In [397…
```
!pip install contractions
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: contractions in /usr/local/lib/python3.7/dist-packages (0.1.72)
Requirement already satisfied: textsearch>=0.0.21 in /usr/local/lib/python3.7/dist-packages (from contractions) (0.0.24)
Requirement already satisfied: pyahocorasick in /usr/local/lib/python3.7/dist-packages (from textsearch>=0.0.21->contractions) (1.4.4)
Requirement already satisfied: anyascii in /usr/local/lib/python3.7/dist-packages (from textsearch>=0.0.21->contractions) (0.3.1)
```

In [398…
```
! pip install bs4 # in case you don't have it installed

# Dataset: https://s3.amazonaws.com/amazon-reviews-pds/tsv/amazon_reviews_us_Jewelry_v
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: bs4 in /usr/local/lib/python3.7/dist-packages (0.0.1)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.7/dist-packages (from bs4) (4.6.3)
```

In [399…
```
!pip install scikit-learn
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages (1.0.2)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from scikit-learn) (1.1.0)
Requirement already satisfied: numpy>=1.14.6 in /usr/local/lib/python3.7/dist-packages (from scikit-learn) (1.21.6)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from scikit-learn) (3.1.0)
Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.7/dist-packages (from scikit-learn) (1.7.3)
```

In [400…
```
import pandas as pd
import numpy as np
import nltk
nltk.download('wordnet')
import re
from bs4 import BeautifulSoup
import contractions
from nltk.corpus import stopwords
nltk.download('stopwords')
from nltk.stem import WordNetLemmatizer
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

Out[400]:
```
True
```

In [401…
```
!python --version
```

```
Python 3.7.13
```

In [402…
```
#Python 3.7.13
```

Read Data

In [403…
```
df = pd.read_csv("/content/data.tsv",
                 usecols = ['star_rating','review_body'], sep='\t', on_bad_lines='ski
```

Keep Reviews and Ratings

In [404…
```
df
```

Out[404]:

	star_rating	review_body
0	5.0	so beautiful even tho clearly not high end .....
1	5.0	Great product.. I got this set for my mother, ...
2	5.0	Exactly as pictured and my daughter's friend l...
3	5.0	Love it. Fits great. Super comfortable and nea...
4	5.0	Got this as a Mother's Day gift for my Mom and...
...	...	...
1054299	5.0	These are just want I wanted, nothing flashy. ...
1054300	4.0	I purchased it as a gift and my family member ...
1054301	1.0	I am SO TIRED of fraudulent claims in Product ...
1054302	4.0	For $1.99 this is a great little buy. Very shi...
1054303	5.0	Pretti

1054304 rows × 2 columns

Shuffling the rows

In [405…
```
df = df.sample(frac=1).reset_index(drop=True)
df
```

Out[405]:

	star_rating	review_body
0	5.0	We have a tough time finding earrings that wil...
1	4.0	Understated but very sparkly and it definitely...
2	2.0	Nice looking necklace but arrived with a broke...
3	5.0	Very nice, simple gold lariat. Love it! Would ...
4	4.0	The ring was big on me but I loved the way it ...
...	...	...
1054299	5.0	Beautifull pendant ... love the contrast and ...
1054300	5.0	Bought one from kohls on sale for 60% off and ...
1054301	2.0	Fair warning they're huge. I didn't realize ho...
1054302	5.0	Beautiful piece
1054303	5.0	I really love this necklace. When I first got ...

1054304 rows × 2 columns

Removing rows with null values

In [406...
```
df.isnull().sum()
```

Out[406]:
```
star_rating      1
review_body    235
dtype: int64
```

In [407...
```
df.dropna(inplace=True)
```

In [408...
```
df.isnull().sum()
```

Out[408]:
```
star_rating    0
review_body    0
dtype: int64
```

In [409...
```
df.star_rating.unique()
```

Out[409]:
```
array([5., 4., 2., 3., 1.])
```

We select 20000 reviews randomly from each rating class.

In [410...
```
df.star_rating.value_counts()
```

Out[410]:
```
5.0    658539
4.0    151134
1.0     94668
3.0     91655
2.0     58073
Name: star_rating, dtype: int64
```

In [411…
```
df.dtypes
```

Out[411]:
```
star_rating     float64
review_body      object
dtype: object
```

In [412…
```
df['star_rating'] = df['star_rating'].astype(int)
df['review_body'] = df['review_body'].astype('string')
```

In [413…
```
df.dtypes
```

Out[413]:
```
star_rating      int64
review_body     string
dtype: object
```

In [414…
```
df.star_rating.value_counts()
```

Out[414]:
```
5    658539
4    151134
1     94668
3     91655
2     58073
Name: star_rating, dtype: int64
```

In [415…
```
s1 = df[df.star_rating.eq(1)].sample(20000)
s2 = df[df.star_rating.eq(2)].sample(20000)
s3 = df[df.star_rating.eq(3)].sample(20000)
s4 = df[df.star_rating.eq(4)].sample(20000)
s5 = df[df.star_rating.eq(5)].sample(20000)
```

In [416…
```
newdf = pd.concat([s1,s2,s3,s4,s5], ignore_index=True)
newdf
```

Out[416]:

	star_rating	review_body
0	1	so disappointed the backs for the earnings whe...
1	1	I ordered 2 of these and got 2 1mm rings inste...
2	1	The owl pendant was beautiful for being so ine...
3	1	Arrived broken with no possible way to fix it.
4	1	It is a very pretty ring but I am sooo very di...
...	...	...
99995	5	beautiful.
99996	5	Quick shipment. And has been great use it whil...
99997	5	They are perfect for the little girls I bought...
99998	5	The item does what it is supposed to do and it...
99999	5	Beautiful

100000 rows × 2 columns

Shuffling the rows

In [417…]
```
newdf = newdf.sample(frac=1).reset_index(drop=True)
newdf
```

Out[417]:

	star_rating	review_body
0	3	Absolutely beautiful!! Great shine looks like ...
1	5	Love it!!! So does my husband cause we both lo...
2	5	Great, she loves them!
3	1	Inscription on this bracelet is so tiny you al...
4	3	Adds style to my boring suits
...	...	...
99995	5	Beautiful ring, more subtle than it looks from...
99996	5	Absolutely loves this! She couldn't stop thank...
99997	3	I liked this, but it was very wide on my wrist...
99998	3	From the front, this set looks amazing. Once y...
99999	2	Very small and not nearly as sparkling as the ...

100000 rows × 2 columns

In [418…]
```
newdf.star_rating.value_counts()
```

Out[418]:
```
3    20000
5    20000
1    20000
4    20000
2    20000
Name: star_rating, dtype: int64
```

In [419…]
```
newdf.isnull().sum()
```

Out[419]:
```
star_rating    0
review_body    0
dtype: int64
```

In [420…]
```
# length= newdf["review_body"].str.len()
# length
```

In [421…]
```
# length.sum()/100000
```

In [422…]
```
before = newdf['review_body'].str.len().mean()
```

Data Cleaning

Converting all the reviews to lower case

In [423…]
```
newdf['review_body'] = newdf['review_body'].str.lower()
```

In [424…]
```
newdf
```

Out[424]:

	star_rating	review_body
0	3	absolutely beautiful!! great shine looks like ...
1	5	love it!!! so does my husband cause we both lo...
2	5	great, she loves them!
3	1	inscription on this bracelet is so tiny you al...
4	3	adds style to my boring suits
...	...	...
99995	5	beautiful ring, more subtle than it looks from...
99996	5	absolutely loves this! she couldn't stop thank...
99997	3	i liked this, but it was very wide on my wrist...
99998	3	from the front, this set looks amazing. once y...
99999	2	very small and not nearly as sparkling as the ...

100000 rows × 2 columns

In [425…]
```
# regex = re.compile('[^a-zA-Z ]')
# #First parameter is the replacement, second parameter is your input string
# regex.sub('', 'ab3  d*E \n g')
```

In [426…]
```
#re.sub(' +', ' ',  '       T     he     ,     quick  bro       wn       fox      ')
```

In [427…]
```
def clean(row):


    soup = BeautifulSoup(row.review_body, "html.parser")

    #this extracts all the text from the document and removes html tags
    text1 = soup.get_text(' ')

    #removing any urls
    text2 = re.sub(r'http\S+', '', text1)
    #ftext = re.sub(r'^https?:\/\/.*[\r\n]*', '', text)

    #performing contractions
    text3 = contractions.fix(text2)

    #removing non-alphabetic characters
    regex = re.compile('[^a-zA-Z ]')
    text4 = regex.sub('', text3)


    #removing extra white spaces
    text5 = re.sub(' +', ' ',  text4)

    row.review_body = text5

    return row
```

In [428…]
```
newdf = newdf.apply(clean, axis='columns')
```

```
/usr/local/lib/python3.7/dist-packages/bs4/__init__.py:336: UserWarning: "https://www.amazon.com/review/review-your-purchases/ref=oh_aui_rev_shipment_o00_s00?_encoding=utf8&asins=b01444jlxe&channel=yacc-wr#" looks like a URL. Beautiful Soup is not an HTTP client. You should probably use an HTTP client like requests to get the document behind the URL, and feed that document to Beautiful Soup.
  ' that document to Beautiful Soup.' % decoded_markup
/usr/local/lib/python3.7/dist-packages/bs4/__init__.py:336: UserWarning: "https://www.amazon.com/review/review-your-purchases/ref=pe_259730_144232530_cm_1_img?_encoding=utf8&asins=b00mand1yw&channel=ec_sft&crauthtoken=gbyfce0y80lijky6eqjb2apfm2iizb3lihevj6oaaaadaaaaafw1ffnyyxcaaaaa&customerid=a34lf5jz8ht63a#" looks like a URL. Beautiful Soup is not an HTTP client. You should probably use an HTTP client like requests to get the document behind the URL, and feed that document to Beautiful Soup.
  ' that document to Beautiful Soup.' % decoded_markup
```

In [429…]
```
newdf
```

Out[429]:

	star_rating	review_body
0	3	absolutely beautiful great shine looks like yo...
1	5	love it so does my husband because we both lov...
2	5	great she loves them
3	1	inscription on this bracelet is so tiny you al...
4	3	adds style to my boring suits
...	...	...
99995	5	beautiful ring more subtle than it looks from ...
99996	5	absolutely loves this she could not stop thank...
99997	3	i liked this but it was very wide on my wrist ...
99998	3	from the front this set looks amazing once you...
99999	2	very small and not nearly as sparkling as the ...

100000 rows × 2 columns

In [430…]
```
newdf.isnull().sum()
```

Out[430]:
```
star_rating    0
review_body    0
dtype: int64
```

In [431…]
```
after = newdf['review_body'].str.len().mean()
```

In [432…]
```
newdf.dropna(inplace=True)
```

In [433…]
```
print(before,',',after)
```

```
140.20527 , 135.11932
```

Pre-processing

remove the stop words

In [434…]
```
stop = stopwords.words('english')
```

In [435…]
```
newdf['review_body'] = newdf['review_body'].apply(lambda x: ' '.join([word for word ir

#newdf['review_body'] = newdf['review_body'].apply(lambda x: [item for item in x.split
```

In [436…]
```
newdf
```

Out[436]:

	star_rating	review_body
0	3	absolutely beautiful great shine looks like we...
1	5	love husband love guns
2	5	great loves
3	1	inscription bracelet tiny almost need magnifyi...
4	3	adds style boring suits
...	...	...
99995	5	beautiful ring subtle looks photo website much...
99996	5	absolutely loves could stop thanking enough wo...
99997	3	liked wide wrist probably appropriate summer w...
99998	3	front set looks amazing flip around see cheapl...
99999	2	small nearly sparkling photo highquality gift ...

100000 rows × 2 columns

In [437…]
```
newdf.isnull().sum()
```

Out[437]:
```
star_rating    0
review_body    0
dtype: int64
```

In [438…]
```
newdf.dropna(inplace=True)
```

In [439…]
```
#newdf['review_body'].str.len().mean()
```

perform lemmatization

In [440…]
```
lemmatizer = WordNetLemmatizer()
```

In [441…]
```
#newdf['review_body'] = newdf['review_body'].apply(lambda x: ' '.join([lemmatizer.lemm
newdf['review_body'] = newdf['review_body'].apply(lambda x: ' '.join([lemmatizer.lemma
```

In [442…]
```
newdf
```

Out[442]:

	star_rating	review_body
0	3	absolutely beautiful great shine look like wea...
1	5	love husband love gun
2	5	great love
3	1	inscription bracelet tiny almost need magnifyi...
4	3	add style boring suit
...	...	...
99995	5	beautiful ring subtle look photo website much ...
99996	5	absolutely love could stop thanking enough wor...
99997	3	liked wide wrist probably appropriate summer w...
99998	3	front set look amazing flip around see cheaply...
99999	2	small nearly sparkling photo highquality gift ...

100000 rows × 2 columns

In [443...]
```
later = newdf['review_body'].str.len().mean()
```

In [444...]
```
#newdf.isnull().sum()
```

In [445...]
```
print(after,',',later)
```

```
135.11932 , 80.74768
```

In [446...]
```
# c=0
# for i in newdf['review_body']:
#   if len(i.split())<3:
#     c+=1
# print(c)
```

TF-IDF Feature Extraction

In [447...]
```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(newdf['review_body'])
```

In [448...]
```
X.shape
```

Out[448]:
```
(100000, 29610)
```

In [449...]
```
#X_features = pd.DataFrame(X.toarray())
```

Train-Test Split

In [450…
```
from sklearn.model_selection import train_test_split
```

In [451…
```
X_train, X_test, y_train, y_test = train_test_split(X, newdf['star_rating'], test_size
```

In [452…
```
y_train.value_counts()
```

Out[452]:
```
5    16058
2    16020
4    16015
1    15962
3    15945
Name: star_rating, dtype: int64
```

In [453…
```
y_test.value_counts()
```

Out[453]:
```
3    4055
1    4038
4    3985
2    3980
5    3942
Name: star_rating, dtype: int64
```

Importing metrics to evaluate

In [454…
```
from sklearn.metrics import f1_score, precision_score, recall_score, confusion_matrix,
from sklearn.metrics import precision_recall_fscore_support
```

Perceptron

In [455…
```
from sklearn.linear_model import Perceptron
```

In [456…
```
p = Perceptron()
p.fit(X_train,y_train)
```

Out[456]:
```
Perceptron()
```

In [457…
```
predictions = p.predict(X_test)
```

In [458…
```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           1       0.45      0.52      0.48      4038
           2       0.30      0.26      0.28      3980
           3       0.31      0.37      0.34      4055
           4       0.33      0.24      0.28      3985
           5       0.52      0.55      0.54      3942

    accuracy                           0.39     20000
   macro avg       0.38      0.39      0.38     20000
weighted avg       0.38      0.39      0.38     20000
```

In [459…
```
metrics = precision_recall_fscore_support(y_test, predictions)
```

Printing precision, recall, f1_score and their average

In [460…
```
for i in range(5):
    print(metrics[0][i] , ',' , metrics[1][i] , ',' , metrics[2][i])

avg = np.mean(metrics[:3], axis=1)
print(avg[0], ',' ,avg[1], ',' ,avg[2])
```

```
0.45156418554476807 , 0.5183259039128282 , 0.4826472962066182
0.30274963820549927 , 0.2628140703517588 , 0.2813718897108272
0.31238175320580197 , 0.36646115906288534 , 0.33726736268724467
0.3298162014976174 , 0.24316185696361356 , 0.27993644373826376
0.5183867141162515 , 0.554287163876205 , 0.5357361775162438
0.3829796985139876 , 0.3890100308334582 , 0.3833918339718395
```

SVM

In [461…
```
from sklearn.svm import LinearSVC
svm = LinearSVC()
```

In [462…
```
svm.fit(X_train,y_train)
```

Out[462]:
```
LinearSVC()
```

In [463…
```
predictions = svm.predict(X_test)
```

In [464…
```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           1       0.54      0.65      0.59      4038
           2       0.37      0.32      0.35      3980
           3       0.40      0.32      0.36      4055
           4       0.43      0.40      0.41      3985
           5       0.59      0.72      0.65      3942

    accuracy                           0.48     20000
   macro avg       0.47      0.48      0.47     20000
weighted avg       0.47      0.48      0.47     20000
```

In [465…
```
metrics = precision_recall_fscore_support(y_test, predictions)
```

Printing precision, recall, f1_score and their average

In [466…
```
for i in range(5):
    print(metrics[0][i] , ',' , metrics[1][i] , ',' , metrics[2][i])

avg = np.mean(metrics[:3], axis=1)
print(avg[0], ',' ,avg[1], ',' ,avg[2])
```

```
0.542263759086189 , 0.6466072313026251 , 0.5898565458036823
0.37281722933643774 , 0.32185929648241207 , 0.3454692556634304
0.3991507430997877 , 0.324537607891492 , 0.3579978237214364
0.4311604003245875 , 0.4 , 0.41499609476698773
0.5936908517350158 , 0.7161339421613394 , 0.6491893756467747
0.4678165967164036 , 0.4818276155675737 , 0.47150181912046224
```

Logistic Regression

In [467…]
```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
```

In [468…]
```
lr.fit(X_train,y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: Converg
enceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
```

Out[468]:
```
LogisticRegression()
```

In [469…]
```
predictions = lr.predict(X_test)
```

In [470…]
```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           1       0.57      0.66      0.61      4038
           2       0.41      0.35      0.37      3980
           3       0.42      0.38      0.40      4055
           4       0.45      0.44      0.45      3985
           5       0.62      0.69      0.66      3942

    accuracy                           0.50     20000
   macro avg       0.49      0.50      0.50     20000
weighted avg       0.49      0.50      0.50     20000
```

In [471…]
```
metrics = precision_recall_fscore_support(y_test, predictions)
```

Printing precision, recall, f1_score and their average

In [472…]
```
for i in range(5):
    print(metrics[0][i] , ',' , metrics[1][i] , ',' , metrics[2][i])

avg = np.mean(metrics[:3], axis=1)
print(avg[0], ',' ,avg[1], ',' ,avg[2])
```

```
0.5655216284987278 , 0.6604754829123328 , 0.6093214530500343
0.4061855670103093 , 0.3464824120603015 , 0.37396610169491523
0.4157150625339859 , 0.37706535141800246 , 0.395448079658606
0.4548783677739995 , 0.43638644918444164 , 0.4454405737704918
0.6235186873290793 , 0.6940639269406392 , 0.6569027611044418
0.49316386262922035 , 0.5028947245031435 , 0.4962157938556978
```

Naive Bayes

In [473…]
```
from sklearn.naive_bayes import MultinomialNB
nb = MultinomialNB()
```

In [474…]
```
nb.fit(X_train, y_train)
```

Out[474]:
```
MultinomialNB()
```

In [475…]
```
predictions = nb.predict(X_test)
```

In [476…]
```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           1       0.59      0.60      0.59      4038
           2       0.38      0.36      0.37      3980
           3       0.40      0.39      0.40      4055
           4       0.43      0.42      0.43      3985
           5       0.63      0.67      0.65      3942

    accuracy                           0.49     20000
   macro avg       0.49      0.49      0.49     20000
weighted avg       0.48      0.49      0.49     20000
```

In [477…]
```
metrics = precision_recall_fscore_support(y_test, predictions)
```

Printing precision, recall, f1_score and their average

In [478…]
```
for i in range(5):
    print(metrics[0][i] , ',' , metrics[1][i] , ',' , metrics[2][i])

avg = np.mean(metrics[:3], axis=1)
print(avg[0], ',' ,avg[1], ',' ,avg[2])
```

```
0.5858463035019456 , 0.5965824665676077 , 0.5911656441717792
0.37966368891224384 , 0.3630653266331658 , 0.37117903930131
0.39914486921529174 , 0.39136868064118374 , 0.39521852820321257
0.43473765830964073 , 0.42208281053952323 , 0.42831678125795775
0.6263865942884116 , 0.6732623033992897 , 0.6489790927986306
0.48515582284550673 , 0.4892723175561541 , 0.486971817146578
```

In [478…]
```
```