Flipkart Reviews Sentiment Analysis using Python

Ghufran Ahmed

School of Computer Science&

Artificial Intelligence

SR University

Warangal, Telangana State, India

ghufranahmedw02@gmail.com

Abstract - Customer feedback on the ecommerce platform Flipkart is crucial for businesses, impacting decisions ranging from product enhancements to marketing strategies. This research explores sentiment analysis of the flipkart review using advanced Natural language processing (NLP) techniques to extract actionable insights. The study begins with extensive data collection, gathering a diverse range of Flipkart reviews across various product categories. Through thorough preprocessing steps, including noise removal and tokenization, text data is to prepare for the sentiment analysis. Advanced machine learning models like deep learning classifiers or sentiment lexicons are then employed to determine sentiment polarity of each reviews.

Analysis uncovers the spectrum of sentiment, ranging from highly positive endorsements to critical feedback. By analyzing sentiment trends across different product categories and time frames, the study reveals valuable patterns and customer preferences.

In summary, this study contributes to the evolved field of the sentiment analysis in ecommerce to offering actionable intelligence derived from Flipkart reviews. These insights empower business to enhance the customer experience, optimize product offerings, and the drive sustainable growth in the competitive digital marketplace.

Keywords: Flipkart Reviews, Sentiment analysis, Ecommerce, Customer feedback, Natural Language Processing, Machine Learning, Business Intelligence

I. INTRODUCTION

In the era of digital commerce, online platforms like Flipkart serve as hubs for consumer feedback, offering invaluable insights into product satisfaction, user experiences, and market trends. Sentiment Analysis of Flipkart review emerges as a vital tool to businesses seeking to understand customer sentiments, gauge brand perception, and make data driven decisions. Flipkart, being one of India's leading e-commerce platforms, host a diverse array of product reviews across categories such as electronics, fashion, home appliances, and more. These reviews encapsulate customer opinions, ranging from glowing recommendations to constructive criticisms, shaping purchasing behaviors and influencing brand loyalty. Sentiment analysis, the subfield of the Natural language processing (nlp), enables businesses to extracting sentiment polarity from the text data, categorize reviews as the positive, negative, or the neutral base on the express sentiments. By employing machine learning algorithms and nlp technique, Sentiment analysis algorithm can discern the underly emotions and opinions conveyed in Flipkart reviews. primary objective of the research project is conducting a sentiment analysis of Flipkart reviews, leveraging advanced NLP methodologies and machine learning models. The analysis aims to uncover patterns, trends, and sentiments prevalent in Flipkart reviews across different product categories.

Key components of the sentiment analysis process include data collection from Flipkart's review platform, preprocessing of textual data to remove noise and standardize formats, sentiment polarity classification using sentiment analysis algorithms, and interpretation of sentiment insights for actionable decision-making. By analyzing Flipkart reviews sentiments, business can gain value insight into the customer preference, identifying the area for product improving tailor marketing strategies, and enhance the overall customer satisfactions. This research can contribute to the growing field sentiment analysis in e commerce, providing practical application for businesses operating in digital marketplaces.

II. LITERATURE SURVEY

Sentiment Analysis of Flipkart reviews involve analyzing text data determining the sentiment express by customer towards products or services on the Flipkart platform. This literature survey delves into various studies and approaches related to sentiment analysis, focusing on Flipkart reviews.

As online reviews increasingly shape consumer purchasing decisions, understanding and interpreting customer sentiments have become essential for online retailers seeking to enhance customer satisfaction and product evaluation processes. Leverage Natural language processing software, study focus on analyzing text of online reviews to classify sentiments and generate overall scores that reflect customer perceptions of specific products or services.

By dividing the dataset into the training & testing set in an 80:20 ratios, study reveals that positive attitudes constitute 50.56% of the total data, while negative sentiments make up 49.44%. Machine learning algorithm are trained on the training dataset, with Random Forest classifier being a key tool used for the sentiment classification. Evaluation metric such as precision value, recall value, F1-score, and balanced accuracies are employed to assessing the performance Sentiment analysis model.

Research showcases impressive result, with classifier achieve high levels of recall values, precision, and the F1-score, all at 98%. Python programming language is utilized alongside libraries like pandas, NumPy, Seaborn, and Matplotlib for data analysis and visualization, as well as Natural Language Tool Kit packages for text processing. Goal of the study is to develop a robust Sentiment analysis model that accurately categorize customer reviews based on sentiment and generate an overall score indicative of customer Sentiment towards specific products or services. This research underscores the significance of Sentiment analysis in e-commerce for extract the valuable insight from the customer feedback and improving decision making processes within the industry.

A popular online shopping platform. The study aims to leverage user comments as valuable insights to evaluate product quality in the realm of e-commerce, where online shopping is gaining significant traction due to factors like competitive pricing and efficient delivery services.

The research explores the fundamentals of opinion mining, encompassing various key approaches such as Extraction, Clustering, and Classification. By extracting laptop product reviews from Flipkart and storing the data in a CSV file, the researchers conduct clustering on the pre-processed reviews to categorize them effectively. Through the application of classification techniques, the reviews are segmented into positive and the negative sentiments, enable a comprehensive analysis of customer feedback.

The literature survey section provides comprehensive overview of existing the researches in field of opinion mining and sentiment analysis. previous study has focused on tools and the techniques utilized in opinion mining, emphasizing processes like Opinion Retrieval, Classification, and Summarization. Additionally, research has been conducted on Sentiment analysis of the review from prominent e-commerce platforms like Amazon, employ algorithms such as the Naive bayes, Logistic regression, and Senti Wordnet for analysis.

The study also highlights the development of Decision Making and Analysis tools based on sentiment analysis dictionaries, catering to both customers and manufacturers for product evaluation.

By analyzing reviews from various e-shopping websites, the research aims to classifying review into the Positive, Negative, and Neutral category, aiding consumers in making inform decision based on the Sentiment analysis of products.

Presented in the publication showcases the significance of leveraging user feedback for evaluating product quality in dynamic landscapes of ecommerce. Research contributes valuable insights into opinion mining and Sentiment analysis methodologies, paving way for enhanced decision-making processes for consumers and manufacturers alike.

The focuses on utilizing techniques to extracting valuable insights from customer reviews on Flipkart, one of India's prominent online marketplaces. This endeavor is crucial in the context of the highly competitive ecommerce landscape, where understanding customer sentiment play pivotal role in the business success.

The abstract highlights the project's core objectives, which include collecting and preprocessing a substantial dataset of Flipkart reviews. Preprocessing tasks involve texted cleaning, tokenization's, and the stemming ensure data quality and

consistency. Once the data is prepared, advanced Analysis algorithms are applying to classification reviews into Positive, Negative, sentiment. Introduction delves deeper into the project's purpose, emphasizing the need for a knowledge system that provides comprehensive insights into product reviews on ecommerce platforms.

By analyzing customer comments and presenting them in a polarity-based manner, the system aims to help users understand sentiments more effectively. This is particularly relevant as ecommerce sites like Flipkart serve as major sources for customers to gauge product quality and make informed purchasing decisions. The sentiment analysis process involves several key components. First, the comments provided by customers are collected and separated into bag-of-words. These bag-of-words are then compared with a trained classifier to identify and categorize sentiments as either positive or negative. This classification process is a crucial for underestimate of Customer Satisfaction level the overall sentiment surrounding products on Flipkart. The project's significance lies in its ability to save time for users by compiling ratings and reviews into a coherent format. This not only provides an accurate overview of product opinions but also streamlines the decision-making process for potential buyers. Moreover, the project's use of advanced NLP and ML techniques enhances the accuracies and efficiency of analysis, ensuring reliable result.

Introduction together provides a comprehensive overview of the project's goals, methodology, and significance. By leveraging NLP and ML, the project aims to contribute valuable insights into customer sentiment on Flipkart, ultimately benefiting both consumers and businesses operating in the ecommerce domain.

The project focuses on Sentiment analysis in context of e commerce websites, where vast amounts of data are generated and play a crucial role in assessing product quality. E-commerce platforms have become increasingly popular as primary sources for customers to obtain accurate ratings and reviews of products.

The system's advantage lies in saving time and simplifying the process for users to aggregate ratings and reviews, providing a comprehensive view of a product. The methodology involves organizing product reviews into positive and negative categories, with the use of a trained classifier to categorize sentiments. Sentiment analysis aims to understand consumer sentiments, thoughts, and emotions related to specific products, categorizing these sentiments into different polarities. By

gathering various reviews and ratings from diverse online products, the project processes this data to determine sentiment polarity.

The system employs Natural language Processing, textual analysis, and computation linguistic to recognize, extraction, quantifying and examine sentiments and subjective information. Through the extraction of contextual information from content, businesses can understand the social Sentiment surrounding of their brand, product, and service. The project enhances the product review process by focusing on customer comments and reviews, enabling users to gauge the sentiment expressed in these comments.

Preprocessing and the feature extract steps are the crucial in filtering and determining insights from reviews and ratings. The bag of words method is employed for effective feature extraction, creating a vocabulary for training machine learning algorithms. The system presents sentiments in an easily comprehensible manner, allowing customers to discern the polarity of reviews on e-commerce websites. Analysis simplifies the process of analyzing vast amount of data, providing users with relevant and suitable product reviews. Sentiment analysis can apply to evaluate customer feedback on Flipkart products. Positive sentiment indicates Customer satisfactions, while Negative sentiment highlights areas for improvements. This analysis helps companies enhance product features, quality, and customer experience.

Monitoring sentiment in Flipkart reviews enables businesses to manage their brand reputation effectively. positive sentiment can be leveraged for the marketing purposes, while negative alerts company to issues that require attentions, as product defects Poor customer Services. Sentiment analysis helps in comparing customer sentiments across products offered by different brands on Flipkart. analyzing trends, business can gain the insights in to how their products are perceived relative to competitors, informing competitive strategies and market positioning.

Flipkart reviews serve as a valuable source of market insights. Sentiment analysis uncovers customer preferences, emerging trends, and market sentiment towards specific products or brands. This information guides companies in product development, marketing campaigns, and overall business strategies. NLP is fundamental for understanding and processing textual data from Flipkart reviews.

Technique like tokenization part of speech tagging's, and sentiment algorithms are applied to extract meaningful insights.

These algorithms classifying the sentiments of review into categories. Techniques like Bag of Words (bow), Term frequency inverse document frequency (TFIDF), and the learning models such support vector machines (SVM) the deep learning model like Recurrent neural networks (Rnn) can employed.

Machine Learning model are train on the labeled data recognize patterns and sentiments in reviews. deep learning model, especially neural Networks like convolutional neural Networks (cnn) or the Long short-term memory (Lstm) networks, excel in capturing complex relationships within text data. Words embedding such as Word 2Vec, glove, or Fast text are used to represent the words in a continue Vector space. These embeddings capture semantic relationship between word, enhancing the accuracy of sentiment analysis. Data preprocessing involves tasks like text cleaning (Removing stop word, punctuations, and special character), stemming and lemmatize to reduce words to their root form, and handling of unstructured text data. Python library such as nltk (Natural language toolkit), spaCy, Text Blob, genism provides extensive functional abilities for textual process, Sentiment analysis, and machine learning tasks. Tools like Matplotlib, Seaborn, or Plotly are used to visualize sentiment analysis results.

This involves cleaning the textual data by remove noises, such as the html tag, punctuations, character, and stop words. It also includes tasks like lowercasing, stemming, and lemmatization to standardize the texted. Tokenization break down the textual into individual word or token. This step is essential for further Analysis as it creates a structure formatting for the textual data.

Feature extraction involves converts the text data into numeric feature that learning model can understand. Techniques like Bag of words (Bow), term frequency Inverse document frequency (TF IDF), and words embedding (such as Word 2Vec or glove) are commonly uses for this purpose.

Sentiment lexico are dictionary containing word and their corresponding the sentiment score (positive, negative, neutral). These lexicons help assigning sentiment label to text based on the sentiment polarity of words present in the text. machine learning model, such as support vector machines (svm), Naive bayes,

logistic regression, and deep learning models like Recurrent Neural Networks (RNNs) or convolutional neural networks (CNNs), can train on label data to classify the textual into sentiment category. This technique focuses on identify and analyze specific aspects or features mention in text.

(e.g., product features in Flipkart reviews) and associating sentiment scores with each aspect. It provides a more detailed analysis of sentiment by considering different aspects separately. Gather the data you want to analyze. This would be texts data from the social medias, Customer review, surveys, or any other source where sentiment is expressed. Cleans the data by removing noises such as special character, punctuations, and stop words. You may also need to handle issues like misspellings, slang, or abbreviations. Break the text down into smaller units, such as the words or phrases, which are easier to analyze. This step helps to structure the text data for further processing. Standardizing the textual data converts all tokens to Lower case, handling contractions, and dealing with other variations that might affect analysis consistency. This could include word frequency, n-grams (sequences of words), or more complex features like word embeddings.

Apply a sentiment classification algorithm to classify each piece of text into categories such sentiment. machine learning model like support vector Machines, naive bayes, or deep learning model such as recurrent neural networks or transformer are commonly used for this task.
Analyze the results of sentiment classification to gain insights. This could involve calculating overall sentiment scores, identifying trends or patterns in sentiment over time or across different sources, and extracting key themes or topics associated with different sentiment categories.
Visualize the sentiment analysis results using the chart, graph, or other visuals aids to make easier to understands and interprets the sentiment patterns. Interpret the findings to draw meaningful conclusions and make an informed decision based on the sentiment's insights. This tool is specifical designs for the social media textual analysis. It assigns the sentiments score to individual word and compute overall sentiments of a sentence documents. Text Blob is python Library that offers a simple Api for common Natural language Processing task, including sentiments analysis. It uses a pre trained sentiment analysis model to classify the text into categories. Watson NLU is a powerful tool that provides sentiment analysis

capabilities along with other NLP features like entity recognition, keyword extraction, and emotion analysis. Google's NLP API includes sentiment analysis functionality that can analyze text in multiple languages and provide sentiment scores ranging from 1 negative to the 1 positive Developed by Stanford University, Core nlp offers sentiment analysis features as part of its suite of NLP tools. It provides sentiment scores and sentiment labels for text input.
Microsoft's Text Analytics API includes as one of its features. It can analyze sentiment at both document and sentence levels, providing sentiment scores and confidence scores. Analyzing sentiment in Flipkart reviews helps businesses understand customer opinions, preferences, and satisfaction levels regarding products and services. This insight guides product developments, market strategy, and customer services improvement.
Analyzing sentiment in Flipkart reviews helps businesses understand customer opinions, preferences, and satisfaction levels regarding products and services.

Analyzing the information provided in the academic paper titled sentiment analysis on Reviews of E commerce sites using machine learning algorithm reveals a comprehensive study on sentiment analysis across multiple languages. The paper delves into the intricacies of sentiment analysis in the contextual of e commerce review, a domain where understanding customer sentiments is crucial for business success. The authors begin by acknowledging the challenge inherent in sentiments analysis, which deals with unstructured text data's, capturing nuances in language, and handling sentiment polarity variations across different languages. These challenges set the stage for the methodologies employ in study, which meticulously detail in the paper. Data collection is a foundational step in any machine learning project, and the authors emphasize the importance of collecting a diverse and representative dataset for training and testing their sentiment analysis models. They specifically mention collecting data in Bangla, English, and Romanized Bangla languages, highlighting the multilingual aspect of their research.
Preprocessing is a critical phase in text analytics, and the authors outline their preprocessing pipelines, which likely includes steps such as tokenization, lower casing, punctuations removal, stop word removal, and stemming or lemmatizations. These preprocessing techniques help standardize the text data and make it suitable for feature extraction and model training.

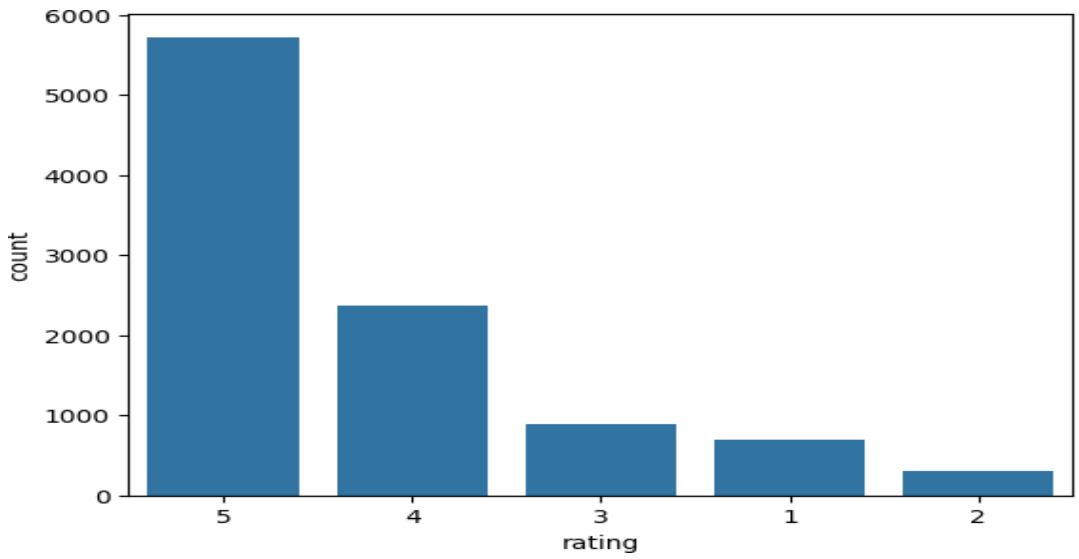
Oversampling is mentioned in the methodology, indicating that the authors likely dealt with class imbalance in their datasets. Class imbalance is a common issue in sentiment, where one sentiment class may dominate the dataset, leading to biased model predictions. Oversampling Techniques such as smote (synthetics minority over sampling technique) are used to address this imbalance and improve the model abilities to generalize to unseen data.

Feature extraction plays crucial role in transformation text data into the numerical feature that machine learning model can process. The author describes their feature extractions method, which may include technique like Bag of Words (Bow), TFIDF (Term frequency Inverse document frequency), word embeddings, or a combination of these methods. These features capture semantic information from the text, enabling the models to learn meaningful patterns related to sentiment. The paper then discusses the machine learning algorithms employed in sentiment analysis, namely multinomial naive Bayes, support vector Machine (svm), logistic regressions, random Forest, k-Nearest neighbors (Knn), and Decision tree. Each algorithm has its strengths and weaknesses, and the authors likely conducted experiments to evaluate the performance of these algorithms on their datasets. evaluation metrics such as accuracy's, precisions, recalls, F1 score, and ROC area are used to assess the models' performance across different languages. These metrics provide insights into how well the models can classifying sentiments correctly, including their abilities to handle false positives and false negatives.

The result sections the paper presents a detail analyze of the performance of each machine learning algorithm on Bangla, English, and Romanized Bangla datasets. This analysis includes comparisons of accuracy's, precisions, recalls, F1 scores, and roc area, providing a comprehensive view of how each algorithm performs in sentiment analysis across languages.

Overall, the paper contributes the field of sentiment analyze by address the challenge of multilingual analysis in the e commerce domain. It provides valuable insights into the effectiveness of various machine learning algorithms and lays the foundation for further research in improve sentiment analysis accuracy and scalability across languages and domains. Identifying recurring negative sentiments in reviews allows businesses to pinpoint areas for product enhancement

or bug fixing. Positive sentiments can highlight features that customers love, aiding in product prioritization and refinement.
Comparing sentiment trends across similar products or brands on Flipkart can reveal competitive strengths and weaknesses. Businesses can use this information to differentiate their offerings and stays ahead in market. sentiment analyze can assess the effectiveness of marketing campaigns by measuring customers sentiments before and after promotional activities. Positive shifts in sentiment can indicate successful campaigns, while negative shifts may signal areas needing improvement. Analyzing sentiment can highlight Recurring issue pain point mentioned in interviews. This information can guide improvement in customers service processes, leading to better customers experiences and higher satisfaction levels. Sentiment analysis can detect emerging trends or topics of interest among customers. Businesses can capitalize on positive trends and address negative ones proactively.



Data set:

https://drive.google.com/file/d/1pNSAgVsWkBjPFOOIYx8RNDEHvk2YU0kp/view

	review	rating
0	It was nice produt. I like it's design a lot. ...	5
1	awesome sound....very pretty to see this nd th...	5
2	awesome sound quality. pros 7-8 hrs of battery...	4
3	I think it is such a good product not only as ...	5
4	awesome bass sound quality very good bettary l...	5

III. PROBLEM DEFINITION

The object for this project is to conduct the analysis of Flipkart reviews, leveraging Learning and the Natural language processing of technique for classifying customers sentiment expressed the reviews as positives, negatives, or the neutral. The goal is extraction value insight from review to understand the customers satisfactions and identify areas for improvements and inform businesses decision-making processes.

PROPOSED MODEL:

Studying customer sentiments for popular e commerce platform, Flipkart. This dataset typically includes a large collections user review along corresponding sentiment labels assigned to each review. Researcher and data analysts often uses this dataset to trains and test sentiment analysis models, aiming to extract insights into customer on preference, product of satisfactions level, and trends in consumer behavior. The dataset's diversity in product categories, languages, and review lengths makes it a rich source for exploring various techniques in the natural language of processing and sentiment analysis, contributing significant to advancement the sentiment analysis methodologies and applications in real-world scenarios.

1. Data Gathering and Integration: Collect review from the Flipkart using of web scrapings or APIs. Integrate data in the structured formats for the analyze.

2. Data Preprocessing: clean textual data by removed noises (special characters, stop words), normalizing text (lowercasing, stemming/lemmatization), and handling missing values if any.

3. Feature Extraction: Extract features from the pre-processed text, such as bag-of-words, TF-IDF (Term Frequency-Inverse Document Frequency), n-grams, word embeddings (Word2Vec, Glove), or transformer-based embeddings (BERT, RoBERTa) to represent the text numerically.

4. Sentiment Analysis: Apply sentiment lexicons (like VADER or AFINN) or models to classification of the sentiment for each review (positive's, negative's, neutral's).

5. Aspect-Based Sentiment Analysis (Optional): If needed, performs the aspect-based identification sentiment related to specific of the aspect or feature mention in for the reviews (e.g., product quality, delivery, customer service).

6. Model Evaluation: Evaluate models using the metrics like accuracy's, precisions, recalls, and F1-scores. Use technique to like cross the validation to ensure robustness.

7. Visualization: Visualize the sentiment analysis results using dashboards or plots to provide actionable insights to stakeholders.

This model covers the key steps from data collection to visualization, incorporating techniques like text preprocessing, feature extraction and optionally aspects based sentimental analysis for a comprehensive analysis of Flipkart reviews.

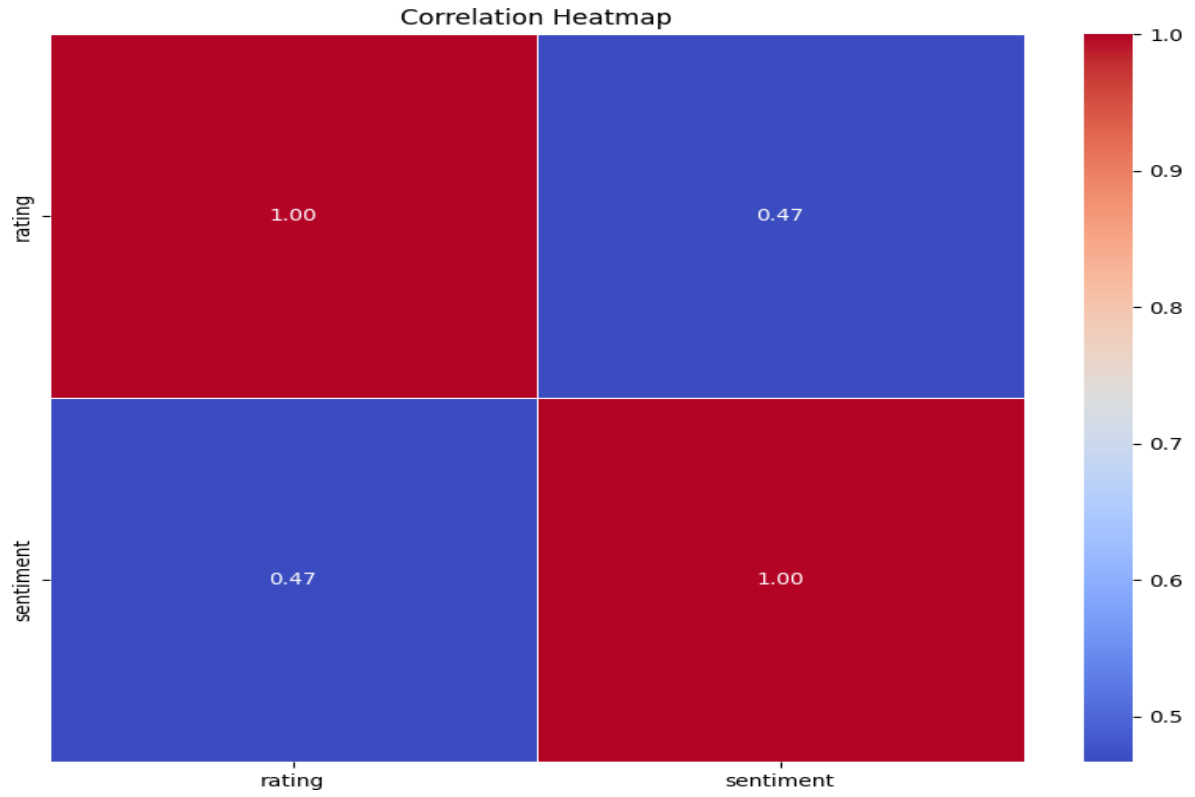
ALGORITHMS:

Various algorithms techniques can used for depends on the complexities of the analysis and desires accuracy. Here are some commonly used algorithms:

- Logistic regression
- Support vector machines (SVM)
- Decision trees
- Naive bayes
- Multi-Layer Perceptron's

IV. RESULTS AND COMPARITIVE STUDY:

These accuracy ranges provide an estimation of the performance you might expect from each algorithm when applied to sentiment analysis specifically on Flipkart reviews. Actual accuracy's can be based on specifics to your dataset, preprocessing techniques, feature engineering, and model tuning.

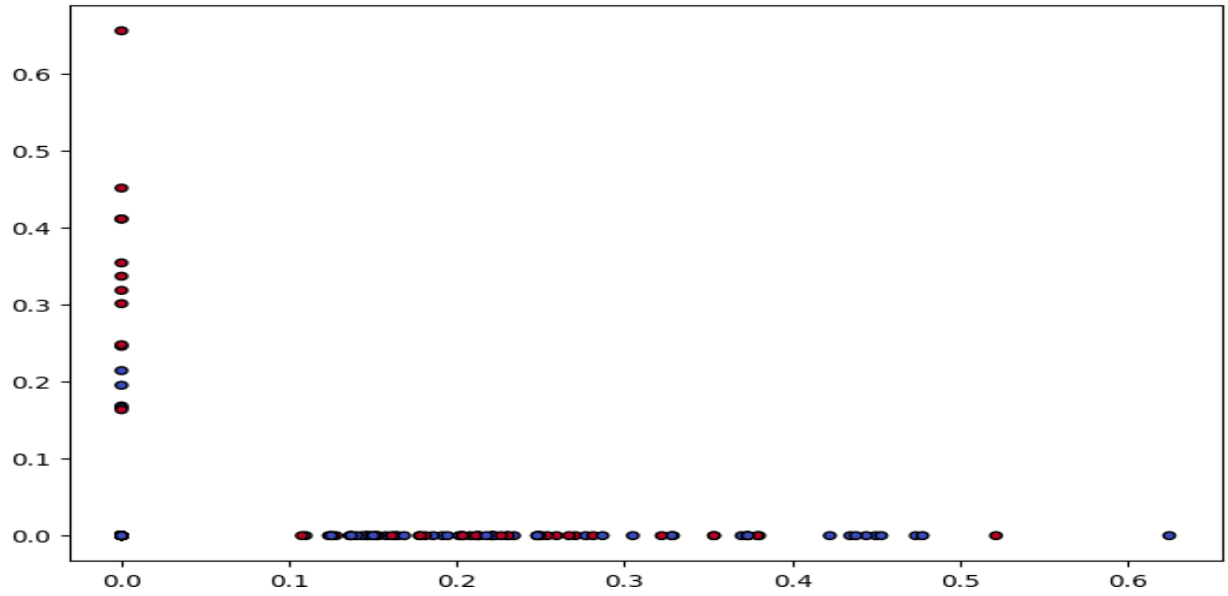


Fig(a). Heatmap

Heatmap representing the analysis of Flipkart review. It's divided into two main sections, indicating negative and positive sentiments. The heatmap is further divided into rows labeled with ratings from 1 to 5. Each cell within the grid contains a number representing the count of reviews with that rating and sentiment. The color intensity, guided by the scale on the right, indicates the concentration of reviews; however, all cells have low counts as per this scale

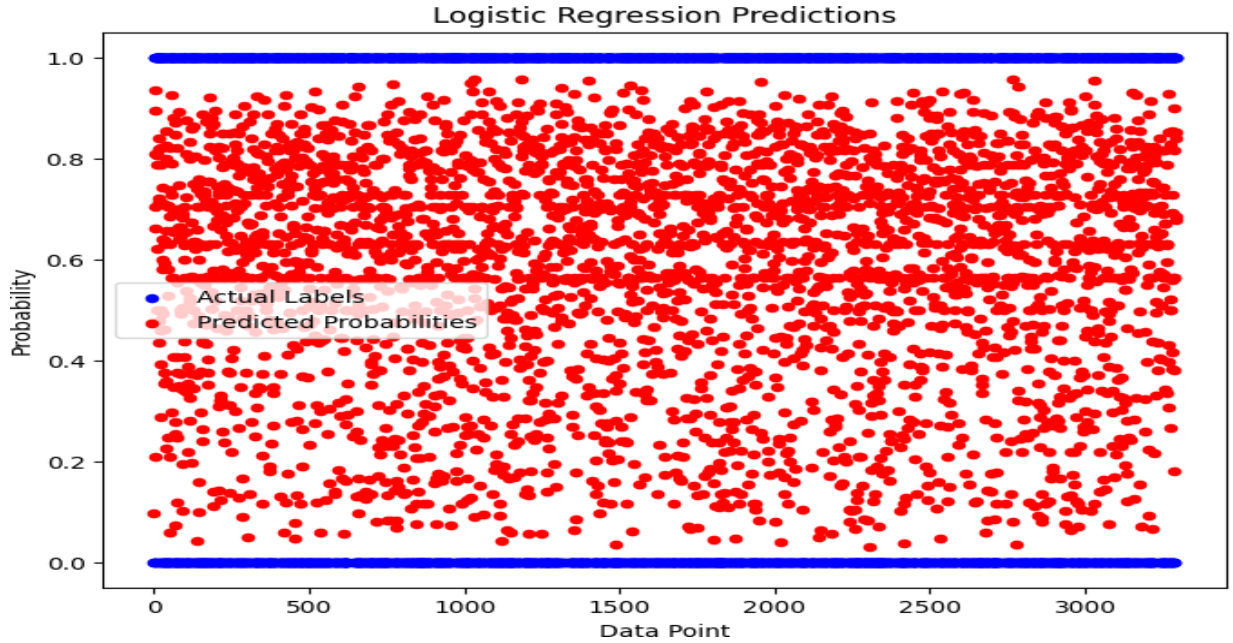
1. logistic regression: Logistic Regression are another commonly used algorithm in sentiment analysis. models of Probabilities to sentiment classes give to the input feature and is interpretable.

Logistic Regression Accuracy: 0.7309444275736411



Fig(b)

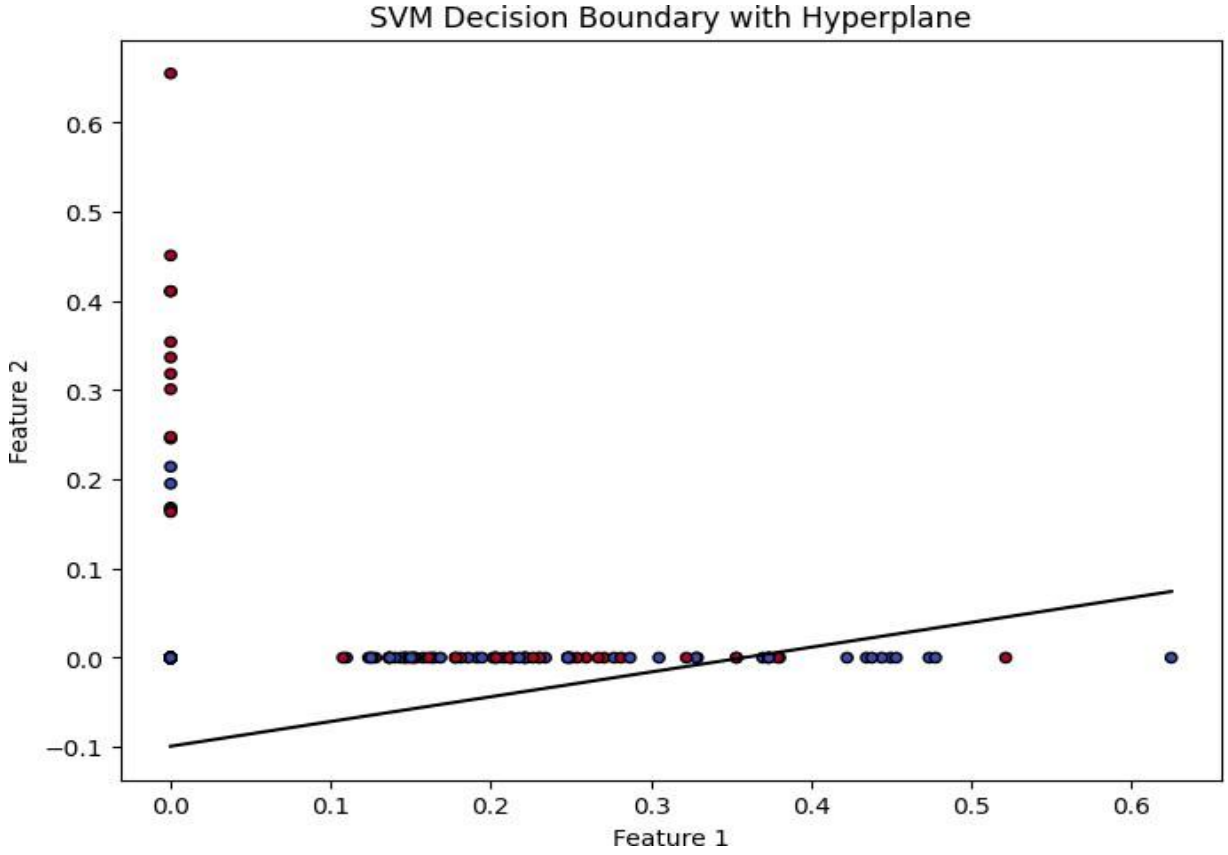
It displays data points represented by red and blue dots. X axis ranges from the 0.0 to 0.6, the Y axis also range from the 0.0 to 0.6. Most of the red dots are concentrated towards the tops left corners of graph, indicate higher values to the y axis and lower values on the x axis. In contrast, blue dots are primarily clustered at the bottom of graph with low y-values but spread across various x-values.



Fig(c)

It displays data points representing actual labels and predicted probabilities. The x-axis is labeled “Data Point” and range from the 0 to 3000, the y axis ranges from 0.0 to 1.0 but is not labeled. The actual labels are depicted as red dots, densely scattered across the plot, while the predicted probabilities are shown as blue dots, which are less in number and appear mainly around the y-value of approximately 0.6.

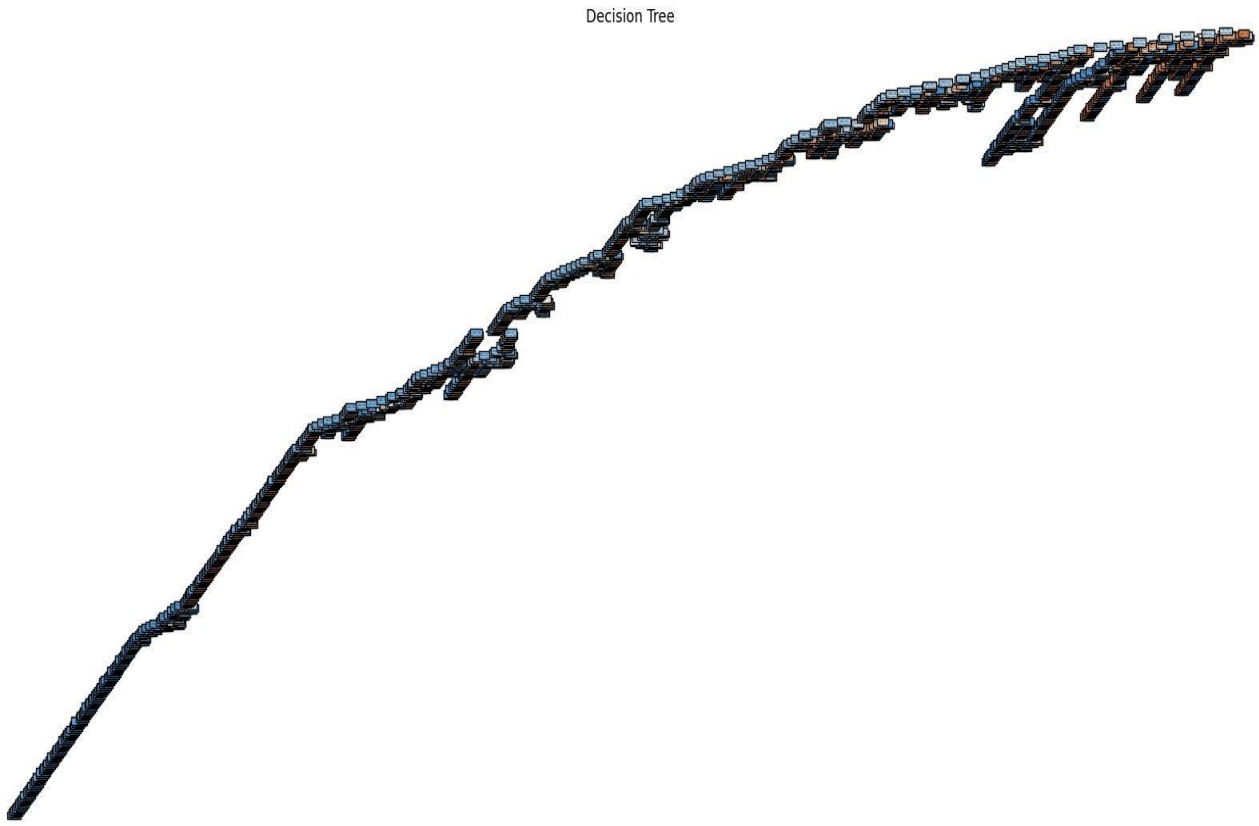
2. Support vector machines: Effective for Binary classifications task like sentiment analysis. Worked on findings the hyperplanes that best separate positives and negatives sentiment instances.



Fig(d)

Support Vector Machine (SVM) decision boundary with hyperplane. It’s a scatter plot with two features, feature 1 and Feature 2, plotted on the x and y-axes respectively. The plot contains two distinct groups of data points, colored red and blue. A black line, representing the SVM hyperplane, effectively separates these two groups of data points. In summary, the image illustrates how an SVM can classify data into distinct categories using a decision boundary.

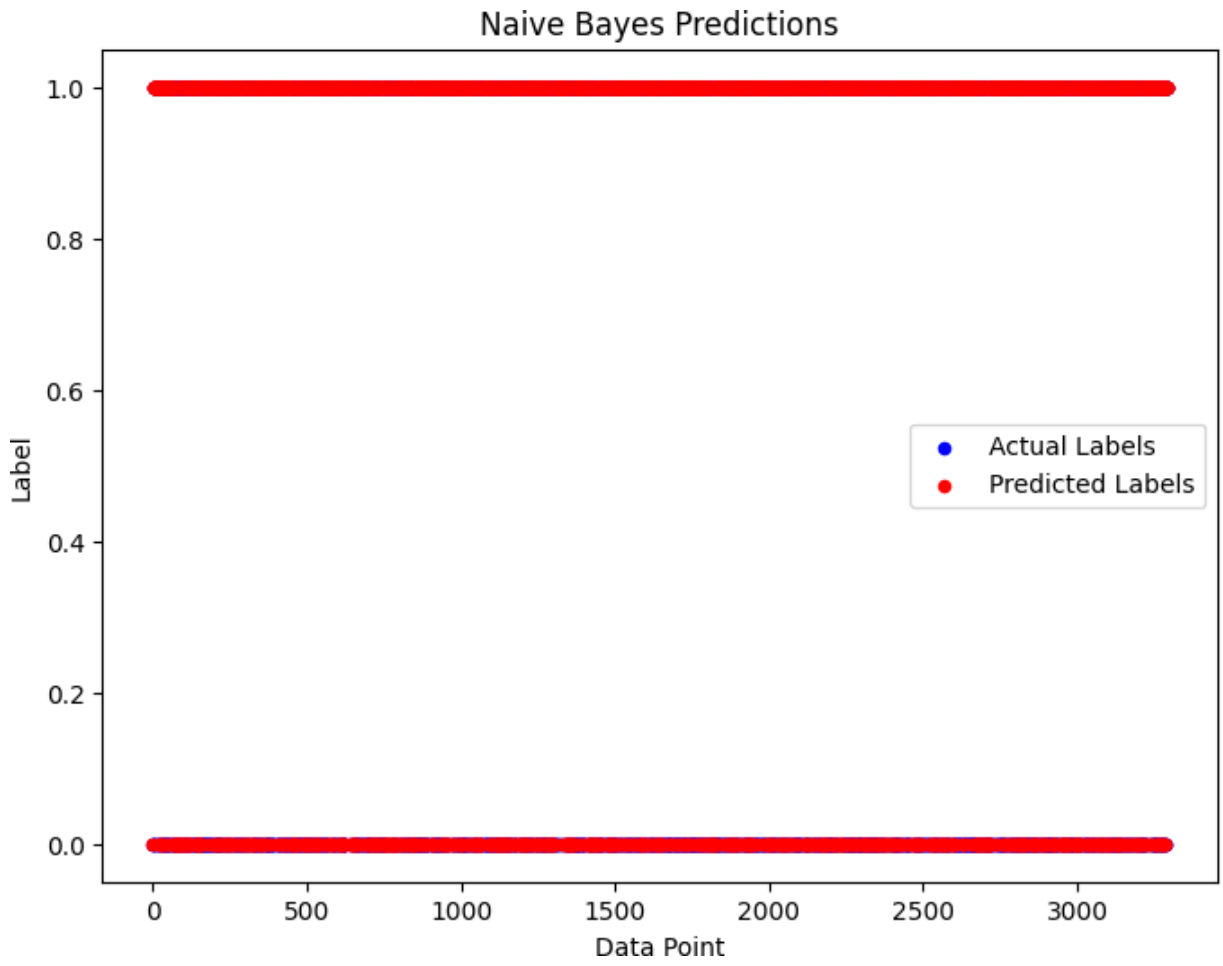
3. Decision tree: Predictive model that used a tree like graphs of the decision and the possible of consequence. It's used in classification and regression tasks, making it a versatile tool for understanding and interpreting data patterns.



Fig(e)

A popular tool in data analytics for decision-making. Unlike typical geometric shapes, the nodes in this tree are formed by intricate structures resembling mechanical or metallic parts. Trees start with a single root's node and branches into multiple paths, indicating different decisions or outcomes. Overall, the image highlights the complex pathways and outcomes involved in decision-making processes.

4. Naive bayes: Naive bayes is the probabilistic classifiers based on the Bayes theorems with on assumptions of the independence between the feature. It's efficient for text classifications and work well with the large datasets.
Naive Bayes Accuracy: 0.673246279987853

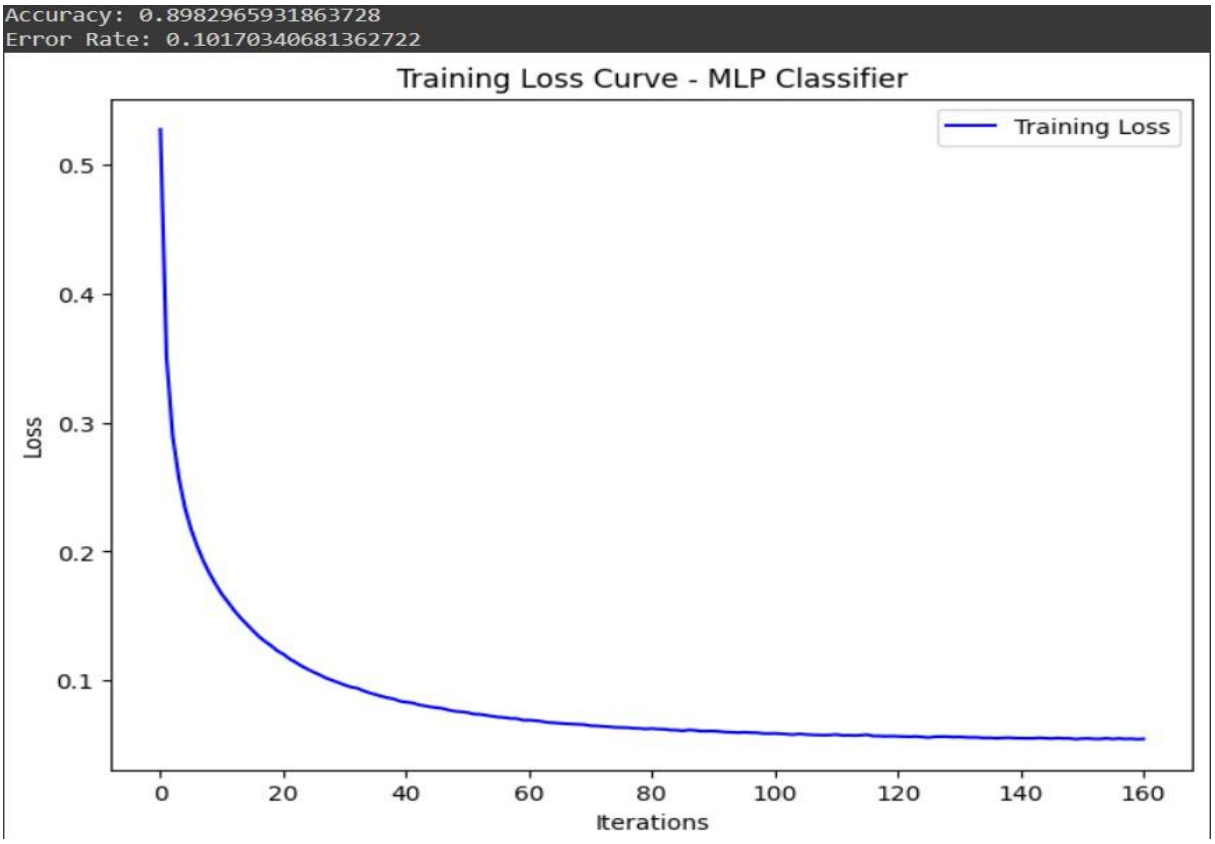


Fig(f)

Features: The x-axis represents data points, ranges from the 0 to 3000.
Labels: The y axis represents to label, ranging from 0.0 to 1.0.
Predictions: Two horizontal lines at label 0.0 and label1.0 indicate predicted labels.

Actual Labels: Scattered points represent actual labels (blue dots)

5. Multi-Layer Perceptron's: Multi-Layer Perceptron's (MLPs) are artificial composed of multiple layers of nodes, including input, hidden, and output layers. They're effective for complex pattern recognition tasks, such as image and speech recognition, using backpropagation for training and learning features hierarchically.



Fig(g)

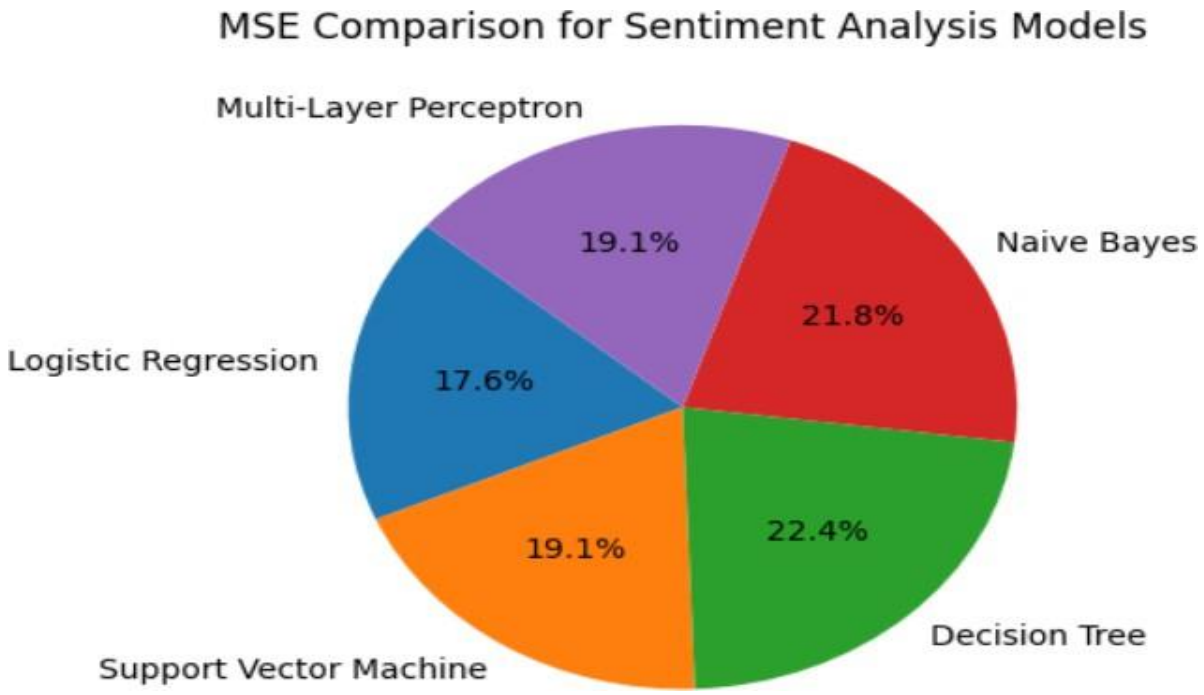
A training loss curve for an MLP Classifier, with x-axis representing data points, y-axis showing the error metric, red lines indicating predicted labels, and blue dots representing actual labels. It's a tool to assess the MLP Classifier's performance and refine it for better accuracy in machine learning.

Classification Report:

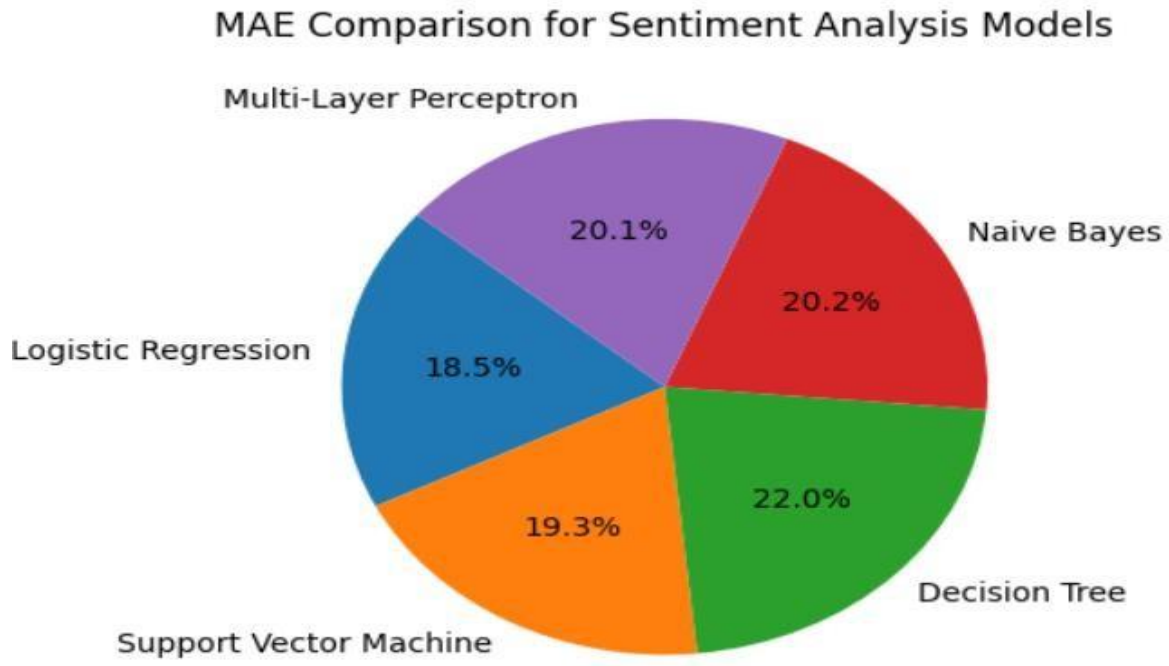
S.NO	Algorithm model	Accuracy's	Error rate	F1 Score	precision	recall	Support
1	Logistic Regression	0.73	0.27	0.94	0.91	0.96	1640

2	Support vector machine	0.73	0.27	0.93	0.92	0.95	1640
3	Decision Tree	0.60	0.40	0.92	0.92	0.92	1640
4	Naive Bayes	0.67	0.33	0.39	0.94	0.24	1640
5	Multi-layer Perceptron	0.89	0.11	0.94	0.93	0.95	1640

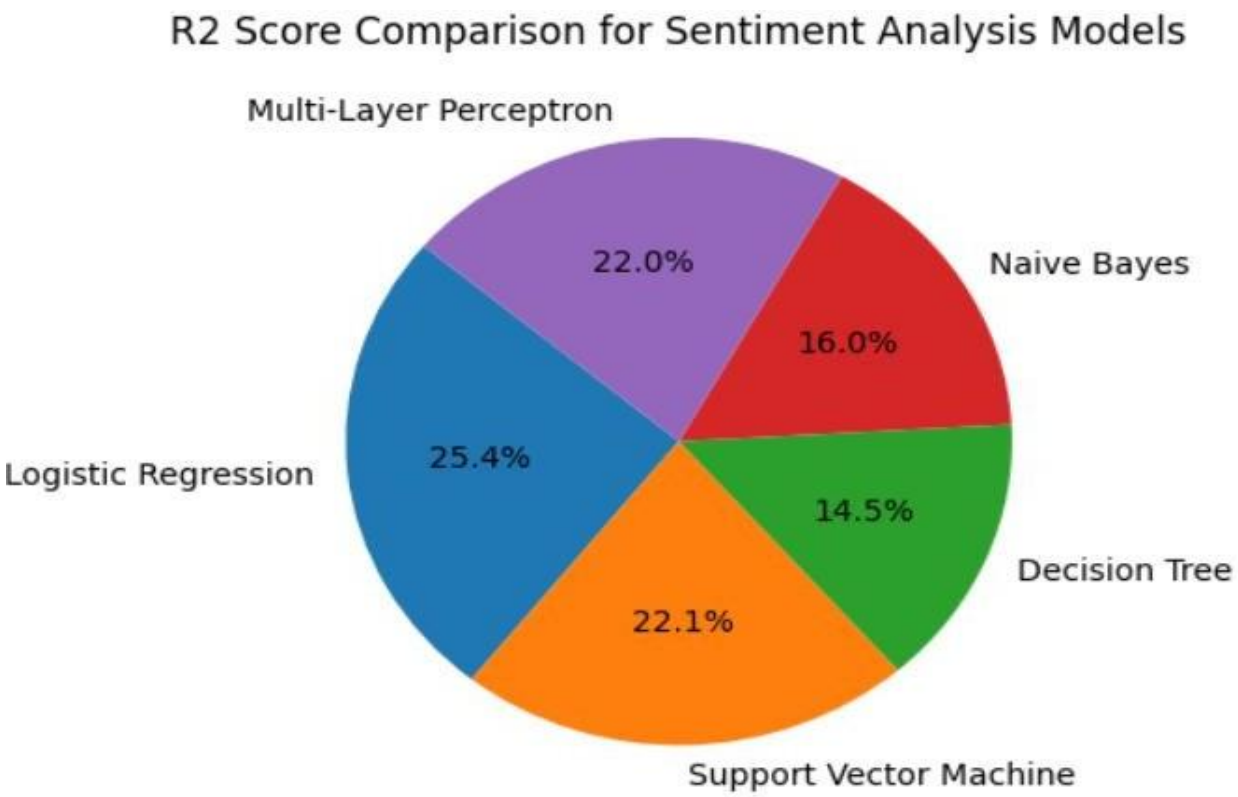
S.NO	**Algorithm Model**	**Mse (Mean squared error)**	**Mae (Mean absolute error)**	**R2 score**
1	logistic regression	0.8351703	0.4734468	0.3871236
2	Support vector machine	0.9038076	0.4939879	0.3367552
3	Decision Tree	1.0626252	0.5646292	0.2202095
4	Naive Bayes	1.0310621	0.5170340	0.2433716
5	Multi-layer Perceptron	0.9058116	0.5150300	0.3352846



Fig(h). MSE Pie Graph



Fig(i) MAE Pie Graph



Fig(j) R2 Score Pie Graph

V. CONCLUSION & FUTURE SCOPE

For a conclusion on your Flipkart reviews sentiment analysis, you could highlight the key findings and insights gained from your research.

In conclusion, they offer valuable insight into the customers perception and the preferences. Through utilization of techniques and Algorithms, we were able to categorize reviews into positives, neutral, and negative sentiments accurately. This analysis not only provides an overview of customer satisfaction levels but also identifies area for the improves in the product and service. Furthermore, integration of tools with e-commerce platforms like Flipkart enhances decision-making processes for businesses, enabling them to tailor their strategies based on customer feedback. This study underscores the significance on sentiment analysis in

understanding consumer behavior on shaping marketing strategies in the digital age.

In future, sentiment analysis for Flipkart reviews will focus on aspect-based analysis (product quality, delivery, etc.), multilingual support, and integrating deep learning for better understanding. Real-time analysis and combining sentiment data with sales and social media insights will drive customer-centric strategies and decision-making.

CODE:

```
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import re
import seaborn as sns
from sklearn.feature_extraction.text import TfidfVectorizer
import matplotlib.pyplot as plt
from wordcloud import WordCloud
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
data = pd.read_csv('flipkart_data.csv')
data.head()
# unique ratings
pd.unique(data['rating'])
```

```
sns.countplot(data=data,
      x='rating',
      order=data.rating.value_counts().index)
# rating label(final)
pos_neg = []
for i in range(len(data['rating'])):
  if data['rating'][i] >= 5:
    pos_neg.append(1)
  else:
    pos_neg.append(0)
data['label'] = pos_neg
from tqdm import tqdm
def preprocess_text(text_data):
  preprocessed_text = []
  for sentence in tqdm(text_data):
    # Removing punctuations
    sentence = re.sub(r'[^\w\s]', '', sentence)
    # Converting lowercase and removing stopwords
    preprocessed_text.append(' '.join(token.lower()
                for token in nltk.word_tokenize(sentence)
                if token.lower() not in stopwords.words('english')))
  return preprocessed_text
import nltk
```

```
nltk.download('punkt')
processed_review = preprocess_text(data['review'].values)
data['review'] = processed_review
data.head()
data["label"].value_counts()
cv = TfidfVectorizer(max_features=2500)
X = cv.fit_transform(data['review'] ).toarray()
X
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, data['label'],
                    test_size=0.33,
                    stratify=data['label'],
                    random_state = 42)
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
# Initialize Decision Tree Classifier
model = DecisionTreeClassifier(random_state=0)

# Train the model
model.fit(X_train, y_train)
# Testing the model on training data
pred_train = model.predict(X_train)
```

```
# Calculate accuracy and error rate
accuracy_train = accuracy_score(y_train, pred_train)
error_rate_train = 1 - accuracy_train
print(f"Accuracy on training data: {accuracy_train:.2f}")
print(f"Error rate on training data: {error_rate_train:.2f}")
import sys
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt
# Set recursion limit to a higher value
sys.setrecursionlimit(10000)
# Initialize Decision Tree classifier
decision_tree_model = DecisionTreeClassifier(random_state=0)
# Train the model
decision_tree_model.fit(X_train, y_train)
# Plot decision tree
plt.figure(figsize=(20,10))
plot_tree(decision_tree_model, filled=True,
feature_names=cv.get_feature_names_out(), class_names=["Negative",
"Positive"])
plt.title("Decision Tree")
plt.show()
from sklearn.metrics import confusion_matrix
from sklearn import metrics
```

```
cm = confusion_matrix(y_train,pred)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = cm,
                                            display_labels = [False, True])
cm_display.plot()
plt.show()
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import numpy as np
# Create a new SVM model
svm_model = SVC(kernel='linear')
# Train the model using the training data
svm_model.fit(X_train, y_train)
# Make predictions on the test data
svm_predictions = svm_model.predict(X_test)
# Evaluate the accuracy of the model
svm_accuracy = accuracy_score(y_test, svm_predictions)
# Calculate error rate
svm_error_rate = 1 - svm_accuracy
# Print the accuracy and error rate
print(f"SVM Accuracy: {svm_accuracy:.2f}")
print(f"SVM Error Rate: {svm_error_rate:.2f}")
# Plotting the SVM decision boundary
```

```
plt.figure(figsize=(8, 6))
# Plot the training data points
plt.scatter(X_train[:, 0], X_train[:, 1], c=y_train,
cmap=plt.cm.coolwarm, s=20, edgecolors='k')
# Get the separating hyperplane
w = svm_model.coef_[0]
a = -w[0] / w[1]
xx = np.linspace(X_train[:, 0].min(), X_train[:, 0].max())
yy = a * xx - (svm_model.intercept_[0]) / w[1]
# Plot the hyperplane
plt.plot(xx, yy, 'k-')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('SVM Decision Boundary with Hyperplane')
plt.show()
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import numpy as np
# Create a new Logistic Regression model
log_reg_model = LogisticRegression()
# Train the model using the training data
log_reg_model.fit(X_train, y_train)
```

```
# Make predictions on the test data
log_reg_predictions = log_reg_model.predict(X_test)
# Evaluate the accuracy of the model
log_reg_accuracy = accuracy_score(y_test, log_reg_predictions)
# Calculate error rate
log_reg_error_rate = 1 - log_reg_accuracy
# Print the accuracy and error rate
print(f"Logistic Regression Accuracy: {log_reg_accuracy:.2f}")
print(f"Logistic Regression Error Rate: {log_reg_error_rate:.2f}")
# Plotting the decision boundary for Logistic Regression
plt.figure(figsize=(8, 6))
# Plot the training data points
plt.scatter(X_train[:, 0], X_train[:, 1], c=y_train,
cmap=plt.cm.coolwarm, s=20, edgecolors='k')
# Create a meshgrid of points covering the feature space
x_min, x_max = X_train[:, 0].min() - 1, X_train[:, 0].max() + 1
y_min, y_max = X_train[:, 1].min() - 1, X_train[:, 1].max() + 1
xx, yy = np.meshgrid(np.linspace(x_min, x_max, 100),
                     np.linspace(y_min, y_max, 100))
#add code for decision boundary plot
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('Logistic Regression Decision Boundary')
```

```
plt.show()

import matplotlib.pyplot as plt

import numpy as np

# Make predictions on the test data

log_reg_probabilities = log_reg_model.predict_proba(X_test)[:, 1]  # Get the probabilities for the positive class

# Plotting the Logistic Regression predictions

plt.figure(figsize=(8, 6))

# Scatter plot of actual labels vs. predicted probabilities

plt.scatter(np.arange(len(y_test)), y_test, color='blue', label='Actual Labels', s=20)

plt.scatter(np.arange(len(log_reg_probabilities)), log_reg_probabilities, color='red', label='Predicted Probabilities', s=20)

plt.xlabel('Data Point')

plt.ylabel('Probability')

plt.title('Logistic Regression Predictions')

plt.legend()

plt.show()

from sklearn.naive_bayes import GaussianNB

from sklearn.metrics import accuracy_score

import matplotlib.pyplot as plt

import numpy as np

# Create a new Naive Bayes model (Gaussian)
```

```
naive_bayes_model = GaussianNB()

# Train the model using the training data

naive_bayes_model.fit(X_train, y_train)

# Make predictions on the test data

naive_bayes_predictions = naive_bayes_model.predict(X_test)

# Evaluate the accuracy of the model

naive_bayes_accuracy = accuracy_score(y_test,
naive_bayes_predictions)

# Calculate error rate

naive_bayes_error_rate = 1 - naive_bayes_accuracy

# Print the accuracy and error rate

print(f"Naive Bayes Accuracy: {naive_bayes_accuracy:.2f}")

print(f"Naive Bayes Error Rate: {naive_bayes_error_rate:.2f}")

# Plotting the Naive Bayes predictions

plt.figure(figsize=(8, 6))

# Scatter plot of data points with actual labels

plt.scatter(np.arange(len(y_test)), y_test, color='blue', label='Actual
Labels', s=20)

# Scatter plot of data points with predicted labels

plt.scatter(np.arange(len(naive_bayes_predictions)),
naive_bayes_predictions, color='red', label='Predicted Labels', s=20)

plt.xlabel('Data Point')

plt.ylabel('Label')
```

```
plt.title('Naive Bayes Predictions')
plt.legend()
plt.show()
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
# Load the dataset
df = pd.read_csv('/content/flipkart_data.csv')
# Create a sentiment column based on the rating
df['sentiment'] = df['rating'].apply(lambda x: 'positive' if x >= 4 else 'negative')
# Vectorize the text data
vectorizer = CountVectorizer(max_features=1000)
X = vectorizer.fit_transform(df['review'])
y = df['sentiment']
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Initialize and train the MLP classifier
```

```
mlp_classifier = MLPClassifier(hidden_layer_sizes=(100,),
max_iter=300, random_state=42)
mlp_classifier.fit(X_train, y_train)
# Make predictions on the test data
y_pred = mlp_classifier.predict(X_test)
# Calculate accuracy and error rate
accuracy = accuracy_score(y_test, y_pred)
error_rate = 1 - accuracy
print(f"Accuracy: {accuracy}")
print(f"Error Rate: {error_rate}")
# Plot the training loss curve
train_loss = mlp_classifier.loss_curve_
plt.figure(figsize=(8, 6))
plt.plot(train_loss, label='Training Loss', color='blue')
plt.xlabel('Iterations')
plt.ylabel('Loss')
plt.title('Training Loss Curve - MLP Classifier')
plt.legend()
plt.show()
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier

from sklearn.naive_bayes import MultinomialNB

from sklearn.neural_network import MLPClassifier

from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

import matplotlib.pyplot as plt

# Load the dataset

data = pd.read_csv('flipkart_data.csv')

# Assuming the actual column name for review text is 'review' instead of 'Text'

# Update the code accordingly

# Preprocess the data

# Assuming you have a column 'review' containing the review text and 'rating' containing sentiment labels

# Clean text, encode sentiment labels (e.g., Positive as 1, Negative as 0)

# Split the data into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(data['review'], data['rating'], test_size=0.2, random_state=42)

# Vectorize the text data using CountVectorizer (you can use other vectorization methods as well)

vectorizer = CountVectorizer()

X_train_vectorized = vectorizer.fit_transform(X_train)

X_test_vectorized = vectorizer.transform(X_test)
```

```
# Initialize and train multiple classification models
models = {
    'Logistic Regression': LogisticRegression(),
    'Support Vector Machine': SVC(kernel='linear'),
    'Decision Tree': DecisionTreeClassifier(),
    'Naive Bayes': MultinomialNB(),
    'Multi-Layer Perceptron': MLPClassifier()
}
# Initialize dictionaries to store metrics
mse_values = {}
mae_values = {}
r2_values = {}
for model_name, model in models.items():
    model.fit(X_train_vectorized, y_train)
    y_pred = model.predict(X_test_vectorized)
    mse = mean_squared_error(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    mse_values[model_name] = mse
    mae_values[model_name] = mae
    r2_values[model_name] = r2
# Print the metrics for each model
print("MSE values:")
```

```
for model_name, mse in mse_values.items():
    print(f"{model_name}: {mse}")
print("\nMAE values:")
for model_name, mae in mae_values.items():
    print(f"{model_name}: {mae}")
print("\nR2 Score values:")
for model_name, r2 in r2_values.items():
print(f"{model_name}: {r2}")
# Plot the MSE values in a pie chart
plt.figure(figsize=(16, 6))
plt.subplot(1, 3, 1)
plt.pie(mse_values.values(), labels=mse_values.keys(),
autopct='%1.1f%%', startangle=140)
plt.title('MSE Comparison for Sentiment Analysis Models')
# Plot the MAE values in a pie chart
plt.subplot(1, 3, 2)
plt.pie(mae_values.values(), labels=mae_values.keys(),
autopct='%1.1f%%', startangle=140)
plt.title('MAE Comparison for Sentiment Analysis Models')
# Plot the R2 Score values in a pie chart
plt.subplot(1, 3, 3)
plt.pie(r2_values.values(), labels=r2_values.keys(), autopct='%1.1f%%',
startangle=140)
```

```
plt.title('R2 Score Comparison for Sentiment Analysis Models')
plt.tight_layout()
plt.show()
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
from textblob import TextBlob
data = pd.read_csv('flipkart_data.csv')
# Convert data to DataFrame
df = pd.DataFrame(data)
# Perform sentiment analysis using TextBlob
df['sentiment'] = df['review'].apply(lambda x:
TextBlob(x).sentiment.polarity)
# Calculate correlation matrix
corr_matrix = df[['rating', 'sentiment']].corr()
# Create heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f",
linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
import pandas as pd
from sklearn.model_selection import train_test_split
```

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report
import matplotlib.pyplot as plt
# Load the dataset
df = pd.read_csv('/content/flipkart_data.csv')
# Create a sentiment column based on the rating
df['sentiment'] = df['rating'].apply(lambda x: 'positive' if x >= 4 else 'negative')
# Vectorize the text data
vectorizer = CountVectorizer(max_features=1000)
X = vectorizer.fit_transform(df['review'])
y = df['sentiment']
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Initialize and train the classifiers
log_reg_model = LogisticRegression()
log_reg_model.fit(X_train, y_train)
```

```
svm_model = SVC(kernel='linear')
svm_model.fit(X_train, y_train)
decision_tree_model = DecisionTreeClassifier(random_state=0)
decision_tree_model.fit(X_train, y_train)
naive_bayes_model = GaussianNB()
naive_bayes_model.fit(X_train.toarray(), y_train)  # Naive Bayes requires dense arrays
mlp_classifier = MLPClassifier(hidden_layer_sizes=(100,), max_iter=300, random_state=42)
mlp_classifier.fit(X_train, y_train)
# Make predictions on the test data
log_reg_predictions = log_reg_model.predict(X_test)
svm_predictions = svm_model.predict(X_test)
decision_tree_predictions = decision_tree_model.predict(X_test)
naive_bayes_predictions = naive_bayes_model.predict(X_test.toarray())  # Convert to dense array for Naive Bayes
mlp_predictions = mlp_classifier.predict(X_test)
# Generate and print the classification report for each classifier
classifiers = {
    'Logistic Regression': log_reg_predictions,
    'SVM': svm_predictions,
    'Decision Tree': decision_tree_predictions,
    'Naive Bayes': naive_bayes_predictions,
```

```
    'MLP Classifier': mlp_predictions
}
for clf_name, predictions in classifiers.items():
    class_report = classification_report(y_test, predictions)
    print(f"Classification Report for {clf_name}:")
    print(class_report)
    print("="*50)
# Plot the training loss curve for MLP Classifier
train_loss = mlp_classifier.loss_curve_
plt.figure(figsize=(8, 6))
plt.plot(train_loss, label='Training Loss', color='blue')
plt.xlabel('Iterations')
plt.ylabel('Loss')
plt.title('Training Loss Curve - MLP Classifier')
plt.legend()
plt.show()
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
```

```
# Load your dataset (replace 'path_to_your_dataset.csv' with the actual file path)

data = pd.read_csv('flipkart_data.csv')

# Assuming your dataset has 'review' column for text and 'rating' column for sentiment label

X = data['review']

y = data['rating']

# Get unique classes in your dataset as strings

unique_classes = sorted(y.unique(), key=str)

# Split the data into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Vectorize the text data using CountVectorizer

vectorizer = CountVectorizer()

X_train_vec = vectorizer.fit_transform(X_train)

X_test_vec = vectorizer.transform(X_test)

# Initialize Decision Tree classifier

decision_tree_model = DecisionTreeClassifier(random_state=0)

# Train the model

decision_tree_model.fit(X_train_vec, y_train)

# Predict sentiment labels on the test data

y_pred = decision_tree_model.predict(X_test_vec)

# Calculate accuracy
```

```
accuracy = accuracy_score(y_test, y_pred)

print(f"Accuracy: {accuracy}")

# Calculate error rate

error_rate = 1 - accuracy

print(f"Error Rate: {error_rate}")

# Plot decision tree

plt.figure(figsize=(20, 10))

plot_tree(decision_tree_model, filled=True,
feature_names=vectorizer.get_feature_names_out(), class_names=[str(c)
for c in unique_classes])

plt.title("Decision Tree")

plt.show()
```