=6.0
  Downloading ftfy-6.1.1-py3-none-any.whl (53 kB)
     |████████████████████████████████| 53 kB 2.4 MB/s
Collecting emoji<2.0.0,>=1.0.0
  Downloading emoji-1.7.0.tar.gz (175 kB)
     |████████████████████████████████| 175 kB 61.0 MB/s
Requirement already satisfied: wcwidth>=0.2.5 in /usr/local/lib/python3.7/dist-packag
Building wheels for collected packages: emoji
  Building wheel for emoji (setup.py) ... done
  Created wheel for emoji: filename=emoji-1.7.0-py3-none-any.whl size=171046 sha256=a
  Stored in directory: /root/.cache/pip/wheels/8a/4e/b6/57b01db010d17ef6ea9b40300af72
Successfully built emoji
Installing collected packages: ftfy, emoji, clean-text
Successfully installed clean-text-0.6.0 emoji-1.7.0 ftfy-6.1.1
WARNING:root:Since the GPL-licensed package `unidecode` is not installed, using Pytho
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/p
Collecting word2vec
  Downloading word2vec-0.11.1.tar.gz (42 kB)
     |████████████████████████████████| 42 kB 1.4 MB/s
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
    Preparing wheel metadata ... done
Requirement already satisfied: numpy>=1.9.2 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages (from
Building wheels for collected packages: word2vec
  Building wheel for word2vec (PEP 517) ... done
  Created wheel for word2vec: filename=word2vec-0.11.1-py2.py3-none-any.whl size=1421
  Stored in directory: /root/.cache/pip/wheels/c9/c0/d4/29d797817e268124a32b6cf8beb8b
Successfully built word2vec
Installing collected packages: word2vec
Successfully installed word2vec-0.11.1
```

Data reading From Google drive

```
from google.colab import drive

drive.mount('/content/drive', force_remount=True)

!ls '/content/drive/MyDrive/data.csv'

data = pd.read_csv('/content/drive/MyDrive/data.csv', encoding='utf-8')
df = pd.DataFrame(data)
df.drop(columns=['Unnamed: 0', 'Unnamed: 0.1'], inplace=True)

```

```
Mounted at /content/drive
/content/drive/MyDrive/data.csv
```

```
1 data = df
2 data
```

	comment_text	film_name
0	NaN	BlackCat
1	NaN	BlackCat
2	خیلی قشنگه . روزگار الان و خوب به تصویر کشیده	BlackCat
3	عالی آقای رادان	BlackCat
4	قشنگ بود	BlackCat
...	...	...
143	بسیار خوب و درست	Death_of_Salesman
144	👌مرگ فروشنده عالی	Death_of_Salesman
145	👌جالب بود	Death_of_Salesman
146	عالی بود احسنت به امید فیلم های جدیدتون	Death_of_Salesman
147	عالی	Death_of_Salesman

148 rows × 2 columns

Data Cleaning

```
1 # print data information
2 print('data information')
3 print(data.info(), '\n')
4
5 # print missing values information
6 print('missing values stats')
7 print(data.isnull().sum(), '\n')
8
9
```

```
data information
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 148 entries, 0 to 147
Data columns (total 2 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   comment_text  128 non-null    object
 1   film_name     148 non-null    object
dtypes: object(2)
```

```
memory usage: 2.4+ KB
None

missing values stats
comment_text    20
film_name        0
dtype: int64
```

```
# handle some conflicts with the dataset structure
# you can find a reliable solution, for the sake of the simplicity
# I just remove these bad combinations!


data = data.dropna(subset=['film_name'])
data = data.dropna(subset=['comment_text'])

data = data.reset_index(drop=True)




# print data information
print('data information')
print(data.info(), '\n')

# print missing values information
print('missing values stats')
print(data.isnull().sum(), '\n')

# print some missing values
print('some missing values')
print(data[data['film_name'].isnull()].iloc[:5], '\n')
```

```
data information
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 128 entries, 0 to 127
Data columns (total 2 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   comment_text  128 non-null    object
 1   film_name     128 non-null    object
dtypes: object(2)
memory usage: 2.1+ KB
None

missing values stats
comment_text    0
film_name       0
dtype: int64
```

```
some missing values
Empty DataFrame
Columns: [comment_text, film_name]
Index: []
```

```
1 data
```

	comment_text	film_name
0	خیلی قشنگه . روزگار الان و خوب به تصویر کشیده	BlackCat
1	عالی آقای رادان	BlackCat
2	قشنگ بود	BlackCat
3	عالیییی	BlackCat
4	من دیدم خیلی خفنه	BlackCat
...	...	...
123	بسیار خوب و درست	Death_of_Salesman
124	👌مرگ فروشنده عالی	Death_of_Salesman
125	👌جالب بود	Death_of_Salesman
126	عالی بود احسنت به امید فیلم های جدیدتون	Death_of_Salesman
127	عالی	Death_of_Salesman

128 rows × 2 columns

Texet Cleaning (Text Normalize)

```
1 def sentence_tokenize(text):
2     return nltk.sent_tokenize(text)
```

```
1 def word_tokenize(text):
2 
3         return nltk.word_tokenize(text)
```

```
1 import nltk
2 nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
True
```

```
1 data['comment_text'].apply(sentence_tokenize)
```

```
0      [خیلی قشنگه .. روزگار الان و خوب به تصویر کشیده]
1                                    [عالی آقای رادان]
2                                           [قشنگ بود]
3                                           [عالیییی]
4                                  [من دیدم خیلی خفنه]
                              ...
123                                  [بسیار خوب و درست]
124                                 [👌مرگ فروشنده عالی]
125                                        [👌جالب بود]
126          [عالی بود احسنت به امید فیلم های جدیدتون]
127                                              [عالی]
Name: comment_text, Length: 128, dtype: object
```

```
1 def remove_numbers(text):
2         """
3         take string input and return a clean text without numbers.
4         Use regex to discard the numbers.
5         """
6         output = ''.join(c for c in text if not c.isdigit())
7         return output
```

```
1 def remove_punct(text):
2
3         return ''.join(c for c in text if c not in punctuation)
```

```
1 def remove_wierds(text):
2   wierd_pattern = re.compile("["
3         u"\U0001F600-\U0001F64F"  # emoticons
4         u"\U0001F300-\U0001F5FF"  # symbols & pictographs
5         u"\U0001F680-\U0001F6FF"  # transport & map symbols
6         u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
7         u"\U00002702-\U000027B0"
8         u"\U000024C2-\U0001F251"
9         u"\U0001f926-\U0001f937"
10        u'\U00010000-\U0010ffff'
11        u"\u200d"
12        u"\u2640-\u2642"
13        u"\u2600-\u2B55"
14        u"\u23cf"
15        u"\u23e9"
16        u"\u231a"
17        u"\u3030"
18        u"\ufe0f"
19        u"\u2069"
20        u"\u2066"
21        u"\u200c"
```

```
        u"\u2068"
        u"\u2067"
        "]+", flags=re.UNICODE)
    text = wierd_pattern.sub(r'', text)
    return text
```

```
clean_data = data['comment_text'].apply(remove_wierds)
```

```
data['clean_comment'] = clean_data
```

```
data.head()
```

	comment_text	film_name	clean_comment
0	خیلی قشنگه . روزگار الان و خوب به تصویر کشیده	BlackCat	خیلی قشنگه . روزگار الان و خوب به تصویر کشیده
1	عالی آقای رادان	BlackCat	عالی آقای رادان
2	قشنگ بود	BlackCat	قشنگ بود
3	عالیییی	BlackCat	عالیییی
4	من دیدم خیلی خفنه	BlackCat	من دیدم خیلی خفنه

```
data.drop(columns=['comment_text'], inplace=True)
```

```
data.head(5)
```

	film_name	clean_comment
0	BlackCat	خیلی قشنگه . روزگار الان و خوب به تصویر کشیده
1	BlackCat	عالی آقای رادان
2	BlackCat	قشنگ بود
3	BlackCat	عالیییی
4	BlackCat	من دیدم خیلی خفنه

Building My Model

build phreser for vocabulary

```
from gensim.models.phrases import Phrases, Phraser

sent = [row.split() for row in data['clean_comment']]
phrases = Phrases(sent, min_count=30)
bigram = Phraser(phrases)
```

```
sentences = bigram[sent]
```

word frequncy

```
from collections import defaultdict
word_freq = defaultdict(int)
for sent in sentences:
    for i in sent:
        word_freq[i] += 1
len(word_freq)
```

```
390
```

```
sorted_word_freq = sorted(word_freq, key=word_freq.get, reverse=True)[:20]
```

Training Model

```
import multiprocessing
from gensim.models import Word2Vec
```

```
cores = multiprocessing.cpu_count()
```

```
w2v_model = Word2Vec(min_count=2,
                     window=2,
                     size=100,
                     sample=6e-5,
                     alpha=0.02,
                     min_alpha=0.0007,
                     negative=20,
                     workers=cores-1)
```

```
from time import time
t = time()

w2v_model.build_vocab(sentences, progress_per=10000)
```

```
5
6 print('Time to build vocab: {} mins'.format(round((time() - t) / 60, 5)))
```

```
1 t = time()
2
3 w2v_model.train(sentences, total_examples=w2v_model.corpus_count, epochs=30, report_
4
5 print('Time to train the model: {} mins'.format(round((time() - t) / 60, 5)))
```

```
    Time to train the model: 0.00171 mins
```

```
1 w2v_model.init_sims(replace=True)
```

Example Getting word similarity for weight words

```
1 w2v_model.wv.most_similar(negative=["عالی"])
```

```
    [('0.30535346269607544', 'العاده'),
     ('0.23930777609348297', 'دیدن'),
     ('0.2309877872467041', 'عوامل'),
     ('0.178193137049675', 'ایران'),
     ('0.17481639981269836', 'همه'),
     ('0.1694687306880951', 'جالب'),
     ('0.16051331162452698', 'میکنم'),
     ('0.15472202003002167', 'بود.'),
     ('0.14574342966079712', 'تو'),
     ('0.13523930311203003', 'بعد')]
```

Set vector and PCA for must freq words

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 %matplotlib inline
4
5 import seaborn as sns
6 sns.set_style("darkgrid")
7
8 from sklearn.decomposition import PCA
9 from sklearn.manifold import TSNE
```

```
1 def tsnescatterplot(model, word, list_names):
2     arrays = np.empty((0, 100), dtype='f')
3     word_labels = [word]
4     color_list  = ['red']
```

```

    # adds the vector of the query word
    arrays = np.append(arrays, model.wv.__getitem__([word]), axis=0)

    # gets list of most similar words
    close_words = model.wv.most_similar([word])

    # adds the vector for each of the closest words to the array
    for wrd_score in close_words:
        wrd_vector = model.wv.__getitem__([wrd_score[0]])
        word_labels.append(wrd_score[0])
        color_list.append('blue')
        arrays = np.append(arrays, wrd_vector, axis=0)

    # adds the vector for each of the words from list_names to the array
    for wrd in list_names:
        wrd_vector = model.wv.__getitem__([wrd])
        word_labels.append(wrd)
        color_list.append('green')
        arrays = np.append(arrays, wrd_vector, axis=0)

    # Reduces the dimensionality from 100 to 20 dimensions with PCA
    reduc = PCA(n_components=20).fit_transform(arrays)

    # Finds t-SNE coordinates for 2 dimensions
    np.set_printoptions(suppress=True)

    Y = TSNE(n_components=2, random_state=0, perplexity=15).fit_transform(reduc)

    # Sets everything up to plot
    df = pd.DataFrame({'x': [x for x in Y[:, 0]],
                       'y': [y for y in Y[:, 1]],
                       'words': word_labels,
                       'color': color_list})

    fig, _ = plt.subplots()
    fig.set_size_inches(9, 9)

    # Basic plot
    p1 = sns.regplot(data=df,
                     x="x",
                     y="y",
                     fit_reg=False,
                     marker="o",
                     scatter_kws={'s': 40,
                                  'facecolors': df['color']
                                 }
                    )

```

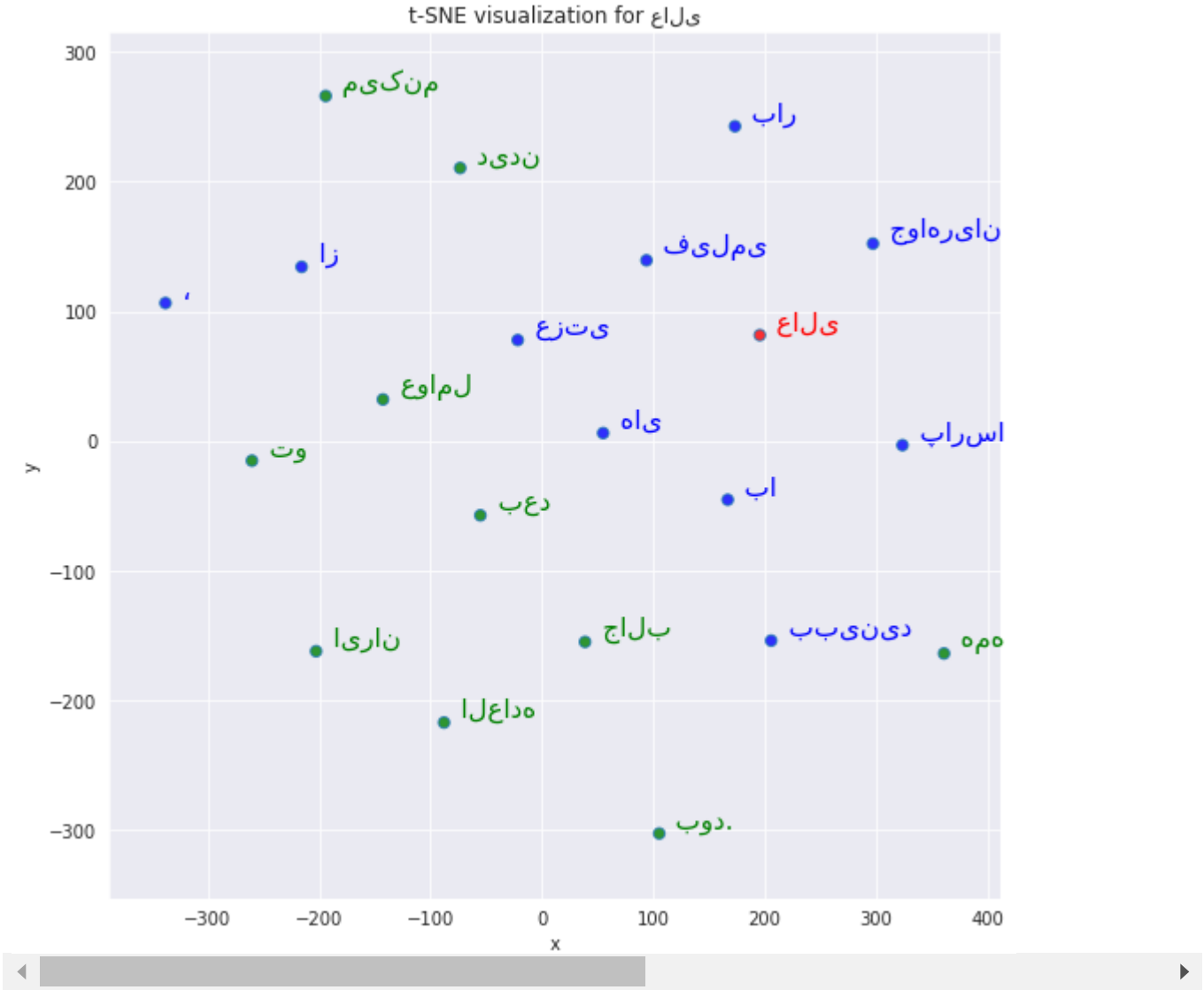
```
    # Adds annotations one by one with a loop
    for line in range(0, df.shape[0]):
         p1.text(df["x"][line],
                 df['y'][line],
                 '  ' + df["words"][line].title(),
                 horizontalalignment='left',
                 verticalalignment='bottom', size='medium',
                 color=df['color'][line],
                 weight='normal'
                ).set_size(15)


    plt.xlim(Y[:, 0].min()-50, Y[:, 0].max()+50)
    plt.ylim(Y[:, 1].min()-50, Y[:, 1].max()+50)

    plt.title('t-SNE visualization for {}'.format(word.title()))
```

```
listofword = ['جالب', 'واقعا', 'العاده','فوق', 'بود', 'زیبا', 'پیشنهاد', 'قشنگ', 'عالی']
tsnescatterplot(w2v_model, 'خوب', listofword)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/manifold/_t_sne.py:783: FutureWarning: Th
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/manifold/_t_sne.py:793: FutureWarning: Th
  FutureWarning,
```

```
1 tsnescatterplot(w2v_model, 'عالی', [i[0] for i in w2v_model.wv.most_similar(negative=
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/manifold/_t_sne.py:783: FutureWarning: Th
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/manifold/_t_sne.py:793: FutureWarning: Th
  FutureWarning,
```



Getting word similarity for weight words

```
1 x = "عالی"
2 w2v_model.wv.most_similar(positive=[x])
```

```
[('0.25735166668891907 ,'فیلمی),
 ('0.200503408908844 ,'از),
 ('،', 0.19855839014053345),
```

```
 ('0.1807647943496704', 'های'),
 ('0.1769590973854065', 'بار'),
 ('0.1753464639186859', 'پارسا'),
 ('0.16088369488716125', 'جواهریان'),
 ('0.14725281298160553', 'عزتی'),
 ('0.1445910930633545', 'با'),
 ('0.1445724368095398', 'ببینید')]
```

```
1 x = "خوب"
2 w2v_model.wv.most_similar(positive=[x])
```

```
[('0.23057061433792114', 'بود.'),
 ('0.17563724517822266', 'بزرگ'),
 ('0.16716060042381287', 'فیلمی'),
 ('0.1634744107723236', 'دیدن'),
 ('0.14385637640953064', 'تو'),
 ('0.14230720698833466', 'آفرینی'),
 ('0.14091373980045319', 'درد'),
 ('0.1406758427619934', 'که'),
 ('0.13018135726451874', 'بار'),
 ('0.1285800337791443', 'بامزه')]
```

```
1 x = "قشنگ"
2 w2v_model.wv.most_similar(positive=[x])
```

```
[('0.20163458585739136', 'فرمان'),
 ('0.19065552949905396', 'لذت'),
 ('0.1825539469718933', 'فیلم'),
 ('0.1747557818889618', 'آفرینی'),
 ('0.17182226479053497', 'خوبی'),
 ('0.1606597751379013', 'ای'),
 ('0.1382228136062622', 'کارگردان'),
 ('0.1373421549797058', 'کارگردانی'),
 ('0.13294094800949097', 'یه'),
 ('0.13134098052978516', 'پیشنهاد')]
```

```
1 x = "پیشنهاد"
2 w2v_model.wv.most_similar(positive=[x])
```

```
[('0.20704875886440277', 'عالیییی'),
 ('0.199214905500412', 'های'),
 ('0.193104550242424', 'کارگردانی'),
 ('0.18775789439678192', 'داستان'),
 ('0.18054604530334473', 'بار'),
 ('0.17335109412670135', 'بی'),
 ('0.17311590909957886', 'بود'),
 ('0.16022425889968872', 'نقش'),
 ('0.15579932928085327', 'مرسی'),
 ('0.15424613654613495', 'احسنت')]
```

```
1 x = "عالی"
```

```
y = "پیشنهاد"
z = "قشنگ"
v = "خوب"
word_predict_instance = w2v_model.wv.most_similar(positive=[x, y, z, v])
```

```
word_predict_instance
```

```
[('فیلمی', 0.27535003423690796),
 ('کارگردانی', 0.21758560836315155),
 ('بار', 0.21535061299800873),
 ('های', 0.21105468273162842),
 ('از', 0.1683465540409088),
 ('جواهریان', 0.16272494196891785),
 ('احسنت', 0.16228868067264557),
 ('که', 0.1590237319469452),
 ('آفرینی', 0.1547035574913025),
 ('پارسا', 0.15276700258255005)]
```

sentiment prediction

```
type(word_predict_instance)
```

```
list
```

```
nude_word = [x[0] for x in word_predict_instance]
```

```
nude_word
```

```
['فیلمی',
 'کارگردانی',
 'بار',
 'های',
 'از',
 'جواهریان',
 'احسنت',
 'که',
 'آفرینی',
 'پارسا']
```

```
comment_box = []
for comment in data['clean_comment']:
  comment_box.append(comment)
```

```
comment_box
```

```
 'بازی جواد عزتی حرف نداره. تست میزراد',
'مرسی کارگردانی و مرسی بازی های قوی',
'عالی بود خیلی تاثیرگذار و قابل تامل برای جامعه ایرانی هست احسنت به کارگردان و بازیگران توانای فیلم',
'خیلی خوب بود برای اولین بار فیلمی دیدم که بجای پنهان کردن مشکلی که وجود داره منطقی تفسیرش کرده بود',
```

```
'فیلم نامه خوب، بازیگران خوب . فیلمی دلنشین و متفاوت به دور از بازی های کلیشه ای',
'واقعا لذت بردم از دیدن فیلم فوق العاده بود',
'واقعا بهترین فیلمی هست که دیدم و بهترین بازیگرا و بهترین بازیها...واقعا مرسی',
'عالی بود',
'بسیار عالی و اموزنده در مورد ترنسهای بیگناه جامعه',
'خوب وتاثیر گذار',
'عالی بود',
'عالی بود با دیدی کاملا زیبا حتما ببینید',
'.بسیار زیبا و تاثیرگذار بود',
'واقعا فیلم زیبا و تاثیر گذاری بود . بازیها بسیار باور پذیر و زیبا .نکته مهم اینه که واقعا برای پدر مادرا مواجهه با این
. شرایط خیلی سخته . فرهنگ سازی تو این زمینه بسیار کار واجب و مهمیه',
'عالی',
'سلام خیلی عالی بود',
'وای خدای من این یکی از مفهومی ترین فیلم هایی بود که تا امروز دیدم آخرای فیلم قلبم به درد اومده بود و به پهنای
صورتم اشک میریختم عالی بود روح خانم فرشته طائرپور شاد و یادش گرامی خدایا خودت کمکشون کن تا از این برزخ در
بیان و با آرامش زندگی کنن',
'بی نظیر بود تنها فیلمی که در ایران ساخته شده و به وضوح بخشی از داستان زندگی این افراد رو نشون میده',
'عالی',
'عالی بود دوسش داشتم',
'خیلی خوب بود ممنونم',
'.عالی بود...فقط باید فرهنگسازی کرد',
'واقعا قشنگ بود',
'خیلی قشنگ بود واقعیت امروزی جامعه ی ما امیدوارم بتونیم بیشتر درکشون کنیم',
'عالی بود',
' بامزه بود',
'خیلی عالی بود.طنز به جا و بامزه',
'خیلی خوب بود',
'خیلی عالی بود پیشنهاد میشه حتما ببینید',
'عالی',
'عالیه توصیه میکنم حتما ببینید',
'عالی بود',
'عالیییی',
'عالیه ارزش وقت گذاشتن داره',
'وای عالی',
'عالی',
'عالی',
'عالی بود',
'بازی بهرنگ علوی فوق العاده زیبا بود',
'بهرنگ علوی عالیه',
'خیلی داستان جدید و نویی داشت...عالی بود',
'قشنگ بود ارزش دیدن داره',
'نازنین بیاتی خیلی قشنگ بازی کرد',
'خوب بود فیلم مناسبی برای اوقات فراغته',
'عالی بود',
'خیلی عالی بود',
'خوب بود',
'دم سازندش گررررم',
'عالی',
'عالی بود',
'بسیار خوب و درست',
'مرگ فروشنده عالی',
'جالب بود',
'عالی بود احسنت به امید فیلم های جدیدتون',
'عالی']
```

```
from hazm import word_tokenize


word_token = word_tokenize(str(comment_box))
```

```
! pip install datasketch
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
Collecting datasketch
  Downloading datasketch-1.5.8-py2.py3-none-any.whl (76 kB)
     |████████████████████████████████| 76 kB 6.7 MB/s
Requirement already satisfied: numpy>=1.11 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.7/dist-packages (
Installing collected packages: datasketch
Successfully installed datasketch-1.5.8
```

Jacard similarity for get similarity point of must freq word with comments

```
from datasketch import MinHash

data1 = ['فیلم' ,'نبود' ,'بد','عالیه' ,'بود' ,'خوب','پیشنهاد','عالی' ,'بازی']
data2 = word_token

m1, m2 = MinHash(), MinHash()
for d in data1:
    m1.update(d.encode('utf8'))
for d in data2:
    m2.update(d.encode('utf8'))
print("Estimated Jaccard for data1 and data2 is", m1.jaccard(m2))
s1 = set(data1)
s2 = set(data2)
actual_jaccard = float(len(s1.intersection(s2)))/float(len(s1.union(s2)))
print("Actual Jaccard for data1 and data2 is", actual_jaccard)
```

```
Estimated Jaccard for data1 and data2 is 0.0078125
Actual Jaccard for data1 and data2 is 0.02122641509433962
```

Cosine Similarity for get similarity point of must freq word with

```
from collections import Counter
from math import sqrt

def word2vec(word):
    # count the characters in word
    cw = Counter(word)
    # precomputes a set of the different characters
    sw = set(cw)
    # precomputes the "length" of the word vector
    lw = sqrt(sum(c*c for c in cw.values()))
    return cw, sw, lw

def cosdis(v1, v2):
    # which characters are common to the two words?
    common = v1[1].intersection(v2[1])
    # by definition of cosine distance we have
    return sum(v1[0][ch]*v2[0][ch] for ch in common)/v1[2]/v2[2]


list_A = ['عالی']
list_B = comment_box

result1 = []

for key in list_A:
    for word in list_B:
      res = cosdis(word2vec(word), word2vec(key))
      result1.append(res)
      print("The cosine similarity between : {} and : {} is: {}".format(word, key, r
```

```
The cosine similarity between : عالییی and : عالی is: 86.60254037844388
The cosine similarity between : بازی جواد عزتی حرف نداره. دست مریزاد and : عالی is: 38.138503
The cosine similarity between : مرسی کارگردانی و مرسی بازی های قوی and : عالی is: 44.9013255

The cosine similarity between : قابل تامل برای جامعه ایرانی هست احسنت به کارگردان و بازیگران توانای فیلم
The cosine similarity between : یلمی دیدم که بجای پنهان کردن مشکلی که وجود داره منطقی تفسیرش کرده بود
The cosine similarity between : وب، بازیگران خوب . فیلمی دلنشین و متفاوت به دور از بازی های کلیشه ای
The cosine similarity between : واقعا لذت بردم از دیدن فیلم فوق العاده بود and : عالی is: 47.140452
The cosine similarity between : ا بهترین فیلمی هست که دیدم و بهترین بازیگرا و بهترین بازیها...واقعا مرسی
The cosine similarity between : عالی بود and : عالی is: 70.71067811865474
The cosine similarity between : بسیار عالی و اموزنده در مورد ترنسهای بیگناه جامعه and : عالی is: 45.
The cosine similarity between : خوب وتاثیر گذار and : عالی is: 31.277162108561217
The cosine similarity between : عالی بود and : عالی is: 70.71067811865474
The cosine similarity between : عالی بود با دیدی کاملا زیبا حتما ببینید and : عالی is: 56.373452100
The cosine similarity between : بسیار زیبا و تاثیرگذار بود. and : عالی is: 41.537358036784866
The cosine similarity between : ین شرایط خیلی سخته . فرهنگ سازی تو این زمینه بسیار کار واجب و مهمیه
The cosine similarity between : عالی and : عالی is: 100.0
The cosine similarity between : سلام خیلی عالی بود and : عالی is: 72.99963950884315
The cosine similarity between : امی خدایا خودت کمکشون کن تا از این برزخ در بیان و با آرامش زندگی کنن
The cosine similarity between : ایران ساخته شده و به وضوح بخشی از داستان زندگی این افراد رو نشون میده
```

```
The cosine similarity between : عالی and : عالی is: 100.0
The cosine similarity between : عالی بود دوسش داشتم and : عالی is: 41.09974682633932
The cosine similarity between : خیلی خوب بود ممنونم and : عالی is: 22.360679774997898
The cosine similarity between : عالی بود...فقط باید فرهنگسازی کرد. and : عالی is: 44.172610429
The cosine similarity between : واقعا قشنگ بود and : عالی is: 31.980107453341567
The cosine similarity between : یلی قشنگ بود واقعیت امروزی جامعه ی ما امیدوارم بتونیم بیشتر درکشون کنیم
The cosine similarity between : عالی بود and : عالی is: 70.71067811865474
The cosine similarity between : بامزه بود and : عالی is: 13.36306209562122
The cosine similarity between : خیلی عالی بود.طنز به جا و بامزه and : عالی is: 48.245064067700
The cosine similarity between : خیلی خوب بود and : عالی is: 31.980107453341567
The cosine similarity between : خیلی عالی بود پیشنهاد میشه حتما ببینید and : عالی is: 55.331674499
The cosine similarity between : عالی and : عالی is: 100.0
The cosine similarity between : عالیه توصیه میکنم حتما ببینید and : عالی is: 51.28225940683707
The cosine similarity between : عالی بود and : عالی is: 70.71067811865474
The cosine similarity between : عالیییی and : عالی is: 75.59289460184544
The cosine similarity between : عالیه ارزش وقت گذاشتن داره and : عالی is: 45.95725150090289
The cosine similarity between : وای عالی and : عالی is: 77.45966692414834
The cosine similarity between : عالی and : عالی is: 100.0
The cosine similarity between : عالی and : عالی is: 100.0
The cosine similarity between : عالی بود and : عالی is: 70.71067811865474
The cosine similarity between : بازی بهرنگ علوی فوق العاده زیبا بود and : عالی is: 52.203689766
The cosine similarity between : بهرنگ علوی عالیه and : عالی is: 68.64064729836443
The cosine similarity between : خیلی داستان جدید و نویی داشت...عالی بود and : عالی is: 53.25007
The cosine similarity between : قشنگ بود ارزش دیدن داره and : عالی is: 20.22599587389726
The cosine similarity between : نازنین بیاتی خیلی قشنگ بازی کرد and : عالی is: 49.266463908214
The cosine similarity between : خوب بود فیلم مناسبی برای اوقات فراغته and : عالی is: 41.25143236
The cosine similarity between : عالی بود and : عالی is: 70.71067811865474
The cosine similarity between : خیلی عالی بود and : عالی is: 72.98004491997617
The cosine similarity between : خوب بود and : عالی is: 0.0
The cosine similarity between : دم سازندش گرررررم and : عالی is: 8.57492925712544
The cosine similarity between : عالی and : عالی is: 100.0
The cosine similarity between : عالی بود and : عالی is: 70.71067811865474
The cosine similarity between : بسیار خوب و درست and : عالی is: 18.257418583505537
The cosine similarity between : مرگ فروشنده عالی and : عالی is: 44.721359549995796
The cosine similarity between : جالب بود and : عالی is: 31.622776601683793
The cosine similarity between : عالی بود احسنت به امید فیلم های جدیدتون and : عالی is: 50.8913357
The cosine similarity between : عالی and : عالی is: 100.0
```

```
1 result1
```

```
0.6546536707079772,
0.228747855498907,
0.3434014098717226,
0.39086797998528583,
0.5218624584427538,
0.7071067811865475,
0.17677669529663687,
0.30151134457776363,
0.6246950475544243,
1.0,
0.3180732125814321,
0.08838834764831843,
0.45813068106189153,
0.4152697672499609,
0.3764734830828951
```

```
0.5393598899705937,
0.36380343755449945,
0.0,
0.40147753427348304,
0.45045960013229974,
0.4539545862254728,
0.7071067811865475,
0.3898846627545847,
0.5753964555687505,
0.3136075378219869,
0.5829988340034981,
0.5482823149915702,
0.21320071635561041,
0.4537426064865151,
0.3913118960624632,
0.4191368221424546,
0.5188745216627708,
0.4190581774617469,
0.5060243137049899,
0.7808688094430304,
0.354374653931171,
0.6864064729836442,
0.7216878364870323,
0.4472135954999579,
0.3988620176087328,
0.4558423058385518,
0.20225995873897262,
0.5163977794943222,
0.7071067811865475,
0.6669729688499156,
0.5238227218271836,
0.7205766921228921,
0.3849001794597505,
0.7071067811865475,
0.5941331052724881,
0.15309310892394865,
0.8660254037844387,
0.38138503569823695,
0.4490132550669373,
0.563300712149108,
0.3671238906694095,
0.4448992785373876,
0.4714045207910317.
```

Now We See Cosine Similarity Have Better Result

Work Cosine Similarity For other weited words

```
1 d_f = pd.DataFrame()
```

```
d_f['perfect_rate'] = result1
```

```
def word2vec(word):
    # count the characters in word
    cw = Counter(word)
    # precomputes a set of the different characters
    sw = set(cw)
    # precomputes the "length" of the word vector
    lw = sqrt(sum(c*c for c in cw.values()))
    return cw, sw, lw

def cosdis(v1, v2):
    # which characters are common to the two words?
    common = v1[1].intersection(v2[1])
    # by definition of cosine distance we have
    return sum(v1[0][ch]*v2[0][ch] for ch in common)/v1[2]/v2[2]


list_A = ['خوب']
list_B = comment_box

result2 = []

for key in list_A:
    for word in list_B:
      res = cosdis(word2vec(word), word2vec(key))
      result2.append(res)
      #print("The cosine similarity between : {} and : {} is: {}".format(word, key,


```

```
d_f['good_rate'] = result2
```

```
def word2vec(word):
    # count the characters in word
    cw = Counter(word)
    # precomputes a set of the different characters
    sw = set(cw)
    # precomputes the "length" of the word vector
    lw = sqrt(sum(c*c for c in cw.values()))
    return cw, sw, lw

def cosdis(v1, v2):
    # which characters are common to the two words?
    common = v1[1].intersection(v2[1])
```

```
    # by definition of cosine distance we have
    return sum(v1[0][ch]*v2[0][ch] for ch in common)/v1[2]/v2[2]


list_A = ['پیشنهاد']
list_B = comment_box

result4 = []

for key in list_A:
    for word in list_B:
      res = cosdis(word2vec(word), word2vec(key))
      result4.append(res)
      #print("The cosine similarity between : {} and : {} is: {}".format(word, key,
```

```
d_f['propose_rate'] = result4
```

```
def word2vec(word):
    # count the characters in word
    cw = Counter(word)
    # precomputes a set of the different characters
    sw = set(cw)
    # precomputes the "length" of the word vector
    lw = sqrt(sum(c*c for c in cw.values()))
    return cw, sw, lw

def cosdis(v1, v2):
    # which characters are common to the two words?
    common = v1[1].intersection(v2[1])
    # by definition of cosine distance we have
    return sum(v1[0][ch]*v2[0][ch] for ch in common)/v1[2]/v2[2]


list_A = ['بد']
list_B = comment_box

result3 = []

for key in list_A:
    for word in list_B:
      res = cosdis(word2vec(word), word2vec(key))
      result3.append(res)
      #print("The cosine similarity between : {} and : {} is: {}".format(word, key,
```

```
d_f['bad_rate'] = result3
```

```
def word2vec(word):
    # count the characters in word
    cw = Counter(word)
    # precomputes a set of the different characters
    sw = set(cw)
    # precomputes the "length" of the word vector
    lw = sqrt(sum(c*c for c in cw.values()))
    return cw, sw, lw

def cosdis(v1, v2):
    # which characters are common to the two words?
    common = v1[1].intersection(v2[1])
    # by definition of cosine distance we have
    return sum(v1[0][ch]*v2[0][ch] for ch in common)/v1[2]/v2[2]


list_A = ['نبود']
list_B = comment_box

result5 = []

for key in list_A:
    for word in list_B:
      res = cosdis(word2vec(word), word2vec(key))
      result5.append(res)
      #print("The cosine similarity between : {} and : {} is: {}".format(word, key,


```

```
d_f['not_rate'] = result5
```

```
def word2vec(word):
    # count the characters in word
    cw = Counter(word)
    # precomputes a set of the different characters
    sw = set(cw)
    # precomputes the "length" of the word vector
    lw = sqrt(sum(c*c for c in cw.values()))
    return cw, sw, lw

def cosdis(v1, v2):
    # which characters are common to the two words?
    common = v1[1].intersection(v2[1])
    # by definition of cosine distance we have
    return sum(v1[0][ch]*v2[0][ch] for ch in common)/v1[2]/v2[2]
```

```

list_A = ['فیلم']
list_B = comment_box

result6 = []

for key in list_A:
    for word in list_B:
      res = cosdis(word2vec(word), word2vec(key))
      result6.append(res)
     # print("The cosine similarity between : {} and : {} is: {}".format(word, key,


```

```
d_f['film_rate'] = result6
```

```
def word2vec(word):
    # count the characters in word
    cw = Counter(word)
    # precomputes a set of the different characters
    sw = set(cw)
    # precomputes the "length" of the word vector
    lw = sqrt(sum(c*c for c in cw.values()))
    return cw, sw, lw

def cosdis(v1, v2):
    # which characters are common to the two words?
    common = v1[1].intersection(v2[1])
    # by definition of cosine distance we have
    return sum(v1[0][ch]*v2[0][ch] for ch in common)/v1[2]/v2[2]


list_A = ['بازی']
list_B = comment_box

result7 = []

for key in list_A:
    for word in list_B:
      res = cosdis(word2vec(word), word2vec(key))
      result7.append(res)
      #print("The cosine similarity between : {} and : {} is: {}".format(word, key,

d_f['roleplay_rate'] = result7
```

I Get point of similarity for weited words and build dataframe with that

```
1 d_f
```

	perfect_rate	good_rate	propose_rate	not_rate	film_rate	roleplay_rate	bad_rat
0	0.344124	0.353209	0.433555	0.344124	0.229416	0.382360	0.16222
1	0.718421	0.000000	0.543075	0.179605	0.269408	0.538816	0.12700
2	0.000000	0.408248	0.400892	0.707107	0.000000	0.176777	0.50000
3	0.755929	0.000000	0.428571	0.000000	0.566947	0.566947	0.00000
4	0.328798	0.189832	0.497096	0.328798	0.575396	0.246598	0.23249
...	...	...	...	...	...	...	.
123	0.182574	0.527046	0.207020	0.456435	0.091287	0.365148	0.38729
124	0.447214	0.129099	0.507093	0.335410	0.447214	0.223607	0.15811
125	0.316228	0.547723	0.239046	0.632456	0.158114	0.474342	0.67082
126	0.508913	0.195881	0.544995	0.424094	0.424094	0.466504	0.35985
127	1.000000	0.000000	0.377964	0.000000	0.500000	0.500000	0.00000

128 rows × 7 columns

```
1 d_f['film_name'] = data['film_name']
```

```
1 d_f
```

	perfect_rate	good_rate	propose_rate	not_rate	film_rate	roleplay_rate	bad_rat
0	0.344124	0.353209	0.433555	0.344124	0.229416	0.382360	0.16222

```
1 d_f['comment'] = data['clean_comment']
```

```
1 df = d_f
```

```
1 df.head()
```

	perfect_rate	good_rate	propose_rate	not_rate	film_rate	roleplay_rate	bad_rate
0	0.344124	0.353209	0.433555	0.344124	0.229416	0.382360	0.162221
1	0.718421	0.000000	0.543075	0.179605	0.269408	0.538816	0.127000

```
1 df = df[['comment', 'film_name', 'roleplay_rate', 'film_rate', 'not_rate', 'bad_rate
```

Now We have New DataFrame with comments, film_name label and points of similariy must weighted words for each comments

```
1 df.head(10)
```

	comment	film_name	roleplay_rate	film_rate	not_rate	bad_rate	propose_rate	good_
0	خیلی قشنگه . روزگار الان و خوب به تصویر کشیده	BlackCat	0.382360	0.229416	0.344124	0.162221	0.433555	0.3
1	عالی آقای رادان	BlackCat	0.538816	0.269408	0.179605	0.127000	0.543075	0.0
2	قشنگ بود	BlackCat	0.176777	0.000000	0.707107	0.500000	0.400892	0.4
3	عالیییی	BlackCat	0.566947	0.566947	0.000000	0.000000	0.428571	0.0
4	من دیدم خیلی خفنه باحال بود	BlackCat	0.246598	0.575396	0.328798	0.232495	0.497096	0.1

adding new columns Nmae that sentiment for set sentiment for each coment

```
1 df['sentiment'] = 1.0
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
  """Entry point for launching an IPython kernel.
```

```
1 df.head()
```

	comment	film_name	roleplay_rate	film_rate	not_rate	bad_rate	propose_rate	good_
0	خیلی قشنگه . روزگار الان و خوب به تصویر کشیده	BlackCat	0.382360	0.229416	0.344124	0.162221	0.433555	0.3
1	عالی آقای	BlackCat	0.538816	0.269408	0.179605	0.127000	0.543075	0.0

Now for Set Sentiment labels Calcuting Mean Of rate point

```
1 def rate_mean():
2   x = (df['roleplay_rate'] + df['film_rate'] + df['not_rate'] +
3        df['bad_rate'] + df['propose_rate'] + df['good_rate'] +
4        df['perfect_rate'] )
5   rs = x / 7
6   return rs
```

```
1 df['rate_mean'] = rate_mean()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
  """Entry point for launching an IPython kernel.
```

```
1 df.head()
```

	comment	film_name	roleplay_rate	film_rate	not_rate	bad_rate	propose_rate	good
0	خیلی قشنگه . روزگار الان و خوب به تصویر کشیده	BlackCat	0.382360	0.229416	0.344124	0.162221	0.433555	0.3
1	عالی آقای	BlackCat	0.538816	0.269408	0.179605	0.127000	0.543075	0.0

Set Label

```
1 A = []
2 for i in df['rate_mean']:
3   if i > 0.30:
4     A.append('positive')
5   else:
6     A.append("negative")
7
8 df['sentiment'] = A
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
```

```
1 df
```

	comment	film_name	roleplay_rate	film_rate	not_rate	bad_rate	propose_r
0	خیلی قشنگه . روزگار الان و خوب به تصویر کشیده	BlackCat	0.382360	0.229416	0.344124	0.162221	0.433
1	عالی آقای	BlackCat	0.538816	0.269408	0.179605	0.127000	0.543

```
1 df['sentiment'].unique()
```

```
array(['positive', 'negative'], dtype=object)
```

```
1 import matplotlib.pyplot as plt
2
3 plt.hist(df['sentiment'])
4 plt.show()
```

Conclusion

```
1 df.drop(columns=['roleplay_rate','comment','perfect_rate', 'good_rate', 'film_rate',
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/frame.py:4913: SettingWithCopyWarning
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
  errors=errors,
```

```
1 df.head(10)
```

	film_name	sentiment
0	BlackCat	positive
1	BlackCat	positive
2	BlackCat	positive
3	BlackCat	positive
4	BlackCat	positive
5	BlackCat	positive
6	BlackCat	positive
7	BlackCat	positive
8	BlackCat	positive
9	BlackCat	negative

```
1 df.groupby(by=["sentiment"]).count()
```

sentiment	film_name
negative	10
positive	118

```
1
2 df.groupby(by=["film_name"]).count()
3
```

film_name	sentiment
BlackCat	18
Death_of_Salesman	10
Departed	15
Facing_Mirrors	23
LoserMan	22
NoChoice	20
The_Late_Father	20

```
1 print(df.groupby(by=["sentiment"]).count())
2 print(f'percent of data label for each film: {10 / 128 *100}')
```

```
print(f' percent of data label for each film: {(10 / 128 * 100)}')
```

```
            film_name
sentiment
negative          10
positive         118
percent of data label for each film: 7.8125
```

```
def percent_of_negative_lebel_for_each_film(x):
  perc = 10 /128
  neg_perc = perc * x
  return neg_perc
```

```
print(f' Score of Ngative Sentiment for "BalckCats" film: {format(percent_of_negativ
print(f' Score of Ngative Sentiment for "Death_of_Salesman" film: {format(percent_of
print(f' Score of Ngative Sentiment for "Departed" film: {format(percent_of_negative
print(f' Score of Ngative Sentiment for "Facing_Mirrors" film: {format(percent_of_ne
print(f' Score of Ngative Sentiment for "LoserMan" film: {format(percent_of_negative
print(f' Score of Ngative Sentiment for "NoChoice" film: {format(percent_of_negative
print(f' Score of Ngative Sentiment for "The_Late_Father" film: {format(percent_of_n

```

```
 Score of Ngative Sentiment for "BalckCats" film: 1.41
 Score of Ngative Sentiment for "Death_of_Salesman" film: 0.78
 Score of Ngative Sentiment for "Departed" film: 1.17
 Score of Ngative Sentiment for "Facing_Mirrors" film: 1.80
 Score of Ngative Sentiment for "LoserMan" film: 1.72
 Score of Ngative Sentiment for "NoChoice" film: 1.56
 Score of Ngative Sentiment for "The_Late_Father" film: 1.56
```

```
def percent_of_psitive_lebel_for_each_film(x):
  perc = 118 /128
  pos_perc = perc * x
  return pos_perc
```

```
print(f' Score of Positive Sentiment for "BalckCats" film: {format(percent_of_psitiv
print(f' Score of Positive Sentiment for "Death_of_Salesman " film: {format(percent_
print(f' Score of Positive Sentiment for "Departed" film: {format(percent_of_psitive
print(f' Score of Positive Sentiment for "Facing_Mirrors" film: {format(percent_of_p
print(f' Score of Positive Sentiment for "LoserMan" film: {format(percent_of_psitive
print(f' Score of Positive Sentiment for "NoChoice" film: {format(percent_of_psitive
print(f' Score of Positive Sentiment for "The_Late_Father" film: {format(percent_of_
```

```
 Score of Positive Sentiment for "BalckCats" film: 16.59
 Score of Positive Sentiment for "Death_of_Salesman     " film: 9.22
 Score of Positive Sentiment for "Departed" film: 13.83
 Score of Positive Sentiment for "Facing_Mirrors" film: 21.20
 Score of Positive Sentiment for "LoserMan" film: 20.28
 Score of Positive Sentiment for "NoChoice" film: 18.44
 Score of Positive Sentiment for "The_Late_Father" film: 18.44
```

Colab paid products - Cancel contracts here

✓ 0s completed at 5:58 AM