Imports

In [ ]:
```
import pandas as pd
import numpy as np
import nltk
nltk.download('wordnet')
import re
from bs4 import BeautifulSoup

import gzip
import contractions
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split, GridSearchCV
import string
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import Perceptron
from sklearn.metrics import classification_report
from sklearn.svm import LinearSVC
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import precision_score, recall_score, f1_score
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\YASH\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Read Data

For preparing the data, I am using the gzip package to open and read the dataset. The datasets consists of 15 columns of which I am extracting the 'review_body' and 'rating' column for this assignment. The dataframe 'df_review_rating' holds these extracted columns. I am converting the 'ratings' to a standard format which I am then using to create our binary classes. For simplicity, I have created a copy of the 'df_review_rating' called 'binary_df' which has an extra column called as 'classes'. This columns holds the labels for our dataset. Finally, I extracted 50000 reviews randomly from each class and stored it in the 'dataset_df' dataset

In [ ]:
```
dataset_path = 'amazon_reviews_us_Office_Products_v1_00.tsv.gz'
with gzip.open(dataset_path, 'rt', encoding='utf-8') as file:
    df = pd.read_csv(file, sep='\t', on_bad_lines='skip', low_memory=False)
```

In [ ]:
```
print(df.columns.values)
```

```
['marketplace' 'customer_id' 'review_id' 'product_id' 'product_parent'
 'product_title' 'product_category' 'star_rating' 'helpful_votes'
 'total_votes' 'vine' 'verified_purchase' 'review_headline' 'review_body'
 'review_date']
```

In [ ]:
```
df.head(5)
```

Out[ ]:

	marketplace	customer_id	review_id	product_id	product_parent	produ
0	US	43081963	R18RVCKGH1SSI9	B001BM2MAC	307809868	Cushio 7 Inche
1	US	10951564	R3L4L6LW1PUOFY	B00DZYEXPQ	75004341	D Com Gas P
2	US	21143145	R2J8AWXWTDX2TF	B00RTMUHDW	529689027	Stand At Taggi
3	US	52782374	R1PR37BR7G3M6A	B00D7H8XB6	868449945	Amazo 1 High-S Micr
4	US	24045652	R3BDDDZMZBZDPU	B001XCWP34	33521401	D (Inkte P

Keep Reviews and Ratings

In [ ]:
```
df_review_rating = df[['star_rating','review_body']]
df_review_rating
```

Out[ ]:

	star_rating	review_body
0	5	Great product.
1	5	What's to say about this commodity item except...
2	5	Haven't used yet, but I am sure I will like it.
3	1	Although this was labeled as "new" the...
4	4	Gorgeous colors and easy to use
...	...	...
2640249	4	I can't live anymore whithout my Palm III. But...
2640250	4	Although the Palm Pilot is thin and compact it...
2640251	4	This book had a lot of great content without b...
2640252	5	I am teaching a course in Excel and am using t...
2640253	5	A very comprehensive layout of exactly how Vis...

2640254 rows × 2 columns

In [ ]:
```
df_review_rating['star_rating']=pd.to_numeric(df_review_rating['star_rating'], e
df_review_rating = df_review_rating[pd.notna(df_review_rating['star_rating'])]

df_review_rating.head(5)
```

```
C:\Users\YASH\AppData\Local\Temp\ipykernel_24704\2485622692.py:1: SettingWithCopy
Warning:
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/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_review_rating['star_rating']=pd.to_numeric(df_review_rating['star_rating'],
errors='coerce')
```

Out[ ]:

	star_rating	review_body
0	5.0	Great product.
1	5.0	What's to say about this commodity item except...
2	5.0	Haven't used yet, but I am sure I will like it.
3	1.0	Although this was labeled as "new" the...
4	4.0	Gorgeous colors and easy to use

We form two classes and select 50000 reviews randomly from each class.

In [ ]:
```
df_review_rating['star_rating'].unique()
```

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

In [ ]:
```
binary_df = df_review_rating.copy()

def category(row):
    if row['star_rating'] == 1 or row['star_rating'] == '1' or row['star_rating'
        return 1

    else:
        return 2
```

In [ ]:
```
binary_df['class'] = df.apply(lambda row: category(row), axis=1)
binary_df
```

Out[ ]:

	star_rating	review_body	class
0	5.0	Great product.	2
1	5.0	What's to say about this commodity item except...	2
2	5.0	Haven't used yet, but I am sure I will like it.	2
3	1.0	Although this was labeled as "new" the...	1
4	4.0	Gorgeous colors and easy to use	2
...	...	...	...
2640249	4.0	I can't live anymore whithout my Palm III. But...	2
2640250	4.0	Although the Palm Pilot is thin and compact it...	2
2640251	4.0	This book had a lot of great content without b...	2
2640252	5.0	I am teaching a course in Excel and am using t...	2
2640253	5.0	A very comprehensive layout of exactly how Vis...	2

2640237 rows × 3 columns

In [ ]:
```
binary_df['class'].value_counts()
```

Out[ ]:
```
2    2001183
1     639054
Name: class, dtype: int64
```

In [ ]:
```
binary_df['class'].unique()
```

Out[ ]:
```
array([2, 1], dtype=int64)
```

In [ ]:
```
# Reading rows belonging to classes 1 and 2
class1_df = binary_df[binary_df['class'] == 1]
class2_df = binary_df[binary_df['class'] == 2]

# Randomly choosing 50,000 reviews of each class
random_class1_df = class1_df.sample(n = 50000, random_state=42)
random_class2_df = class2_df.sample(n = 50000, random_state=42)

# Combining the two classes to create a single dataset
dataset_df = pd.concat([random_class1_df, random_class2_df])

# Reset the indexes
```

```
dataset_df.reset_index(drop=True, inplace=True)

dataset_df
```

Out[ ]:

	star_rating	review_body	class
0	1.0	it says right in the specifications that it wo...	1
1	1.0	These things were horrible. Used 6 to hang a ...	1
2	2.0	Not sure why, but works for one Epson printer ...	1
3	1.0	I ordered two letter organizers and received t...	1
4	3.0	This kit comes with 5 gears, 4 of which go int...	1
...	...	...	...
99995	5.0	I BUY THIS EVERY YEAR.	2
99996	5.0	Good shredder at a good price.	2
99997	5.0	We have actually had this system for almost 5 ...	2
99998	5.0	Awesome!! Work and print just like the origina...	2
99999	5.0	works well.	2

100000 rows × 3 columns

In [ ]:
```
dataset_df['class'].astype(int)
dataset_df['class'].value_counts()
```

Out[ ]:
```
1    50000
2    50000
Name: class, dtype: int64
```

Data Cleaning

The following tasks were performed for cleaning the dataset -

1) Firstly, I looked for rows with missed values and replaced it with an empty string.

2) Converted all the reviews to lowercase using the lower() function.

3) Removed punctuations from the review by using the string.punctuation package.

4) Removed any kind of non-alphabetical characters from the reviews by tokenizing the words and checking if each character is between A-Z or a-z.

5) Removed all HTML tags and URLs from the reviews.

6) Removed any use of emojis in the reviews.

7) Finally, removed all the extra spaces from the reviews.

I performed contractions on the reviews as well. However, I got slightly better results when contractions was avoided.

Pre-processing

Removing empty reviews

In [ ]:
```
print(dataset_df.isnull().values.any())
print(dataset_df.isnull().sum())

dataset_df = dataset_df.fillna('')
```

```
True
star_rating    0
review_body    1
class          0
dtype: int64
```

Storing average length of the reviews in terms of character length in your dataset before cleaning

In [ ]:
```
reviewLen = pd.DataFrame()
reviewLen['before'] = dataset_df['review_body'].str.len()
print(reviewLen.head(5))
```

```
   before
0     174
1     335
2     109
3     355
4     427
```

Converting reviews into lowercase

In [ ]:
```
dataset_df['review_body'] = dataset_df['review_body'].str.lower()
reviewLen['lowercase'] = dataset_df['review_body'].str.len()
print(dataset_df.head(5))
```

```
   star_rating                                        review_body  class
0          1.0  it says right in the specifications that it wo...      1
1          1.0  these things were horrible.  used 6 to hang a ...      1
2          2.0  not sure why, but works for one epson printer ...      1
3          1.0  i ordered two letter organizers and received t...      1
4          3.0  this kit comes with 5 gears, 4 of which go int...      1
```

Remove Punctuations

In [ ]:
```
def remove_punctuations(text):
    if isinstance(text, str):
        return ''.join(char for char in text if char not in string.punctuation)
    else:
        return text

dataset_df['review_body'] = dataset_df['review_body'].apply(remove_punctuations)
```

```
reviewLen['punctations'] = dataset_df['review_body'].str.len()

print(dataset_df.head(5))
```

```
   star_rating                                        review_body  class
0          1.0  it says right in the specifications that it wo...      1
1          1.0  these things were horrible  used 6 to hang a s...      1
2          2.0  not sure why but works for one epson printer b...      1
3          1.0  i ordered two letter organizers and received t...      1
4          3.0  this kit comes with 5 gears 4 of which go into...      1
```

Remove non-alphabetical characters

In [ ]:
```
def remove_non_alphabetical(text):
    if isinstance(text, str):
        return re.sub(r'[^a-zA-Z]', ' ', text)
    else:
        return text

dataset_df['review_body'] = dataset_df['review_body'].apply(remove_non_alphabeti
reviewLen['non_alphanum'] = dataset_df['review_body'].str.len()

print(dataset_df.head(5))
```

```
   star_rating                                        review_body  class
0          1.0  it says right in the specifications that it wo...      1
1          1.0  these things were horrible  used   to hang a s...      1
2          2.0  not sure why but works for one epson printer b...      1
3          1.0  i ordered two letter organizers and received t...      1
4          3.0  this kit comes with   gears   of which go into...      1
```

Remove HTML and URLs from the reviews

In [ ]:
```
dataset_df['review_body'] = dataset_df['review_body'].astype(str)

def remove_html_tags(text):
    try:
        clean_text = re.sub(r'<.*?>', '', text)
        return clean_text
    except TypeError:
        return text

def remove_urls(text):
    try:
        clean_text = re.sub(r'http\S+', '', text)
        return clean_text
    except TypeError:
        return text

dataset_df['review_body'] = dataset_df['review_body'].apply(remove_html_tags)
dataset_df['review_body'] = dataset_df['review_body'].apply(remove_urls)

reviewLen['HTML_URLs'] = dataset_df['review_body'].str.len()

print(dataset_df.head(5))
```

	star_rating	review_body	class
0	1.0	it says right in the specifications that it wo...	1
1	1.0	these things were horrible used to hang a s...	1
2	2.0	not sure why but works for one epson printer b...	1
3	1.0	i ordered two letter organizers and received t...	1
4	3.0	this kit comes with gears of which go into...	1

Remove Emojis

In [ ]:
```
def remove_emojis(text):
    emoji_pattern = re.compile("["
                           u"\U0001F600-\U0001F64F"  # emoticons
                           u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                           u"\U0001F680-\U0001F6FF"  # transport & map symbols
                           u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                           u"\U00002702-\U000027B0"
                           u"\U000024C2-\U0001F251"
                           "]+", flags=re.UNICODE)

    return emoji_pattern.sub(r'', text)

dataset_df['review_body'] = dataset_df['review_body'].apply(remove_emojis)
reviewLen['Emojis'] = dataset_df['review_body'].str.len()

print(dataset_df.head(5))
```

	star_rating	review_body	class
0	1.0	it says right in the specifications that it wo...	1
1	1.0	these things were horrible used to hang a s...	1
2	2.0	not sure why but works for one epson printer b...	1
3	1.0	i ordered two letter organizers and received t...	1
4	3.0	this kit comes with gears of which go into...	1

Remove extra spaces

In [ ]:
```
dataset_df['review_body'] = dataset_df['review_body'].str.replace(r'\s+', ' ', r
reviewLen['extraSpaces'] = dataset_df['review_body'].str.len()

print(dataset_df.head(5))
```

	star_rating	review_body	class
0	1.0	it says right in the specifications that it wo...	1
1	1.0	these things were horrible used to hang a s...	1
2	2.0	not sure why but works for one epson printer b...	1
3	1.0	i ordered two letter organizers and received t...	1
4	3.0	this kit comes with gears of which go into...	1

Contractions on the reviews

In [ ]:
```
# def expand_contractions(text):
#     return contractions.fix(text)

# dataset_df['review_body'] = dataset_df['review_body'].apply(expand_contraction
# reviewLen['contractions'] = dataset_df['review_body'].str.len()

# print(dataset_df.head(5))
```

Average length of the reviews in terms of character length in your dataset before and after cleaning

In [ ]:
```
reviewLen['after'] = dataset_df['review_body'].str.len()

print("The average length of the reviews before clearning - ",reviewLen['before'
print("The average length of the reviews after cleaning - ", reviewLen['after'].
```

```
The average length of the reviews before clearning -  316.02996
The average length of the reviews after cleaning -  304.46909
```

Using NLKT package to remove stopwords from the reviews and perform lemmatization on it.

Remove the stop words

In [ ]:
```
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
dataset_df['review_body'] = dataset_df['review_body'].apply(lambda x: ' '.join([


print(dataset_df.head(5))
```

```
   star_rating                                        review_body  class
0          1.0  says right specifications works iphone lifepro...      1
1          1.0  things horrible used hang small unframed canva...      1
2          2.0      sure works one epson printer supposed use ink      1
3          1.0  ordered two letter organizers received promptl...      1
4          3.0  kit comes gears go printer big tan one goes fu...      1
```

Perform lemmatization

In [ ]:
```
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

def lemmatize_text(text):
    words = nltk.word_tokenize(text)
    filtered_words = [lemmatizer.lemmatize(word) for word in words]
    filtered_text = ' '.join(filtered_words)
    return filtered_text

dataset_df['review_body'] = dataset_df['review_body'].apply(lemmatize_text)
reviewLen['lemmatization'] = dataset_df['review_body'].str.len()

print(dataset_df.head(5))
```

```
   star_rating                                        review_body  class
0          1.0  say right specification work iphone lifeproof ...      1
1          1.0  thing horrible used hang small unframed canvas...      1
2          2.0       sure work one epson printer supposed use ink      1
3          1.0  ordered two letter organizer received promptly...      1
4          3.0  kit come gear go printer big tan one go fuser ...      1
```

Average length of the reviews in terms of character length in your dataset before and after preprocessing

In [ ]:
```
print("The average length of the reviews after cleaning - ", reviewLen['after'].
print("The average length of the reviews in terms of character length after prep
```

```
The average length of the reviews after cleaning -  304.46909
The average length of the reviews in terms of character length after preprocessin
g -  190.75987
```

Using sklearn to divide our dataset into train and test and finally perform TF-IDF and BOW feature extraction on the training and testing datasets.

TF-IDF and BoW Feature Extraction

Dataset Splitting

In [ ]:
```
Xtrain, Xtest, Ytrain, Ytest = train_test_split(dataset_df['review_body'], datas

print("X_train shape:", Xtrain.shape)
print("X_test shape:", Xtest.shape)
print("y_train shape:", Ytrain.shape)
print("y_test shape:", Ytest.shape)
```

```
X_train shape: (80000,)
X_test shape: (20000,)
y_train shape: (80000,)
y_test shape: (20000,)
```

TF-IDF

In [ ]:
```
tf_idf = TfidfVectorizer()
Xtrain_tf_idf = tf_idf.fit_transform(Xtrain)
Xtest_tf_idf = tf_idf.transform(Xtest)
```

BOW

In [ ]:
```
bowVectorizer = CountVectorizer()
Xtrain_BOW = bowVectorizer.fit_transform(Xtrain)
Xtest_BOW = bowVectorizer.transform(Xtest)
```

Generate Scores

In [ ]:
```
def get_stats(Ytest, pred):
    precision = precision_score(Ytest, pred)
    recall = recall_score(Ytest, pred)
    f1 = f1_score(Ytest, pred)

    return precision, recall, f1
```

Although not included, I have made use of GridSearchCV to determine the best parameters for training our models.

Perceptron Using Both Features

TF-IDF

In [ ]:
```
# Initialize the Perceptron model
model_perceptron = Perceptron(alpha=0.001, max_iter=1000)

model_perceptron = model_perceptron.fit(Xtrain_tf_idf, Ytrain)
predPerceptron = model_perceptron.predict(Xtest_tf_idf)

precision_tfidf, recall_tfidf, f1_tfidf = get_stats(Ytest, predPerceptron)
```

BOW

In [ ]:
```
# Initialize the Perceptron model
model_perceptron = Perceptron(tol=1e-03)

model_perceptron = model_perceptron.fit(Xtrain_BOW, Ytrain)
predPerceptron = model_perceptron.predict(Xtest_BOW)

precision_bow, recall_bow, f1_bow = get_stats(Ytest, predPerceptron)
```

In [ ]:
```
print(f"TF-IDF FOR PERCEPTRON - {precision_tfidf}, {recall_tfidf}, {f1_tfidf}")
print(f"BOW FOR PERCEPTRON - {precision_bow}, {recall_bow}, {f1_bow}")
```

```
TF-IDF FOR PERCEPTRON - 0.7814747339150446, 0.8121574489287494, 0.796520719311962
4
BOW FOR PERCEPTRON - 0.7933797577029477, 0.8287992027902342, 0.8107027975436204
```

SVM Using Both Features

TF-IDF

In [ ]:
```
model_svm = LinearSVC(C=0.35,
    tol=0.001,
    max_iter=1000,                  #Total iterations
    random_state=16,                 #Control the random number generation to con
    penalty='l1',                   #Norm of Penalty
    class_weight="balanced",        #Provides the weight to each class
    loss='squared_hinge',           #Specifies the Loss Function
    dual=False
)

model_svm = model_svm.fit(Xtrain_tf_idf , Ytrain)
predSVM = model_svm.predict(Xtest_tf_idf)

precision_tfidf, recall_tfidf, f1_tfidf = get_stats(Ytest, predSVM)
```

BOW

In [ ]:
```
model_svm = LinearSVC(C=0.35,
    tol=0.001,
    max_iter=1000,                  #Total iterations
    random_state=16,                 #Control the random number generation to con
    penalty='l1',                   #Norm of Penalty
    class_weight="balanced",        #Provides the weight to each class
    loss='squared_hinge',           #Specifies the Loss Function
    dual=False
)

model_svm = model_svm.fit(Xtrain_BOW , Ytrain)
predSVM = model_svm.predict(Xtest_BOW)

precision_bow, recall_bow, f1_bow = get_stats(Ytest, predSVM)
```

In [ ]:
```
print(f"TF-IDF FOR SVM - {precision_tfidf}, {recall_tfidf}, {f1_tfidf}")
print(f"BOW FOR SVM - {precision_bow}, {recall_bow}, {f1_bow}")
```

```
TF-IDF FOR SVM - 0.8469055374592834, 0.8550074738415545, 0.8509372210651592
BOW FOR SVM - 0.8604989604989605, 0.8249128051818635, 0.8423301958789113
```

Logistic Regression Using Both Features

TF-IDF

In [ ]:
```
model_LR = LogisticRegression(max_iter=10000)

model_LR = model_LR.fit(Xtrain_tf_idf , Ytrain)
predLR = model_LR.predict(Xtest_tf_idf)

precision_tfidf, recall_tfidf, f1_tfidf = get_stats(Ytest, predLR)
```

BOW

In [ ]:
```
model_LR = LogisticRegression(max_iter=10000)

model_LR = model_LR.fit(Xtrain_BOW , Ytrain)
predLR = model_LR.predict(Xtest_BOW)

precision_bow, recall_bow, f1_bow = get_stats(Ytest, predLR)
```

In [ ]:
```
print(f"TF-IDF FOR Logistic Regression - {precision_tfidf}, {recall_tfidf}, {f1_
print(f"BOW FOR Logistic Regression - {precision_bow}, {recall_bow}, {f1_bow}")
```

```
TF-IDF FOR Logistic Regression - 0.8460023631350926, 0.856203288490284, 0.8510722
599177852
BOW FOR Logistic Regression - 0.8583461736004109, 0.8326856003986048, 0.845321193
7278705
```

Naive Bayes Using Both Features

TF-IDF

In [ ]:
```
model_NB = MultinomialNB(alpha=1)

model_NB = model_NB.fit(Xtrain_tf_idf , Ytrain)
predNB = model_NB.predict(Xtest_tf_idf)

precision_tfidf, recall_tfidf, f1_tfidf = get_stats(Ytest, predNB)
```

BOW

In [ ]:
```
model_NB = MultinomialNB(alpha=1)

model_NB = model_NB.fit(Xtrain_BOW , Ytrain)
predNB = model_NB.predict(Xtest_BOW)

precision_bow, recall_bow, f1_bow = get_stats(Ytest, predNB)
```

In [ ]:
```
print(f"TF-IDF FOR Naive Bayes - {precision_tfidf}, {recall_tfidf}, {f1_tfidf}")
print(f"BOW FOR Naive Bayes - {precision_bow}, {recall_bow}, {f1_bow}")
```

```
TF-IDF FOR Naive Bayes - 0.8062212198101683, 0.8549078226208271, 0.82985103501644
41
BOW FOR Naive Bayes - 0.8342844836868741, 0.7720976581963129, 0.8019873719076701
```