Sentiment Analysis - Amazon Product Reviews

Param Madan

Introduction:

Background:

Finding the attitude, opinion, and emotion expressed in a text, like product reviews on Amazon, is the aim of sentiment analysis.

One of the biggest online retailers in the world, Amazon has millions of consumers and products. Businesses must comprehend consumer attitudes toward their goods if they want to increase customer satisfaction and make wise choices regarding product development and marketing tactics. The positive and negative features of products, the overall sentiment toward products, and areas for improvement can all be revealed by doing a sentiment analysis of Amazon product reviews.

Machine learning techniques are used to analyze the sentiment of Amazon product reviews. Data collection, data cleaning and preprocessing, feature extraction, model training and evaluation, and result interpretation are commonly involved in the process.

Motivation:

There are several motivations behind conducting sentiment analysis for Amazon product reviews:
Customer feedback:
Amazon receives millions of product reviews from customers worldwide. Analysing the sentiment of these reviews can help companies understand the opinions, preferences, and experiences of their customers.
Product development:
Sentiment analysis can identify common customer complaints or issues, which can be used to improve the quality of products and services.
Marketing:
Understanding customer sentiment can help companies identify positive aspects of their products and services that can be highlighted in marketing campaigns.
Competitive advantage:
Analysing customer sentiment can provide insights into how a company's products and services compare to those of its competitors. This information can be used to identify areas for improvement and competitive differentiation.
Customer service:
Sentiment analysis can identify common customer issues, allowing companies to improve their customer service and support offerings.

Goal:

Overall, sentiment analysis for Amazon product reviews can provide valuable insights into customer sentiment and preferences, which can inform product development, marketing, and customer service strategies, ultimately leading to improved customer satisfaction and loyalty

Methodology:

- First, we imported the necessary libraries that will be needed for our project like nltk, textblob, wordcloud etc.

```
import warnings
warnings.filterwarnings("ignore")
import time
import pandas as pd
import numpy as np
from nltk.corpus import stopwords
from textblob import TextBlob
from textblob import Word
from wordcloud import WordCloud
from wordcloud import STOPWORDS
import string
import nltk
from nltk.tokenize import word_tokenize
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import re
import os
import sys
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
vader=SentimentIntensityAnalyzer()
```

- Then we loaded the dataset on which we are going to perform sentiment analysis.

- After loading the data, we checked for the null values and then we ran a code to remove the null values from our dataset, and then we recheck the dataset to check if any null value is still present or no

#Checking Null values

```
APR_dataframe.isnull().sum()
```

```
]: Id              0
   Product_ID      0
   User_ID         0
   ProfileName    16
   Help_num        0
   Help_denom      0
   Ratings         0
   Time            0
   Summary        27
   Comments        0
   dtype: int64
```

DATA CLEANING STEPS THAT WE HAVE PERFORMED ARE:

- Removing Punctuation: we have removed Punctuation marks, @ and hyperlinks from the data as they do not add any value to sentiment analysis. Removing them can help simplify the data and reduce noise, making it easier to analyze and interpret.

DATA CLEANING PROCESS

#Removing @ and punctuation from the column Summary and Comments

In [6]:
```
def cleancomments (text):
    text = re.sub (r'@[A-Za-z0-9]+', '',text)  #remove @
    text = re.sub ('[^\w\s]', '',text)
    return text
```

In [7]:
```
mainframe['Comments']=mainframe['Comments'].apply(cleancomments)
mainframe['Summary']=mainframe['Summary'].apply(cleancomments)
```

In [8]:
```
stop= stopwords.words('english')
mainframe['Comments']=mainframe['Comments'].apply(lambda x: " ".join(x for x in x.split() if x not in stop))
stop= stopwords.words('english')
mainframe['Summary']=mainframe['Summary'].apply(lambda x: " ".join(x for x in x.split() if x not in stop))
```

#Converting the entire column of summary and comments to lower case

In [9]:
```
mainframe['Comments']=mainframe['Comments'].str.lower()
mainframe['Summary']=mainframe['Summary'].str.lower()
```

#Converting the entire column of summary and comments to String

In [10]:
```
mainframe['Comments']=mainframe['Comments'].astype(str)
mainframe['Summary']=mainframe['Summary'].astype(str)
```

- Removing Stop Words: we have removed commonly used words that do not carry much meaning. Removing them can help reduce the number of irrelevant words in the text, making the analysis more accurate and efficient.
- After that, we have converted entire text to lower case as it can help standardize the data and make it easier to analyze. It can also reduce the number of unique words in the text, which can help reduce the computational complexity of the analysis.
- Later on, we have Converted Data Type to String: In sentiment analysis, the input data is typically in the form of unstructured text. Converting it to string can help standardize the data and make it compatible with NLP tools and techniques.

#Creating tokens

```
def CommentsTokenize(Comments):
    tokenize = nltk.word_tokenize(Comments)
    return [g for g in tokenize if g.isalpha()]
```

```
mainframe['Comments_Tokens']=mainframe['Comments'].apply(CommentsTokenize)
```

Calculated POLARITY and SUBJECTIVITY using TextBlob.

- Polarity in sentiment analysis refers to the sentiment or emotional tone conveyed in a piece of text, which can be positive, negative, or neutral. polarity generally ranges between -1 to +1, where -1 represents a strongly negative sentiment, +1 represents a strongly positive sentiment, and 0 represents a neutral sentiment.

#Calculating polarity and subjectivity for each comment in the Comments column

```
def subjectivity(text):
    return TextBlob(text).sentiment.subjectivity
def polarity(text):
    return TextBlob(text).sentiment.polarity
text_blob['Subjectivity']=text_blob['Comments'].apply(subjectivity)
text_blob['Ploarity']=text_blob['Comments'].apply(polarity)
```

#Based on the polarity score, we are deriving the sentiment of comments into a positive, negative, or neutral value.

```
def polanalysis(score):
    if score<0:
        return 'Negative'
    elif score==0:
        return 'Neutral'
    else:
        return 'Positive'
text_blob['Textblob_Analysis']= text_blob['Ploarity'].apply(polanalysis)
```

- Based on the polarity score we then separate the reviews into negative, neutral, and positive where if the score is less than 0 it will be considered negative if it is 0 then it will be considered neutral and if it is higher than 0 it will be considered as positive.

- The reviewers have given number ratings as well to their reviews so now we use those numbers directly use those numbers and consider the reviews as positive, negative or neutral based on the numbers where if the rating is 2 or less than 2 it will be considered as negative, if the number is 3 it will be considered as neutral and if the rating is 3 or more than 3 it will be considered as positive.

#Based on Ratings provided from the user, We are deriving positive, negative, or neutral value.

```
def ratingsanalysis(Ratings):
    if Ratings<3:
        return 'Negative'
    if Ratings==3:
        return 'Neutral'
    if Ratings>3:
        return 'Positive'
text_blob['Ratings_Analysis']=text_blob['Ratings'].apply(ratingsanalysis)
```

Now we use the Vader sentiment analysis on the same reviews.

- We have also used Vader sentiment analyzer from Natural language tool kit.
- In VADER, typically four columns are generated as output:
- Positive score, Negative score, Neutral score, Compound score: It is A metric that calculates the overall sentiment polarity of the text by normalizing the scores of the first three columns and ranging the polarity between -1 to +1 for each review.
- After that, we have calculated the percentage of positive, negative, and neutral reviews for Textblob, Vader, and the ratings column.

SENTIMENT SCORE

We have created a new dataframe with columns that are required for the machine learning model and have also calculated the sentiment score by assigning 0 and 1 values to negative and positive words, respectively.

#Now we are using Vader Sentiment Analysis.

```
vader_data=mainframe
```

```
temp_data=[]
for row in vader_data['Comments']:
    ab= vader.polarity_scores(row)
    temp_data.append(ab)
vader_new=pd.DataFrame(temp_data)
```

#Based on the compound value, we are deriving sentiment of comments into a positive, negative, or neutral value.

```
def vaderanalysis(compound):
    if compound<0:
        return 'Negative'
    elif compound==0:
        return 'Neutral'
    else:
        return 'Positive'
vader_new['Vader_Analysis']= vader_new['compound'].apply(vaderanalysis)
```

```
vader_new=vader_new.drop(columns=['neg','neu','pos'])
```

```
new_combined_data= pd.concat([text_blob.reset_index(drop=True), vader_new], axis=1)
```

- Then we extracted the percentages of all 3 techniques that is Vader analysis, Textblob analysis and analysis of the ratings of the review and we compared the percentages of all 3 analysis.

#We are calculating the percentage of positive, negative, and neutral values in Ratings analysis.

```
(new_combined_data['Ratings_Analysis'].value_counts()/new_combined_data['Ratings_Analysis'].count())*100
```

```
3]: Positive    78.071325
    Negative    14.427413
    Neutral      7.501262
    Name: Ratings_Analysis, dtype: float64
```

#We are calculating the percentage of positive, negative, and neutral values in textblob analysis.

```
(new_combined_data['Textblob_Analysis'].value_counts()/new_combined_data['Textblob_Analysis'].count())*100
```

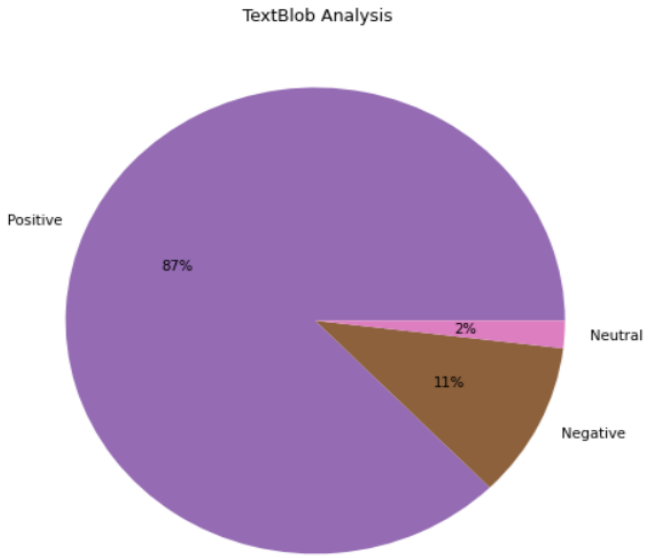
```
22]: Positive    87.316924
     Negative    10.784978
     Neutral      1.898098
     Name: Textblob_Analysis, dtype: float64
```

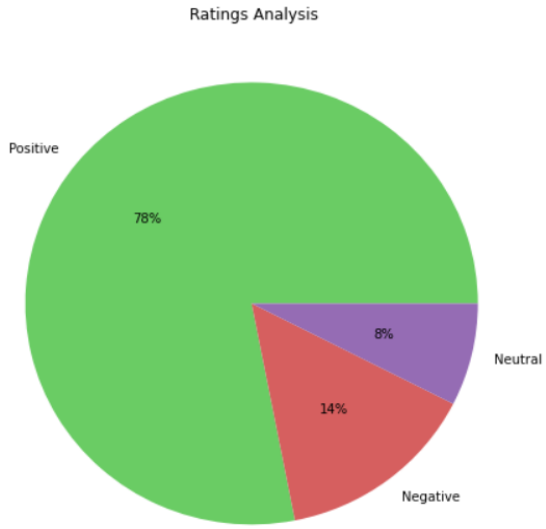
#We are calculating the percentage of positive, negative, and neutral values in Vader analysis.

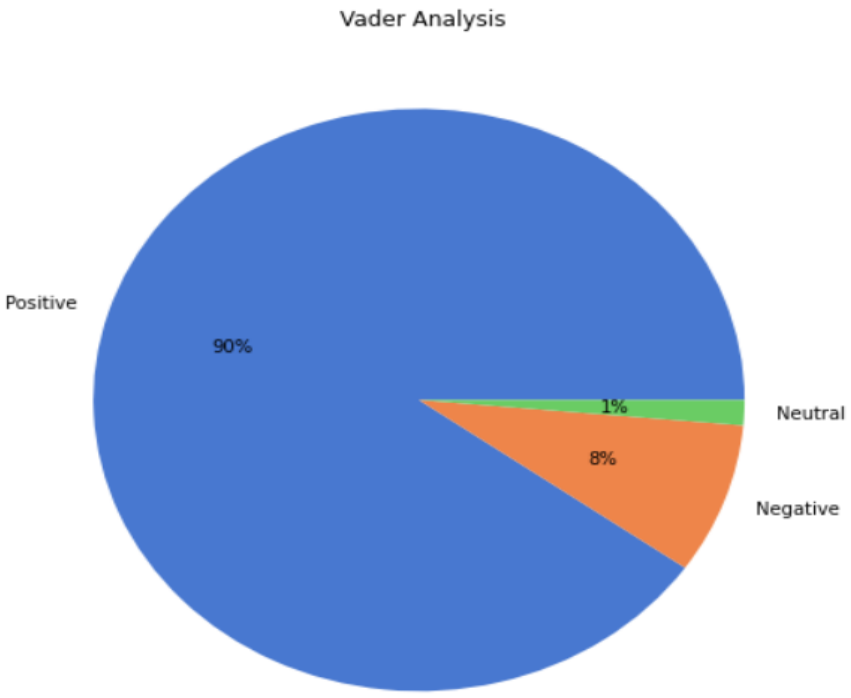
```
(new_combined_data['Vader_Analysis'].value_counts()/new_combined_data['Vader_Analysis'].count())*100
```

```
4]: Positive    90.225910
    Negative     8.385833
    Neutral      1.388256
    Name: Vader_Analysis, dtype: float64
```

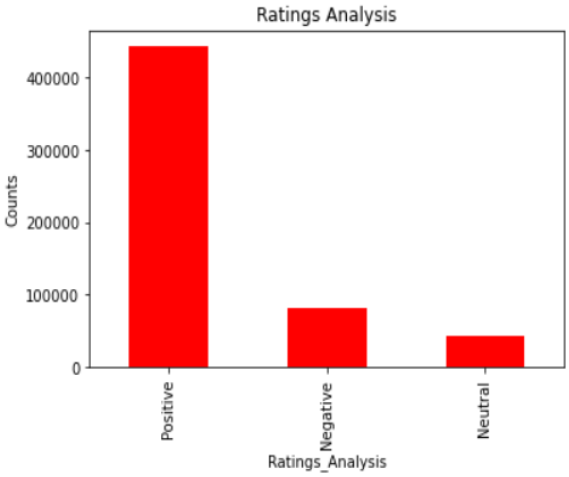
- Then we plotted 3 pie charts of the 3 analysis that we performed, and we plotted the values of the results we received from the analysis.

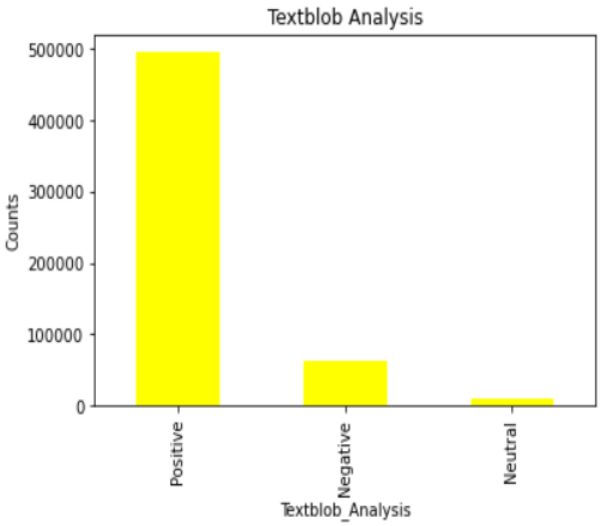


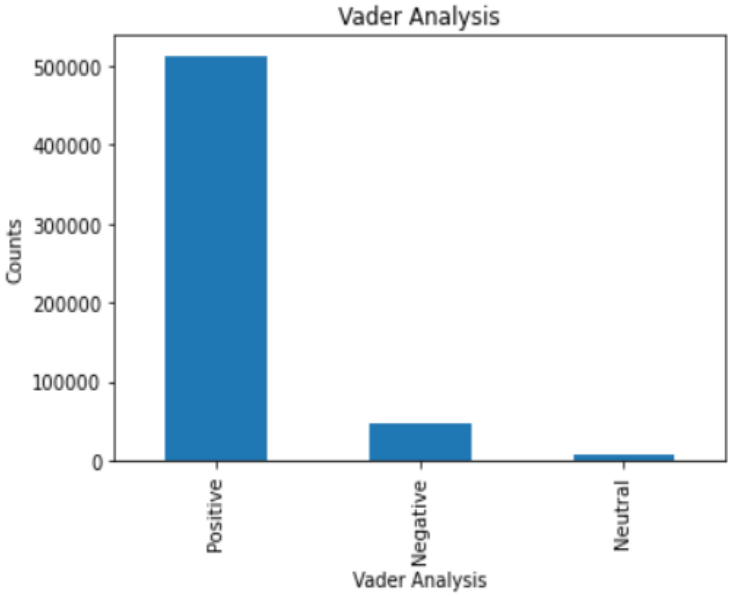




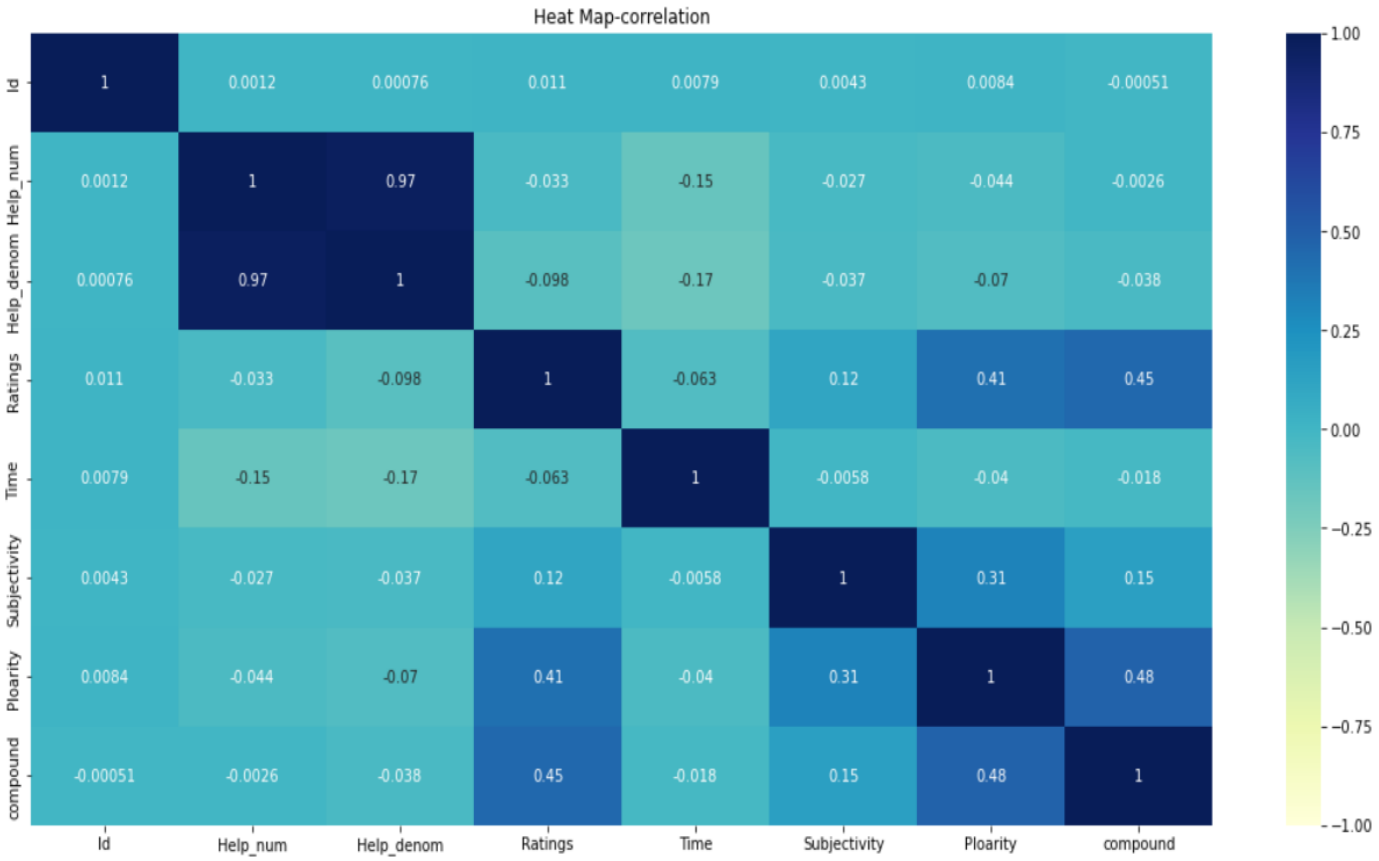
- Then we plotted 3 bar graphs giving the number of positive, negative and neutral reviews on the 3 analysis that we performed.





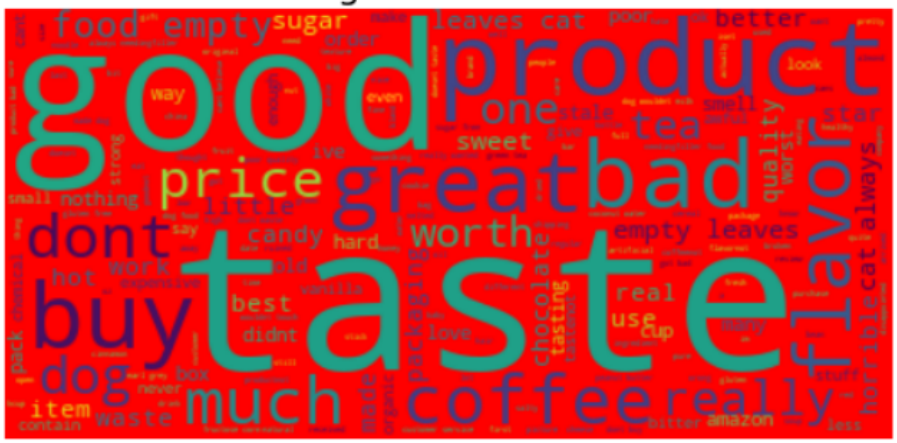


- Then we plotted a heatmap to find out the co-relation.
- Here we can see that the darker the color the more related the attributes are.



- Then we created 3 wordclouds, the first one being a generic wordcloud cloud containing all the keywords, the 2nd wordcloud containing only the positive keywords filtered from the generic wordcloud, and the 3rd wordcloud containing all the negative keywords from the generic wordcloud.

Negative Reviews



Positive Reviews

- Now we start with the feature engineering process, at first we have removed all the unwanted columns that are not required and will not be used in the models
- Then we discarded all the reviews having neutral values and removed them from our columns.

Then we assign 0 and 1 to the negative and positive reviews respectively.

```
model_data= model_data[model_data['Textblob_Analysis'] != 'Neutral']
model_data= model_data[model_data['Vader_Analysis'] != 'Neutral']
```

```
Textblob_model=model_data
```

```
Textblob_model=Textblob_model.drop(columns=['Comments_Tokens','Ratings_Analysis','Vader_Analysis'])
```

```
def Sentiment(Textblob_Analysis):
    if Textblob_Analysis == 'Negative':
        return 0
    else:
        return 1

Textblob_model['Textblob_Analysis_Sentiment']= Textblob_model['Textblob_Analysis'].apply(Sentiment)
```

WHY TFIDF OR BAG OF WORDS?

Before fitting the data into an ML model, we need to preprocess the text data to represent it in a way that the model can understand. Two main techniques used for this task are TFIDF (Term frequency inverse document frequency) and bag of words. These techniques are important for sentiment analysis because they help us to represent the text data in a numerical form that can be processed by ML models.

TF-IDF considers how common or rare a word is across all documents, while Bag-of-Words only looks at the frequency of words in a document. This helps to identify the most important words in a document.

2)Unwanted words that may be present in the data dataset do not add value and can dilute the importance of other words in the bag of words. whereas in TF-IDF, it assigns low weightage to stop words, which improves the focus on important words.

3)TF-IDF is better for ranking search results because it considers both the frequency and rarity of words, helping to identify the most relevant documents.

4)Bag-of-Words can result in large vectors that are hard to handle for large datasets, while TF-IDF can reduce the vector space while maintaining important information, making it easier to analyze large amounts of text data.

5)TF-IDF is more accurate than Bag-of-Words for classification tasks.

#We are using the TF-IDF ON Comments column and transforming it to numerical values.

```
from joblib import parallel_backend
with parallel_backend('threading', n_jobs=-1):
    tfidf_tb = TfidfVectorizer()
    X_train_tfidf_tb = tfidf_tb.fit_transform(X_train)
```

```
X_test_tfidf_tb = tfidf_tb.transform(X_test.astype('U'))
```

Now we have implemented the machine-learning algorithms.

1) **XGBoost:**

- Due to its capacity to handle uneven data, carry out feature selection and engineering, make use of ensemble learning, and scale effectively to big datasets, XGBoost is frequently employed for sentiment analysis of Amazon product evaluations. Its built-in feature selection algorithm can determine which features are crucial for sentiment analysis and it can handle various feature kinds, including text-based data. Sentiment analysis of millions of reviews is a task that XGBoost is well suited for due to its scalability and capacity for handling enormous datasets.

- XGBoost is a powerful and versatile tool that can help to improve the accuracy and efficiency of predictive models, making it a popular choice in many data science and machine learning applications.

XGBOOST TEXTBLOB ANALYSIS

ML ALGORITHM XGBOOST

```
from xgboost import XGBClassifier
```

```
xgboost_model = XGBClassifier()
xgboost_model.fit(X_train_tfidf_tb, y_train)
xgboost_pred = xgboost_model.predict(X_test_tfidf_tb)
print(xgboost_pred)
```

```
[1 1 1 ... 1 1 1]
```

```
xgboost_accuracy=accuracy_score(y_test,xgboost_pred)
```

```
print("XGBoost Accuracy for Textblob Analysis is:",xgboost_accuracy*100)
```

```
XGBoost Accuracy for Textblob Analysis is: 95.14915576479109
```

XGBOOST VADAR ANALYSIS

ML ALGORITHMS FOR VADER ANALYSIS

```
xgboost_model_v = XGBClassifier()
```

```
xgboost_model_v.fit(X_train_tfidf_v, y_train)
xgboost_pred_v = xgboost_model_v.predict(X_test_tfidf_v)
```

```
xgboost_accuracy_v=accuracy_score(y_test,xgboost_pred_v)
```

```
print("XGBoost Accuracy for Vader Analysis is:",xgboost_accuracy_v*100)
```

```
XGBoost Accuracy for Vader Analysis is: 94.55343805169527
```

2) **Decision Tree**:

- Due to their interpretability, ability to handle non-linear relationships, scalability, feature selection capabilities, and ability to be combined with other techniques for improved accuracy, decision tree algorithms are frequently used for sentiment analysis projects.

- We have used the decision tree algorithm because it can help classify and analyze large amounts of text data. Decision tree models work by breaking down a complex problem into a series of smaller, more manageable decisions or branches.

TEXTBLOB ANALYSIS

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
```

```
Decisiontree_model= DecisionTreeClassifier(random_state=34, max_depth=80)
```

```
Decisiontree_model.fit(X_train_tfidf_tb,y_train)
```

```
56]: DecisionTreeClassifier(max_depth=80, random_state=34)
```

```
Decisiontree_pred = Decisiontree_model.predict(X_test_tfidf_tb)
```

```
Decisiontree_accuracy=accuracy_score(y_test,Decisiontree_pred)
```

```
print("Decision Tree Accuracy for Textblob Analysis is:",Decisiontree_accuracy*100)
```

```
Decision Tree Accuracy for Textblob Analysis is: 94.28907699968313
```

VADAR ANALYSIS

Decision Tree

```
Decisiontree_model_v= DecisionTreeClassifier(random_state=34, max_depth=80)
```

```
Decisiontree_model_v.fit(X_train_tfidf_v,y_train)
```

```
6]: DecisionTreeClassifier(max_depth=80, random_state=34)
```

```
Decisiontree_pred_v = Decisiontree_model_v.predict(X_test_tfidf_v)
```

```
decisiontree_accuracy_v=accuracy_score(Decisiontree_pred_v,y_test)
```

```
print("Decision Tree Accuracy for Vader Analysis is:",decisiontree_accuracy_v*100)
```

```
Decision Tree Accuracy for Vader Analysis is: 94.26101127155854
```

3) **Random Forest:**

- The random forest algorithm is trained on a large dataset of labelled text data, such as customer reviews, to accurately predict the sentiment of new, unlabeled text. Additionally, the algorithm is relatively easy to implement and can handle large volumes of data, making it a popular choice for sentiment analysis applications.

RANDOM FOREST TEXTBLOB ANALYSIS

Random Forest

```
from sklearn.ensemble import RandomForestClassifier
```

```
Randomforest_model= RandomForestClassifier(n_estimators=90, max_depth=150)
```

```
Randomforest_model.fit(X_train_tfidf_tb,y_train)
```

```
1]: RandomForestClassifier(max_depth=150, n_estimators=90)
```

```
randomtree_pred=Randomforest_model.predict(X_test_tfidf_tb)
```

```
Randomtree_accuracy=accuracy_score(y_test,randomtree_pred)
```

```
print("Random Tree Accuracy for Textblob Analysis is:",Randomtree_accuracy*100)
```

```
Random Tree Accuracy for Textblob Analysis is: 92.13254266443349
```

RANDOM FOREST VADER ANALYSIS

Random Forest

```
Randomforest_model_v= RandomForestClassifier(n_estimators=90, max_depth=150)
```

```
Randomforest_model_v.fit(X_train_tfidf_v,y_train)
```

```
1]: RandomForestClassifier(max_depth=150, n_estimators=90)
```

```
Randomforest_pred_v = Randomforest_model_v.predict(X_test_tfidf_v)
```

```
Randomforest_accuracy_v=accuracy_score(Randomforest_pred_v,y_test)
```

```
print("Random Forest Accuracy for Vader Analysis is:",Randomforest_accuracy_v*100)
```

```
Random Forest Accuracy for Vader Analysis is: 93.52224887963423
```

Results and Analysis:

At first, the XGBoost gave us an accuracy of 90% later we removed the null values and performed data cleaning and later we got an accuracy of 95.14%
In the decision tree model, we checked the various parameters of random state and max depth and the accuracy we obtained was 91% later after cleaning the data and removing the null values we received an accuracy of 94.28%

Final Results:

XGBoost - Textblob Analysis - 95.14%
XGBoost - Vader Analysis - 94.53%

Decision Tree - Textblob Analysis - 94.28%
Decision Tree - Vader Analysis - 94.26%

Random Forest - Textblob Analysis - 92.13%
Random Forest - Vader Analysis - 93.52%

	ML Models	TB_Accuracy	Vader_Accuracy
0	XGBoost	95.149156	94.553438
1	DecisioN Tree	94.289077	94.261011
2	Random Forest	92.132543	93.522249

Dataset Description:

Model Selection and Evaluation: The dataset includes Amazon product reviews from customers. It consists of more than 500K reviews. It consists of both numerical and text data. There are 10 columns and 568454 rows.
The following are the columns present in the file:
Id, productid, UserId, ProfileName, HelpfulnessNumerator, HelpfulnessDenominator, Score, Time, Summary, Text.
Data Source:
https://www.kaggle.com/datasets/arhamrumi/amazon-product-reviews

Conclusion:
In conclusion, this data science project concentrated on applying machine learning to perform sentiment analysis for Amazon product reviews. The goal was to categorize customer feedback as either positive, negative, or neutral depending on how they felt about the product. Data preprocessing, feature extraction, model selection, and evaluation were all part of the project.
Overall, this study showed the effectiveness of sentiment analysis in comprehending client feedback and pinpointing areas in need of development. Businesses can use machine learning techniques to improve their products and services by using consumer reviews as a source of insightful data.

References:
https://machinelearningmastery.com/develop-first-xgboost-model-python-scikit-learn/
https://youtu.be/AnvrJNLKp0k
https://youtu.be/TRnPslOCbv0
https://youtu.be/Alu_cCXNS-k

Thank You.