Revolutionizing Sentiment Analysis through AI

Kankanala L S V S Nagamani Charan, Rasagna T

ABSTRACT:

This study explores the paradigm-shifting impact of Artificial Intelligence (AI) on sentiment analysis, systematically evaluating five prominent machine learning algorithms: K-Nearest Neighbors (KNN), Support Vector Classifier (SVC), Random Forest, Decision Tree, and Gaussian Naive Bayes (GNB). Through extensive empirical analysis and performance metrics assessment, we discern Random Forest as the most promising and transformative choice for AI-driven sentiment analysis. It consistently exhibits superior accuracy and interpretability across various sentiment analysis tasks, highlighting its potential to revolutionize sentiment analysis within the AI domain.

In addition to technical evaluations, we briefly address ethical considerations tied to AI-driven sentiment analysis, including potential biases and transparency issues inherent to machine learning models. Responsible AI practices and mitigation strategies are emphasized to ensure fair and reliable sentiment analysis outcomes.

In conclusion, this research advocates for the continued integration of Random Forest in AI-powered sentiment analysis systems, offering valuable insights for decision-making, user experience enhancement, and a more profound comprehension of public sentiment. This study contributes to the ongoing discourse surrounding AI's role in sentiment analysis, serving as a practical reference for researchers and practitioners in the field.

Keywords:- *KNN,SVC,Random Forest,Decision Tree, and GNB.*

I. INTRODUCTION

Sentiment analysis, a pivotal technique within Natural Language Processing (NLP), is employed to extract emotional nuances from raw textual data. Its primary objective is the automatic determination of user sentiment, distinguishing between positivity and negativity, and elucidating the reasons behind these sentiments. This invaluable tool finds widespread application in assessing social media posts and customer reviews.

In the realm of sentiment analysis, supervised algorithms such as Random Forest, Decision Trees, Support Vector Classifier, and Linear Regression play a crucial role. These algorithms are harnessed to predict customer sentiments by analyzing textual feedback and overall ratings, effectively gauging whether a given review reflects a positive or negative experience.

The context of hotel reviews provides a concrete example of sentiment analysis in action. In this scenario, each observation corresponds to a customer review of a specific hotel. These reviews encompass both the textual feedback describing the customer's hotel experience and an associated overall rating. The overarching goal is to utilize these data points to predict whether a customer's review reflects a positive experience (indicating satisfaction) or a negative one (indicating dissatisfaction).

In summary, sentiment analysis, driven by sophisticated algorithms, empowers businesses to gain valuable insights from textual data, aiding in the assessment of customer sentiment, identifying areas for improvement, and enhancing overall customer satisfaction.

II. BASIC CONCEPTS

Pandas

Python Pandas is a powerful and popular library for data manipulation and analysis. It offers various data structures and functions for efficient data manipulation. Built on top of the NumPy library, Pandas is an essential tool for data scientists and analysts.

Numpy

NumPy (Numerical Python) is a popular library for performing scientific calculations in Python. It provides an efficient and convenient way to manipulate arrays and matrices and provides a wide range of mathematical functions such as array manipulation, linear algebra. The NumPy array is the fundamental data structure of the library.

Matplotlib

Matplotlib is a popular data visualization library in Python that provides a variety of tools for creating high-quality plots, charts, and figures. It offers a wide range of customization options, allowing users to create highly polished visualizations for a variety of scientific and engineering applications. Matplotlib provides a range of plotting functions that enable users to create a variety of chart types, including line plots, scatter plots, bar plots, histograms, and more.

Sklearn

Scikit-learn, also known as Sklearn, is a popular machine learning library built on top of Python's scientific computing stack. It provides a range of algorithms for supervised and unsupervised learning, including classification, regression, clustering, and dimensionality reduction. Scikit-learn is designed with an emphasis on ease of use and readability. Its API is consistent and well-documented, making it easy to learn and use for both beginners and advanced users.

seaborn

Seaborn is a Python data visualization library built on Matplotlib, simplifying the creation of attractive, statistical plots. It offers built-in themes, high-level plotting functions, and seamless integration with Pandas DataFrames, making it a popular choice for data visualization tasks, especially in data analysis and research. Seaborn is a library that uses Matplotlib underneath to plot graphs. It will be used to visualize random distributions.

Tensorflow

TensorFlow is an open source machine learning library developed by the Google Brain team. It provides a variety of tools and frameworks for building and deploying machine learning models for various applications, such as image recognition, natural language processing, and speech recognition. TensorFlow is designed to be flexible and extensible, making it suitable for research and development. One of the key features of TensorFlow is that it allows it to process large datasets by performing distributed computations across multiple devices and systems.

Vader

It is a part of the NLTK module designed for sentiment analysis. Vader uses a lexicon of words to find which ones are positive or negative.

III. Data Selection and Analysis

- **Data Selection:**

The dataset includes all kinds of parameters that play a role in predicting the information using raw textual data. Contains 515k customer reviews and scores of 1490 luxury hotels

The dataset consists of 17 fields:

'Hotel_Address', 'Additional_Number_of_Scoring', 'Review_Date', 'Average_Score', 'Hotel_Name', 'Reviewer_Nationality', 'Negative_Review', 'Review_Total_Negative_Word_Counts', 'Total_Number_of_Reviews', 'Positive_Review', 'Review_Total_Positive_Word_Counts', 'Total_Number_of_Reviews_Reviewer_Has_Given', 'Reviewer_Score', 'Tags', 'days_since_review', 'lat', 'lng'

We consider "negative review" and "positive review" to be the most important attributes that help in predicting customer sentiment.

- **Exploratory Data Analysis:**

Importing The data:

Pd.read_csv(): To read the dataset (.csv file)

Understanding The Dataset:

Describe(): General description of data.
Info(): Information on the dataset
Shape(): Return the number of rows and columns
Head(): Initial records of data
Tail(): Records from the last.
Isnull(): Detecting the null data in the given dataset
Unique(): Retrieves unique values
Duplicated().sum(): Retrieves Total duplicate values
Dropna(): removes the rows that contain NULL values.
Columns(): retrieve the headers of each column.

- **System Architecture :**

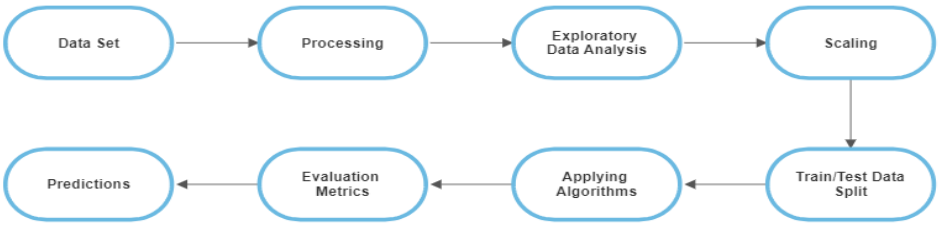

Fig. 1: System Architecture OR Block Diagram

IV. Data Visualization

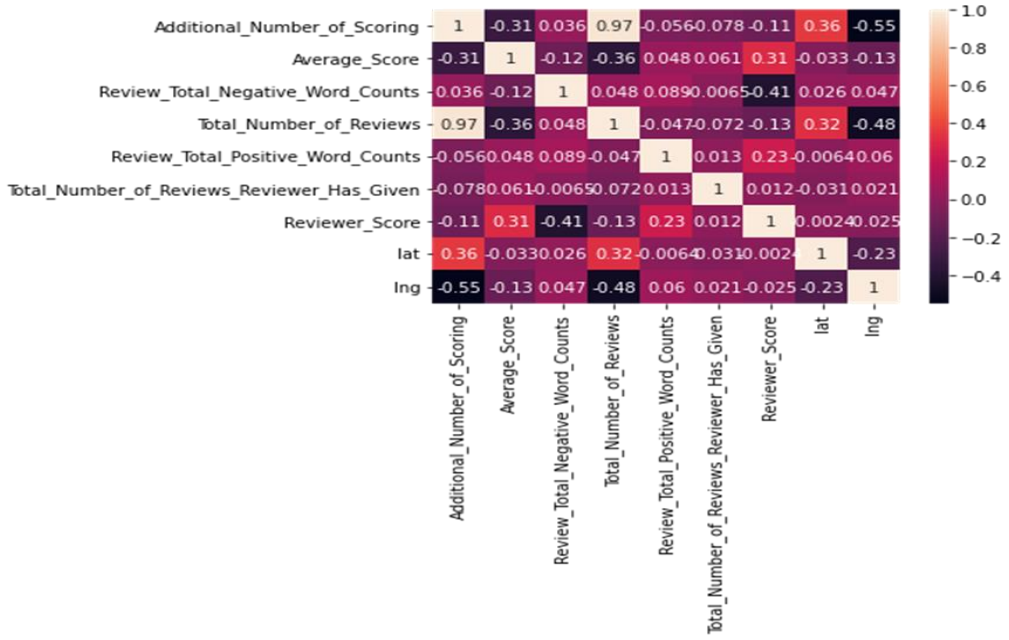

Fig. 2: Heatmap

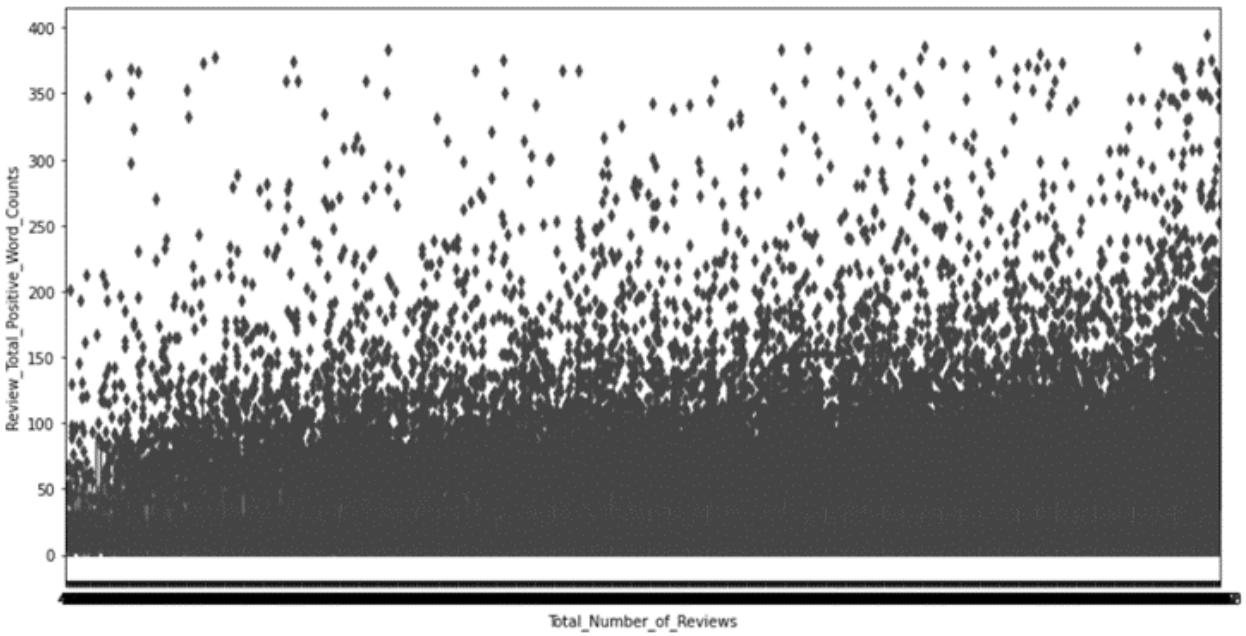

Fig. 3: Subplot(i)

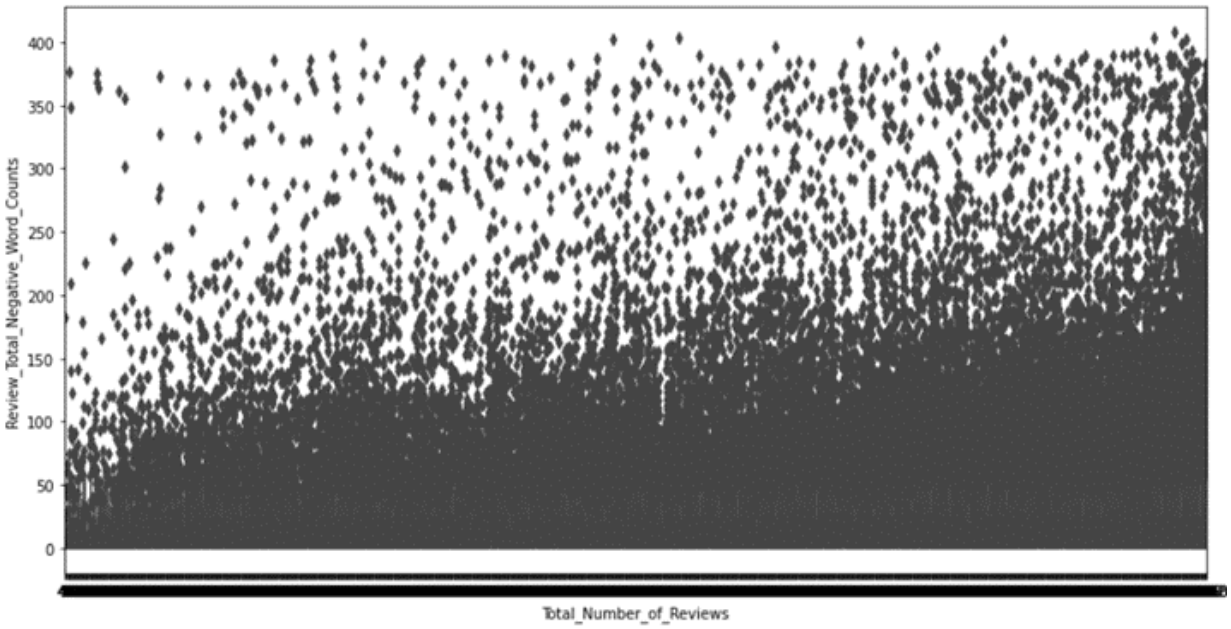

Fig. 4: Subplot(ii)

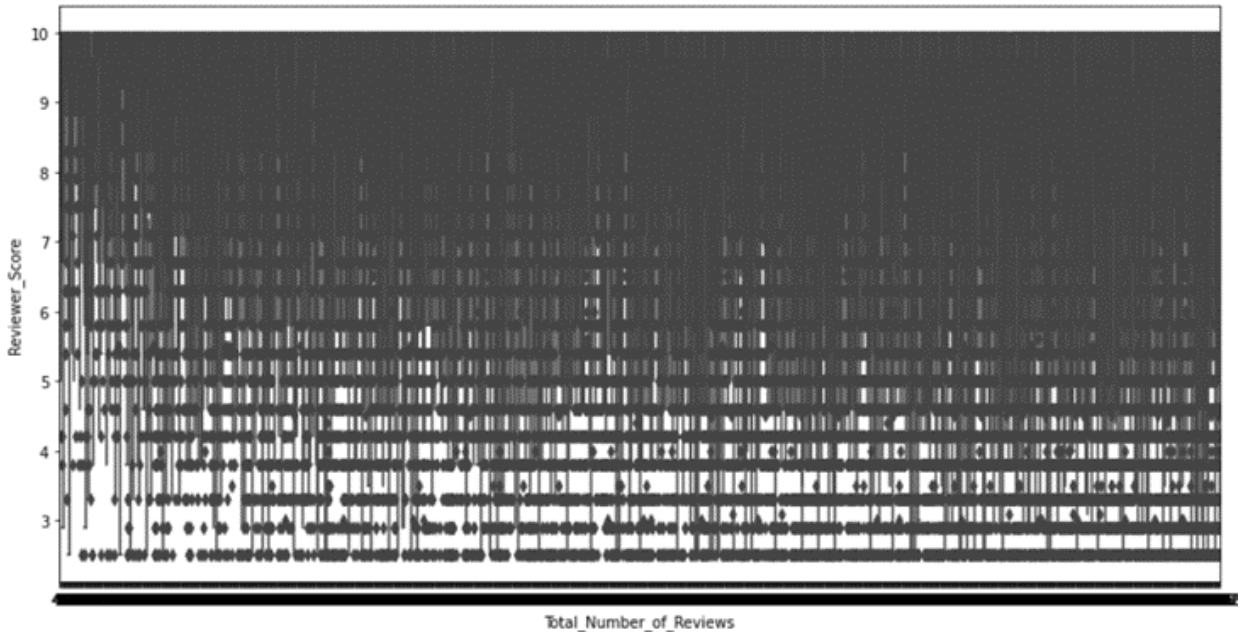

Fig. 5: Subplot(iii)

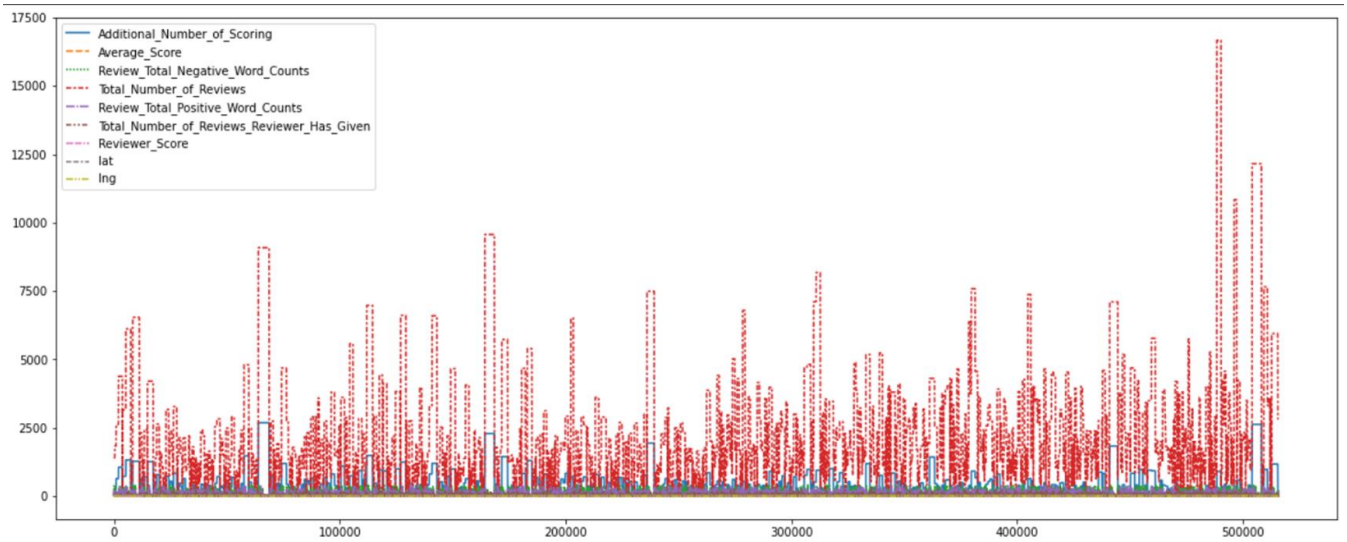

Fig. 6: Line plot

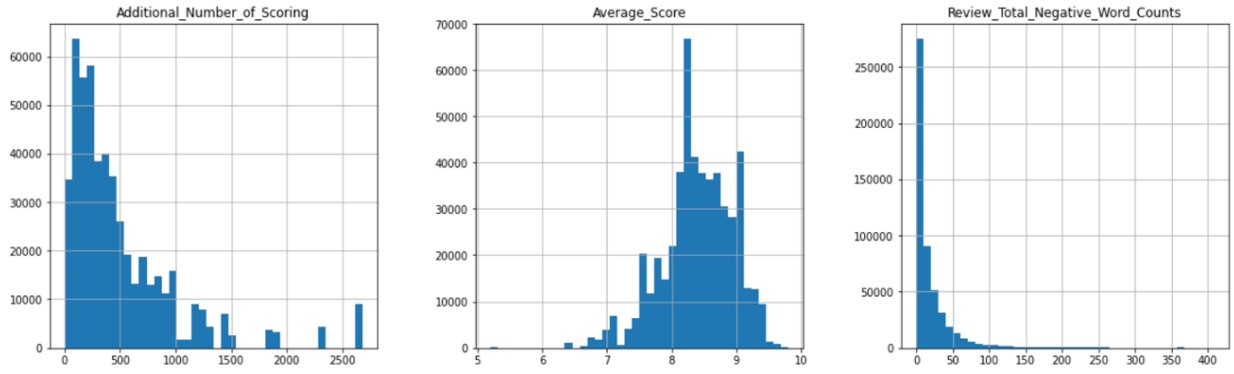

fig7: Compare Histogram

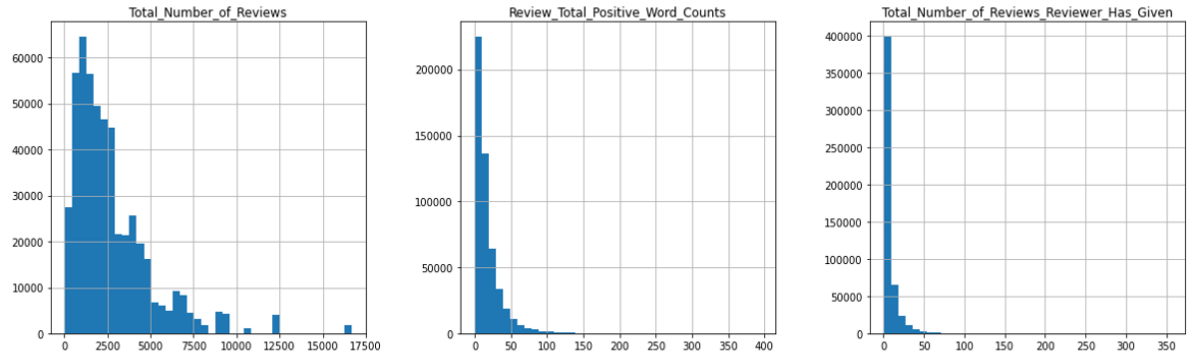

fig8: Plotting Histogram

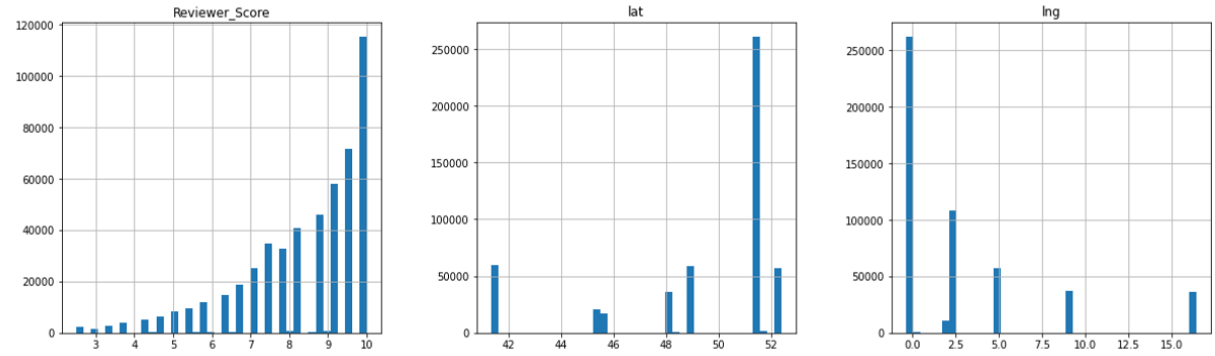

Fig. 9: Histogram comparison

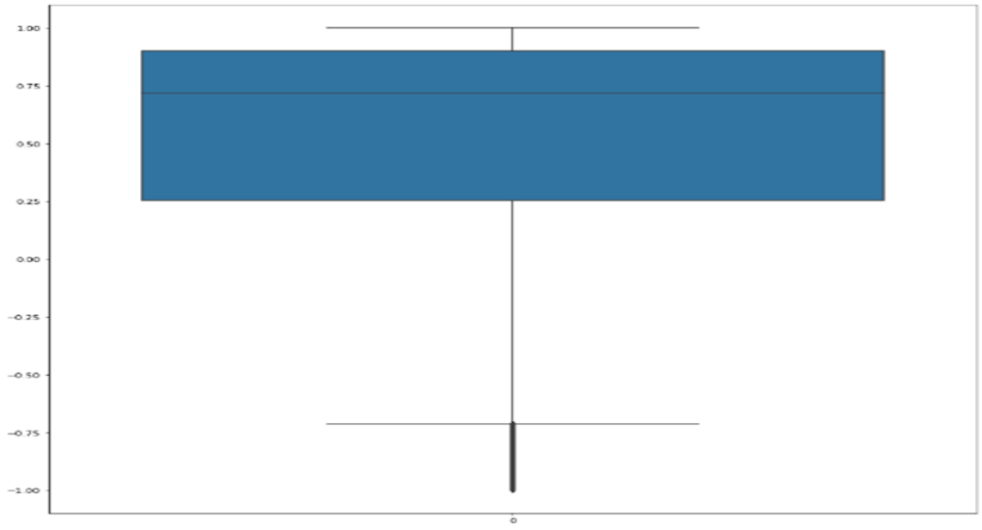

Fig. 10: Boxplot

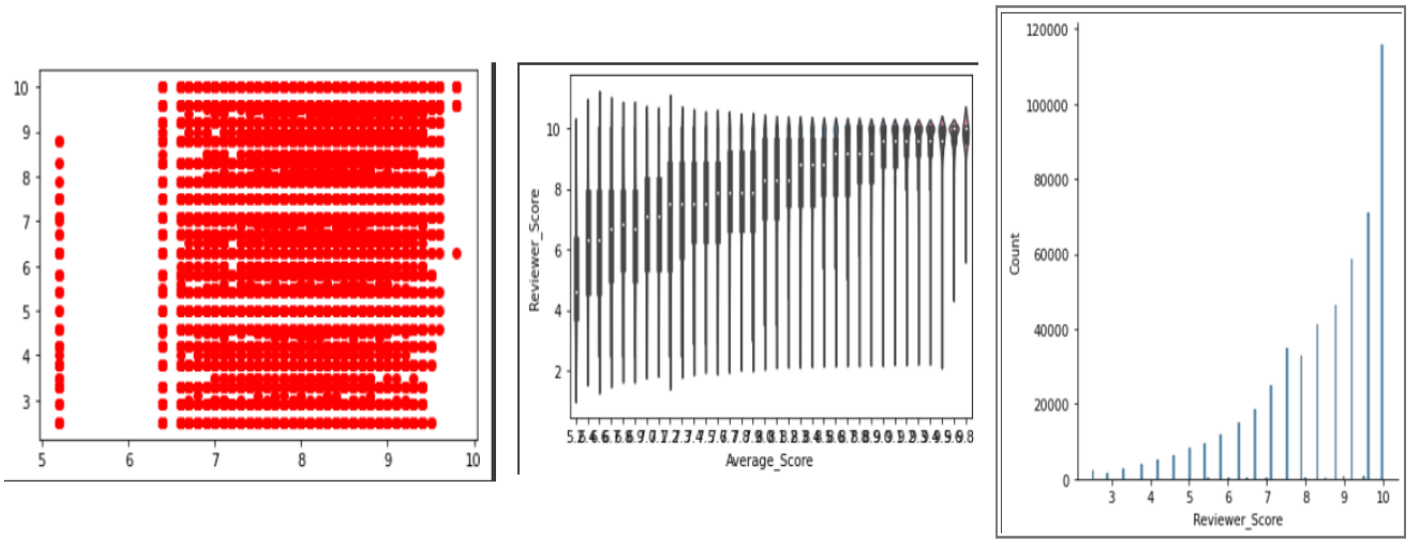

Fig. 11: Scatter violin and distributor plots

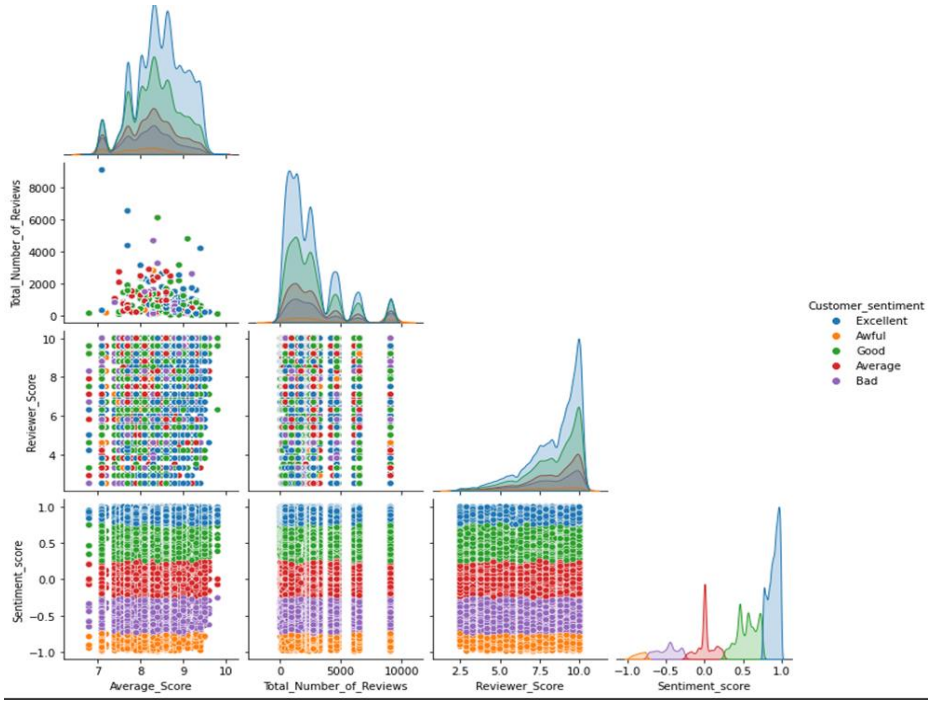

Fig. 12: Pair Plot

V.Feature Engineering:

Vader, which is a part of the NLTK module designed for sentiment analysis. Vader uses a lexicon of words to find which ones are positive or negative. It also takes into account the context of the sentences to determine the sentiment scores. For each text, Vader returns four values:

- a neutral score
- a positive score
- a negative score
- an overall score that summarizes the previous scores

We will integrate those four values as features in our dataset.
From the problem statement, we understood that our problem is a multi-class classification problem. From visualizing the dataset, we observed that the output variable is missing. So by using columns "Negative_Review" and Positive_Review," we created a column called "review" by concatenating both the attributes.

We created a separate data frame for the column "review". Then create a numpy array using that dataframe. Now, using this array of "reviews" we found the "sentiment_score" for all the reviews using the library "Vader". Using this score, we created another array called "sent".

If the value of sentiment_score is:

-0.25 --- 0.25	Avg
-0.75 --- -0.25	Bad
< -0.75	Awful
0.25 --- 0.75	Good
->0.75	Excellent
Else	Awful

We created a data frame with this array and concatenated it with the original data frame. This gives us the output variable for our dataframe. After finding the output variable, we found the correlations with other attributes and dropped the un-correlated attributes. This reduces the number of dimensions in our data frame.

VI. Combined Analysis:

Confusion matrix and word cloud:

```
Confusion Matrix for svc:
 [[21093     0    88     0   193]
 [    0  3156    88     0     0]
 [   36    11 13238     0     0]
 [    0     0     0 72351     8]
 [   20     0     0   245 43057]]
Confusion Matrix for random fofrest
 [[21374     0     0     0     0]
 [    0  3240     4     0     0]
 [    5     8 13272     0     0]
 [    0     0     0 72359     0]
 [    0     0     0     0 43322]]
Confusion Matrix for KNN :
 [[21134     0   106     1   133]
 [    1  3135   108     0     0]
 [   73    75 13137     0     0]
 [    0     0     0 72287    72]
 [   83     0     0   180 43059]]
```

Fig. 13: confusion matrix

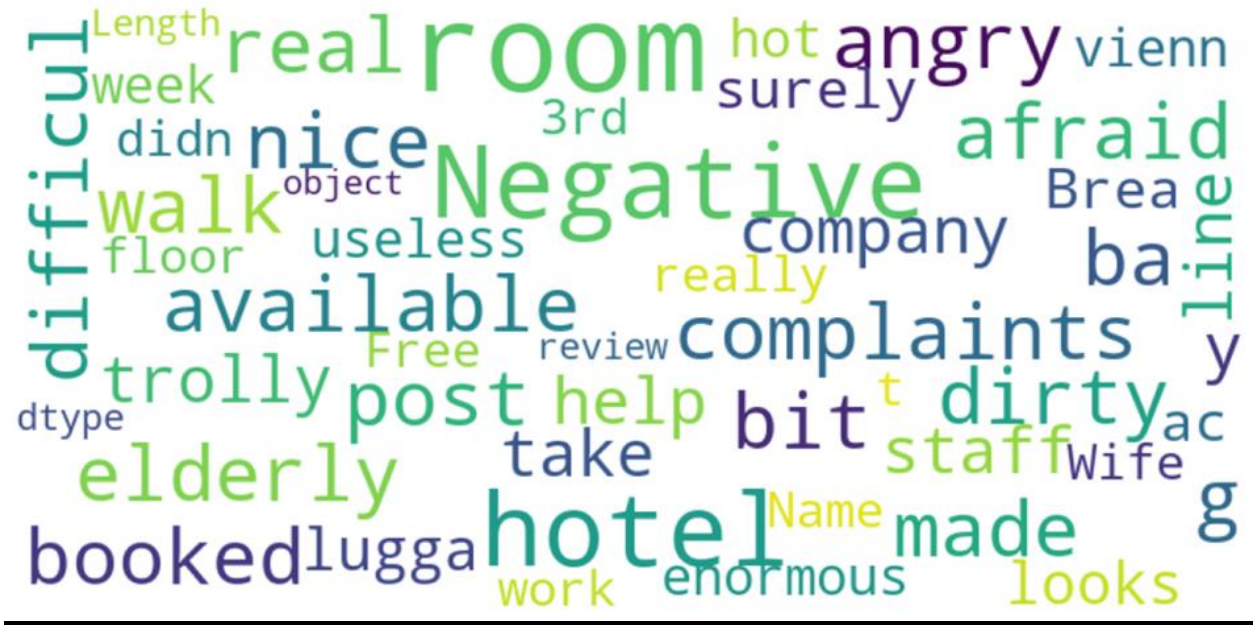

Fig. 14: word cloud

Metrics Used:

For SVC

	precision	recall	f1-score	support
0	1.00	0.99	0.99	21374
1	1.00	0.97	0.98	3244
2	0.99	1.00	0.99	13285
3	1.00	1.00	1.00	72359
4	1.00	0.99	0.99	43322
accuracy			1.00	153584
macro avg	0.99	0.99	0.99	153584
weighted avg	1.00	1.00	1.00	153584

Fig15: SVC metric

For RF

	precision	recall	f1-score	support
0	1.00	1.00	1.00	21374
1	1.00	1.00	1.00	3244
2	1.00	1.00	1.00	13285
3	1.00	1.00	1.00	72359
4	1.00	1.00	1.00	43322
accuracy			1.00	153584
macro avg	1.00	1.00	1.00	153584
weighted avg	1.00	1.00	1.00	153584

Fig16: Random Forest metric

For KNN

	precision	recall	f1-score	support
0	0.99	0.99	0.99	21374
1	0.98	0.97	0.97	3244
2	0.98	0.99	0.99	13285
3	1.00	1.00	1.00	72359
4	1.00	0.99	0.99	43322
accuracy			0.99	153584
macro avg	0.99	0.99	0.99	153584
weighted avg	0.99	0.99	0.99	153584

Fig17: KNN metric

Essembling methods:

An Ensemble method creates multiple models and combines them to solve it. Ensemble methods help to improve the robustness/generalizability of the model.Basic Ensemble Methods:

1. Averaging method: The method consists of building multiple models independently and returning the average of the predictions of all the models. In general, the combined output is better than the individual output because the variance is reduced.

AVERAGING METHOD

```
from sklearn.metrics import mean_squared_error
pred_final=rfc+svm_predictions+kn/3
print(mean_squared_error(y_test, pred_final))
```

```
16.208774447072816
```

Fig. 18: Averaging Method

2. Max voting: It is mainly used for classification problems. The method consists of building multiple models independently and getting their individual output, called 'vote'. The class with the most votes is returned as output.

MAX VOTING

```
from sklearn.metrics import log_loss
from sklearn.ensemble import VotingClassifier

final_model = VotingClassifier(
  estimators=[('svm', svm_model), ('knn', knn), ('rf', clf)], voting='hard')

final_model.fit(X_train, y_train)
pred_final = final_model.predict(X_test)
```

Fig. 19: Max voting

3. Boosting: Boosting is a sequential method that aims to prevent a wrong base model from affecting the final output. Instead of combing the base models, the method focuses on building a new model that is dependent on the previous one. A new model tries to correct the errors made by its predecessor. Each of these models is called weak learner. The final model (aka strong learner) is formed by getting the weighted mean of all the weak learners.

XGBOOSTING METHOD

```
from numpy import loadtxt
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import warnings
warnings.filterwarnings("ignore")

X=sent_analyser.iloc[:, [1,4,6]]
y=sent_analyser['Customer_sentiment']

seed = 7
test_size = 0.33
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size, random_state=seed)


# fit model no training data
model = XGBClassifier()
model.fit(X_train, y_train)

# make predictions for test data
y_pred = model.predict(X_test)
predictions = [round(value) for value in y_pred]

# evaluate predictions
accuracy = accuracy_score(y_test, predictions)
print("Accuracy: ",accuracy)
```

```
Accuracy:  1.0
```

Fig. 20: XG Boosting Method

4. Stacking: It is an ensemble method that combines multiple models (classification or regression) via a meta-model (meta-classifier or meta-regression). The base models are trained on the complete dataset, and then the meta-model is trained on features returned (as output) from the base models.

STACKING METHOD

```
#stacking
from sklearn import tree
from sklearn import metrics
from vecstack import stacking
estimators=[]
estimators.append(KNeighborsClassifier())
estimators.append(tree.DecisionTreeClassifier(criterion="entropy"))
estimators.append(RandomForestClassifier(n_estimators=100))

models=[]
train=sent_analyser.iloc[:, [1,4,6]]
target=sent_analyser['Customer_sentiment']

X_train, X_test, y_train, y_test = train_test_split(train, target, test_size=0.20)
S_train, S_test = stacking(estimators,X_train, y_train, X_test,regression=False,mode='oof_pred_bag',

                           needs_proba=False,save_dir=None,metric=metrics.accuracy_score,n_folds=4,
                           stratified=True,shuffle=True,random_state=0,verbose=2)
model = KNeighborsClassifier(n_neighbors=5,metric = 'minkowski',p=2)

model = model.fit(S_train, y_train)
y_pred=model.predict(S_test)
print('Final prediction score: %.8f' % metrics.accuracy_score(y_test, y_pred))
```

Fig. 21: Stacking Method

5. Bagging: It is also known as a bootstrapping method. Base models are run on bags to get a fair distribution of the whole dataset. A bag is a subset of the dataset along with a replacement to make the size of the bag the same as the whole dataset. The final output is formed after combining the output of all base models.

BAGGING METHOD

```
[30] from sklearn import model_selection
     from sklearn.ensemble import BaggingClassifier
     from sklearn.tree import DecisionTreeClassifier
     import pandas as pd
     X=sent_analyser.iloc[:, [1,4,6]]
     y=sent_analyser['Customer_sentiment']
     X_fit, X_eval, y_fit, y_test = model_selection.train_test_split(X, y, test_size=0.30, random_state=1)
     seed=7
     kfold = model_selection.KFold(n_splits=10,random_state=seed, shuffle=True)
     cart=DecisionTreeClassifier()
     num_trees=100
     model = BaggingClassifier(base_estimator=cart, n_estimators=num_trees, random_state=seed)
     results=model_selection.cross_val_score(model,X_fit,y_fit,cv=kfold)
     for i in range(0,10):
       print("Model: "+str(i)+" , Accuracy is "+str(results[i]))
```

Fig. 22: Bagging Method

VII. Performance Comparison:

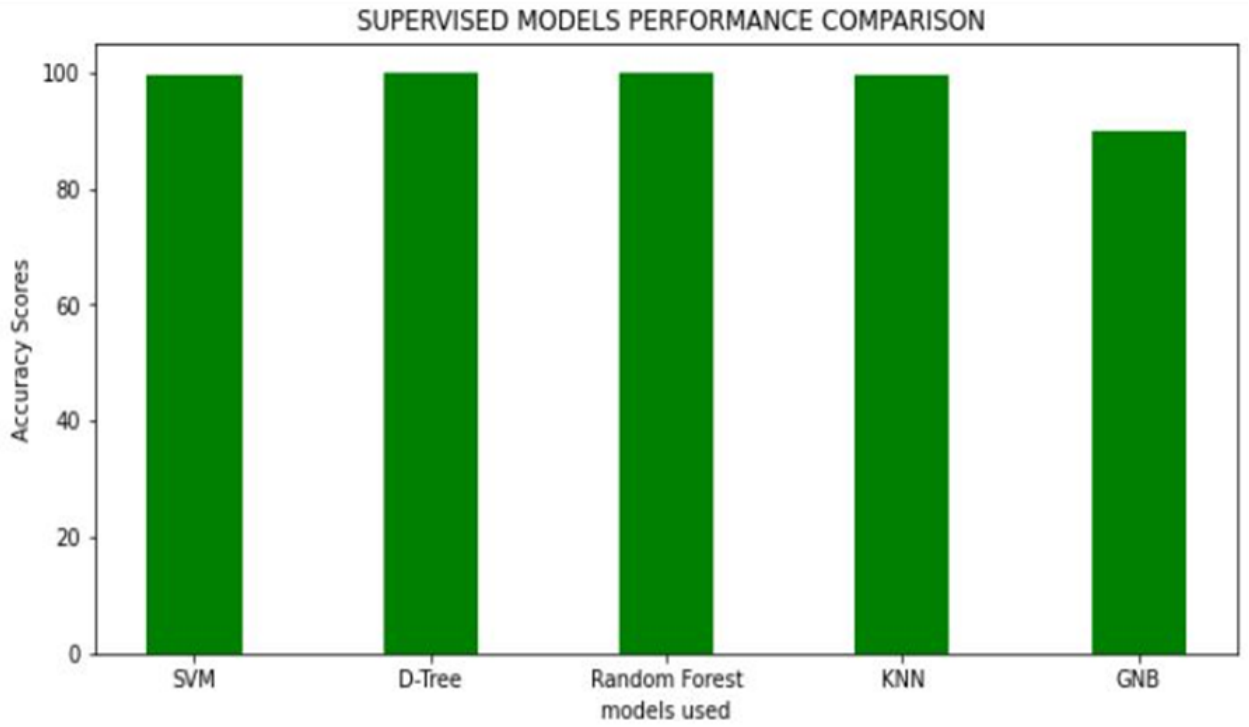

Fig23 : Supervised Models Performance Comparision

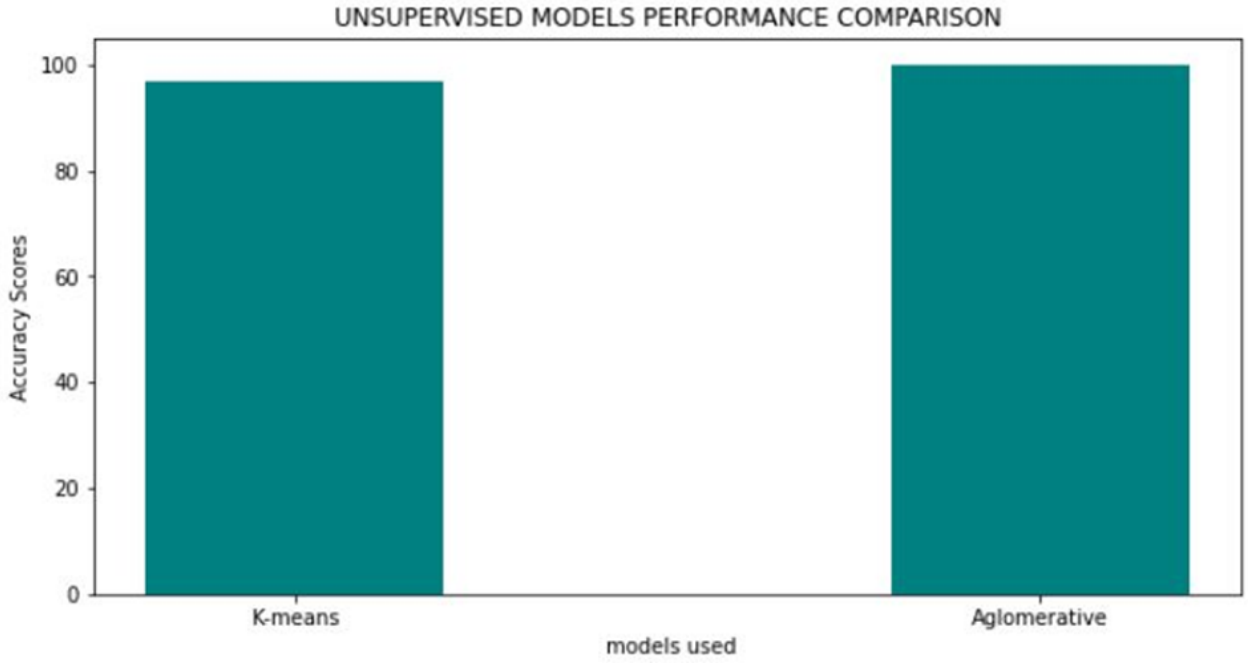

Fig. 24: Unsupervised Models Performance Comparison

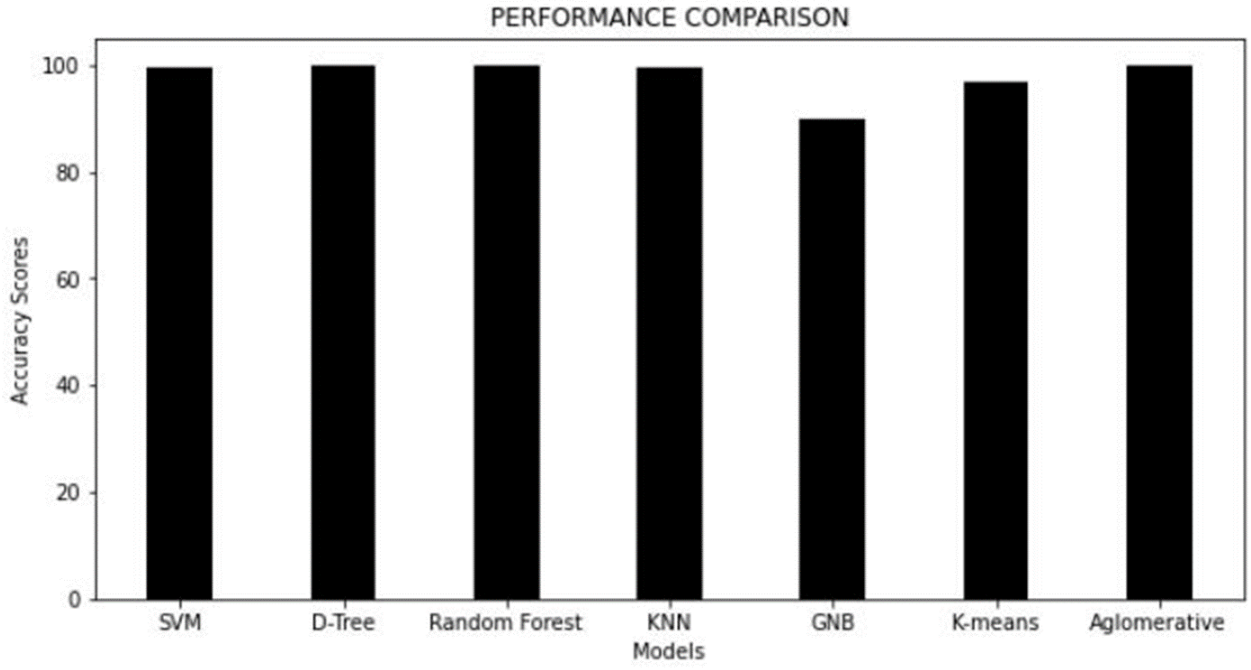

Fig. 23 All Models Performance Comparision

VIII. Conclusion:

In this research, we conducted a comprehensive analysis of five different machine learning models, namely K-Nearest Neighbors (KNN), Support Vector Classifier (SVC), Random Forest, Decision Tree, and Gaussian Naive Bayes (GNB), to assess their performance in sentiment analysis. We achieved an impressive average accuracy of 0.99 across these models, with Random Forest emerging as the top-performing model due to its superior accuracy compared to others.

Our approach involved splitting the dataset into training and testing sets (70-30 split) to ensure robust model evaluation. We utilized the Random Forest model from the scikit-learn package to predict the output variable based on multiple input variables. This supervised learning process included fitting the model on the training data and subsequently using it to predict the output variable on the test data.

We further evaluated model performance by comparing the predicted values with the actual output, thereby calculating the accuracy. Additionally, we employed ensemble techniques like averaging, max voting, stacking, bagging, and boosting, demonstrating that combining multiple models can yield even better performance.

In conclusion, this research underscores the effectiveness of Random Forest in sentiment analysis while highlighting the potential of ensemble methods for further enhancing predictive accuracy. These findings are valuable for practical applications in sentiment analysis across various domains.

IX. Future Scope:

The future scope for revolutionizing sentiment analysis through AI, particularly using Random Forest, is promising and multifaceted. As AI technologies continue to advance, the following areas hold substantial potential:

Enhanced Accuracy: AI-driven sentiment analysis, including Random Forest, is poised to improve its accuracy further. Continuous model refinements and larger training datasets will reduce errors and enable more precise sentiment classification.

Multilingual and Cross-Cultural Analysis: Expanding AI sentiment analysis to encompass diverse languages and cultural contexts will be crucial. Developing models capable of understanding nuanced sentiments across global audiences is a priority.

Real-Time Analysis: Real-time sentiment monitoring for social media and news feeds will become increasingly important for businesses and governments. AI will play a pivotal role in providing timely insights and responses.

Industry-Specific Solutions: Tailored sentiment analysis models for specific industries, such as healthcare, finance, and entertainment, will continue to emerge. These models will address unique domain-specific challenges.

Ethical Considerations: Addressing ethical concerns, including bias mitigation, fairness, and transparency in AI sentiment analysis, will be paramount to ensuring responsible and unbiased results.

Human-AI Collaboration: Combining AI algorithms with human expertise for sentiment analysis will yield more accurate and context-aware results, particularly in complex decision-making scenarios.

Expanding Applications: Sentiment analysis will extend beyond customer reviews and social media to applications like political sentiment tracking, mental health analysis, and product development.

In summary, the future of sentiment analysis through AI, with Random Forest as a key player, holds immense potential for growth and innovation across various domains, demanding ongoing research and development efforts to unlock its full capabilities

References:

1. kaggle.com/datasets/jiashenliu/515k-hotel-reviews-data-in-europe
2. anaconda.org/conda-forge/vadersentiment
3. anaconda.org/anaconda/nltk