This project is done by Student : Ahmed Yasser Mohamed Mohamed Nassar

The purpose of this project is to consider the negative reviews and build a model that predict the reviews either Positive or Negative

- This dataset has 10 columns and alot of unnecessary columns which will not be used so it is removed from the dataset so this includes data cleaning & preprocessing

In [357]:
```
import os, sys
import pandas as pd
import numpy as np

import nltk
import string
import re
from nltk.corpus import stopwords
nltk.download('stopwords')
nltk.download('wordnet')
from nltk.stem import SnowballStemmer, WordNetLemmatizer
from gensim.utils import simple_preprocess

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.model_selection import train_test_split

import pickle
import seaborn as sns
import matplotlib.pyplot as plt
sns.set(font_scale=1.3)
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

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

In [2]:
```
%%time
df = pd.read_csv('Reviews.csv')
```

```
Wall time: 7.15 s
```

In [3]:
```
df.columns
```

Out[3]:
```
Index(['Id', 'ProductId', 'UserId', 'ProfileName', 'HelpfulnessNumerator',
       'HelpfulnessDenominator', 'Score', 'Time', 'Summary', 'Text'],
      dtype='object')
```

```
In [4]: df.shape
```

```
Out[4]: (568454, 10)
```

```
In [5]: f"{df.shape[0]:,} Review"
```

```
Out[5]: '568,454 Review'
```

```
In [6]: cols = ['Text', 'Score']
        df_text = df[cols].copy()
        df_text.head()
```

Out[6]:

	Text	Score
0	I have bought several of the Vitality canned d...	5
1	Product arrived labeled as Jumbo Salted Peanut...	1
2	This is a confection that has been around a fe...	4
3	If you are looking for the secret ingredient i...	2
4	Great taffy at a great price. There was a wid...	5

```
In [7]: df_text.shape
```

```
Out[7]: (568454, 2)
```

```
In [8]: # Drop duplicate
        df_text = df_text.drop_duplicates()
```

```
In [9]: df_text.shape
```

```
Out[9]: (393675, 2)
```

In [10]:
```
df_text1 = pd.DataFrame({'Target':df_text['Score']})
df_text1['Target'] = df_text.loc[df_text['Score'] >= 3,'Target'] = 1
df_text['Target'] = df_text['Target'].fillna(0)
#This was very important to reset the index again as it was set to 568454 w
hich made errors later but it's solved now
df_text = df_text.reset_index(drop=True)
df_text
```

Out[10]:

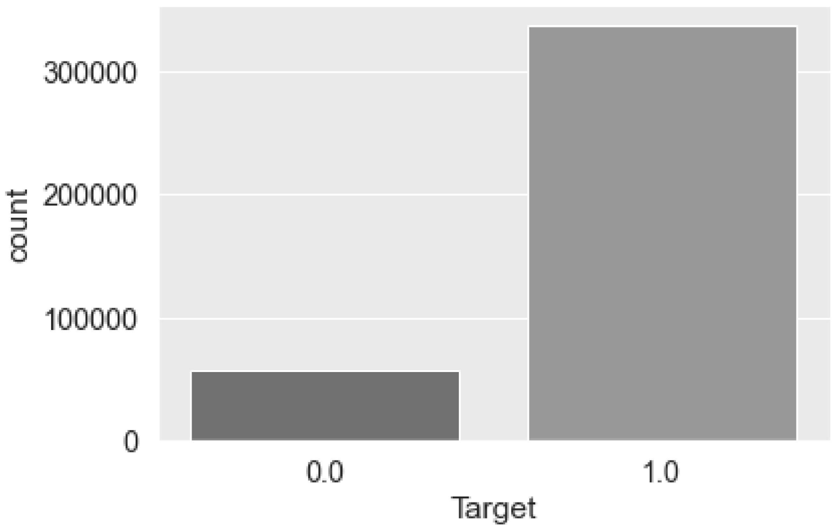
	Text	Score	Target
0	I have bought several of the Vitality canned d...	5	1.0
1	Product arrived labeled as Jumbo Salted Peanut...	1	0.0
2	This is a confection that has been around a fe...	4	1.0
3	If you are looking for the secret ingredient i...	2	0.0
4	Great taffy at a great price. There was a wid...	5	1.0
...	...	...	...
393670	Great for sesame chicken..this is a good if no...	5	1.0
393671	I'm disappointed with the flavor. The chocolat...	2	0.0
393672	These stars are small, so you can give 10-15 o...	5	1.0
393673	These are the BEST treats for training and rew...	5	1.0
393674	I am very satisfied ,product is as advertised,...	5	1.0

393675 rows × 3 columns

In [11]:
```
df_text['Target'].value_counts()
```

Out[11]:
```
1.0    336591
0.0     57084
Name: Target, dtype: int64
```

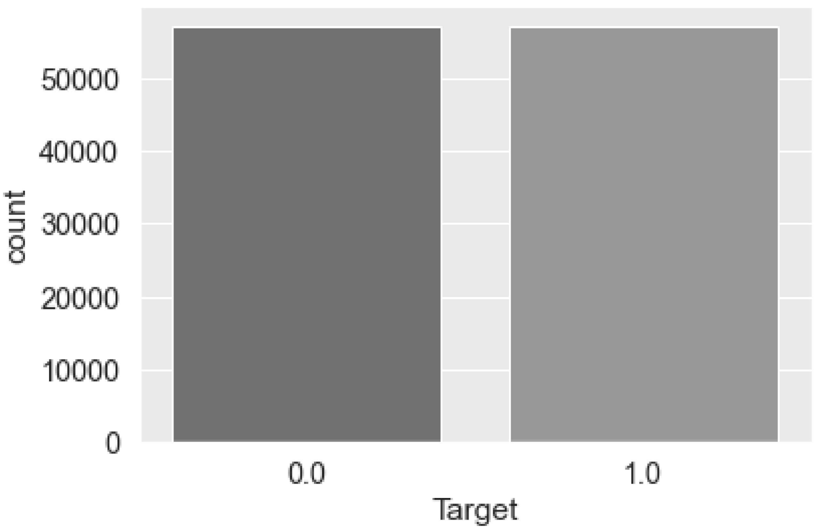
In [12]:
```
ax=sns.countplot(x= "Target", data = df_text)
#ax.set(ylim=(0,8000))
```



In [13]:
```
NEG_N = df_text['Target'].value_counts()[0] #The Number of Negative reviews
df_pos = df_text[df_text['Target'] == 1]['Text'].sample(NEG_N, replace=Fals
e)
df_text_balanced = pd.concat([df_text.iloc[df_pos.index], df_text[df_text
['Target'] == 0]])
df_text_balanced = df_text_balanced.reset_index(drop=True)
```

In [14]:
```
sns.countplot(x= 'Target',data = df_text_balanced)
```

Out[14]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x121d64876d0>
```



In [15]:
```
stop_words = set(stopwords.words('english'))
stop_words.remove('not')
stop_words.remove('no')
stemmer = SnowballStemmer("english")
lemmatizer= WordNetLemmatizer()
```

In [16]:
```
# 1st stopwords
df_text_balanced['NOstopwords_Text'] = df_text_balanced['Text'].apply(lambd
a x: ' '.join([word for word in x.split()

if word not in (stop_words)]))
```

In [17]:
```
# 2nd WordNet Lemmatizer
df_text_balanced['Lemmatized_Text'] = df_text_balanced['NOstopwords_Text'].
apply(lambda x: [lemmatizer.lemmatize(word) for word in x.split()])
```

In [20]:
```
# 3rd Snowball Stemmer
df_text_balanced['SnowballStemmed'] = df_text_balanced['Lemmatized_Text'].a
pply(lambda x: [stemmer.stem(y) for y in x])
```

In [21]:
```python
df_text_balanced.head()
```

Out[21]:

	Text	Score	Target	NOstopwords_Text	Lemmatized_Text	SnowballStemmed
0	This coffee has a good, strong flavor. It does...	4	1.0	This coffee good, strong flavor. It suffer wea...	[This, coffee, good,, strong, flavor., It, suf...	[this, coffe, good,, strong, flavor., it, suff...
1	This is the best Caramel Sauce that I have fou...	5	1.0	This best Caramel Sauce I found. Makes best co...	[This, best, Caramel, Sauce, I, found., Makes,...	[this, best, caramel, sauc, i, found., make, b...
2	For some reason I thought these were actual fr...	3	1.0	For reason I thought actual freeze dried snow ...	[For, reason, I, thought, actual, freeze, drie...	[for, reason, i, thought, actual, freez, dri, ...
3	My 3 cats like the small kibble size and must ...	5	1.0	My 3 cats like small kibble size must enjoy fl...	[My, 3, cat, like, small, kibble, size, must, ...	[my, 3, cat, like, small, kibbl, size, must, e...
4	Albertsons carries a lot of Walden Farms produ...	5	1.0	Albertsons carries lot Walden Farms products ,...	[Albertsons, carry, lot, Walden, Farms, produc...	[albertson, carri, lot, walden, farm, product,...

In [303]:
```python
X = df_text_balanced['Text']
y = df_text_balanced['Target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ra
ndom_state=42)
```

In [304]:
```python
## TFIDF embedding for the Description
vectorizer = TfidfVectorizer()
# fit on training (such vectorizer will be saved for deployment)
vectorizer_tfidf = vectorizer.fit(X_train,y_train)
# transform on training data
X_train = vectorizer_tfidf.transform(X_train)
# transform on testing data
X_test = vectorizer_tfidf.transform(X_test)
```

In [305]:
```python
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[305]:
```
((79917, 56404), (34251, 56404), (79917,), (34251,))
```

In [306]:
```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

In [307]:
```
## initialize your Model
clf = RandomForestClassifier(n_estimators=100)
# Fit your Model on the Training Dataset
clf.fit(X_train,y_train)
# Predict on Test data
preds = clf.predict(X_test)
# Calculate Model Accuracy
acc = accuracy_score(preds, y_test)
print(f"Model Accuracy = {round(acc*100,2)}%")
```

```
Model Accuracy = 84.27%
```

In [388]:
```
def raw_test(review, model, vectorizer):
    lemmatized_word = lemmatizer.lemmatize(review)
    stemmed_word = stemmer.stem(lemmatized_word)
    filtered_words = [word for word in stemmed_word.split() if word not in
stopwords.words('english')]


    embedding = vectorizer.transform(filtered_words)
    #Predict using your model
    predict = model.predict(embedding)
    ZC = np.count_nonzero(predict==0)
    if ZC < 2:
        prediction = 1
    else:
        prediction = 0
    return "Positive" if prediction == 1 else "Negative"
```

In [389]:
```
review_1 = "That's a good Dish, Good Job"
review_2 = "That's the worst Dish ever tasted"
```

In [390]:
```
raw_test(review_1, clf, vectorizer_tfidf)
```

Out[390]:
```
'Positive'
```

In [391]:
```
raw_test(review_2, clf, vectorizer_tfidf)
```

Out[391]:
```
'Negative'
```

In [416]:
```
# This was for testing the model with my self
#review_3 = "but the rest of the experience was really disappointing."
#raw_test(review_3, clf, vectorizer_tfidf)
```

In [417]:
```
# This was for testing the model with my self
#lemmatized_word = lemmatizer.lemmatize(review_3)
#sstemmed_word = stemmer.stem(lemmatized_word)
#filtered_words = [word for word in sstemmed_word.split() if word not in st
opwords.words('english')]
#v=vectorizer_tfidf.transform(filtered_words)
#preds11 = clf.predict(v)
#print(preds11)
#print(np.count_nonzero(preds11==0))
import pickle
```

Saving The Model

```
In [408]: model_name = 'rf_model.pk'
          vectorizer_name = 'tfidf_vectorizer.pk'
          model_path = os.path.join('/', model_name)
          vect_path = os.path.join('/', vectorizer_name)

          pickle.dump(clf,open(model_name,'wb'))
          pickle.dump(vectorizer,open(vectorizer_name,'wb'))
```

Loading The Model

```
In [409]: loaded_model = pickle.load(open(model_name,'rb'))
          loaded_vect = pickle.load(open(vectorizer_name,'rb'))
```

```
In [411]: raw_test(review_1, loaded_model, loaded_vect)
```

```
Out[411]: 'Positive'
```

```
In [412]: raw_test(review_2, loaded_model, loaded_vect)
```

```
Out[412]: 'Negative'
```

```
In [435]: # The Deployment of the model using streamlit
          # Please Note the code for the model is in a seprerate file
          # IMPORTANT to note that the file must include the loaded model that we mad
          e using pickle
          # you will need VScode to make it run as it will not run on jupyter
          # The python file for the model is named ((AFReview.py)) and can be found i
          n the file named (Amazon Food Review)
          # AND THE MODEL WORKS 100% AS IT WORKS HERE IN JUPYTER
          # Thank you
```

```
In [ ]:
```