Report on Tourist Places Recommendations

Using Sentiment Analysis

**PG-DBDA September 2022**

**Submitted by:**

## Project Team 13

**Aakash Rohila**

**Amol Komti**

**Ankita Gupta**

**Apurva Bagde**

**Chetan Kanade**

|  |  |
| --- | --- |
| Introduction | 3 |
| Problem Statement | 4 |
| Literature Survey | 5 |
| Libraries Used | 6 |
| Alternate Model and their disadvantage | 7 |
| Flowchart | 10 |
| Web Scraping | 15 |
| Data Preprocessing in PySpark | 16 |
| Model Creation | 18 |
| App Creation | 21 |
| Conclusion and Future Scope | 23 |
| References | 24 |

# Table of Context

# 1. INTRODUCTION

Tourist Places Recommendations using Sentiment Analysis is a technique that leverages natural language processing (NLP) to help tourists identify the most suitable places based on their preferences and the sentiments expressed by previous visitors. Sentiment analysis is a subfield of NLP that involves the use of machine learning algorithms to identify and extract subjective information from text data, such as opinions, emotions, and attitudes.

When it comes to Tourist Places Recommendations using Sentiment Analysis, the issue of fake reviews is a significant concern. Fake reviews are those that are intentionally created to misrepresent the experience of the reviewer or manipulate the sentiment analysis model's output. These reviews can be created by individuals, competitors, or even bots and can significantly influence the recommendations given by the sentiment analysis model. To address this issue, there are several techniques that can be used to detect and filter out fake reviews. These techniques include machine learning algorithms that can identify patterns in the text and behavior of the reviewer, as well as manual review by human moderators who can analyze the content of reviews for authenticity.

The process involves analyzing reviews, feedback, and other user-generated content to extract sentiment and generate insights about a particular tourist destination. The sentiment analysis model can identify positive, negative, or neutral sentiments from the text, which can be used to recommend tourist destinations to potential visitors.

By leveraging sentiment analysis, tourist destination providers can better understand the experiences of their visitors, and use this information to improve their services, amenities, and overall experience. At the same time, tourists can benefit from personalized recommendations that match their preferences and expectations, leading to a more enjoyable and fulfilling travel experience.

Overall, Tourist Places Recommendations using Sentiment Analysis is a project that helps both tourists and tourism providers by providing valuable insights that can enhance the travel experience for all parties involved.

Models used in this project:

1. LOGISTIC REGRESSION
2. NAÏVE BAYES
3. RANDOM FOREST
4. VADER
5. ROBERTA

# PROBLEM STATEMENT

Tourists often face challenges in identifying the best tourist destinations that match their preferences and expectations. With the vast amount of information available online, it can be overwhelming to sift through reviews and feedback to determine the most suitable places to visit.

Moreover, comments and ratings on many websites consist of fake reviews generated by Spammers, which can manipulate the ratings and lead to inaccurate recommendations. This can result in tourists having a suboptimal experience, which can negatively impact the reputation of the tourist destination.

Therefore, the challenge is to develop a robust and accurate Tourist Places Recommendations system that utilizes sentiment analysis to generate recommendations based on the reviews. The system must also be able to detect and filter out fake reviews to provide reliable and trustworthy recommendations. Ultimately, the goal is to enhance the travel experience of tourists and improve the overall tourism industry's reputation.

# LITERATURE SURVEY

1. **Fake Reviews Detection: A Survey**

**Year of Publication:** April 1, 2021

**Inferences:**

Most existing research works showed that spammers do not write detailed reviews about a service or a product, which might help to detect spammers.

Text features include semantic, grammar, lexicon, and metadata feature about the review, which can help identify the fake reviews.

Traditional machine learning usually achieves good results with small datasets compared to the deep learning models.

POS feature achieves good results in cross domain, it is not effective in detecting fake reviews when

Compared to other features, such as BoW

1. **RoBERTa: A Robustly Optimized BERT Pretraining Approach**

**Year of Publication:** July 26, 2019

**Inferences:**

RoBERTa is a pretraining approach for natural language processing (NLP) tasks that builds on top of the popular BERT model.

Here are some of the main inference points of RoBERTa:Pretraining corpus: RoBERTa is trained on a large corpus of text, which includes a variety of sources, such as books, articles, and web pages. This helps to improve the model's ability to handle diverse and varied text.Performance: RoBERTa is trained on a much larger amount of training data but on larger amount of data it takes longer training time.Achieves state-of-the-art performance: RoBERTa has achieved state-of-the-art performance on a wide range of NLP tasks, including sentiment analysis.

These are some of the main inference points of RoBERTa. Overall, RoBERTa represents an important advancement in the field of NLP, and has helped to push the state-of-the-art in a number of different tasks.



# LIBRARIES USED

### Pandas

Pandas is a popular open-source Python library used for data manipulation, analysis, and

visualization. It provides data structures and functions for efficiently handling and processing large

datasets.

1. **Numpy**

NumPy is a popular open-source Python library used for numerical computing. It provides functions

for working with arrays and matrices of numbers, as well as mathematical functions for performing

operations on those arrays.

### NLTK

The Natural Language Toolkit (NLTK) is a popular open-source Python library used for natural

language processing (NLP). It provides a wide range of tools and resources for working with human

language data, including text processing, tokenization, parsing, and classification.

### 4.TQDM

TQDM is an open-source Python library that provides a progress bar for iterative tasks. TQDM

provides a progress bar that can be added to any iterative task, such as loops or iterators, to

show the progress of the task.

**5. Seaborn**

Seaborn is a popular open-source Python library used for data visualization. It is built on top of

Matplotlib and provides a high-level interface for creating informative and attractive statistical graphics.

**6.Matplotlib**

Matplotlib is a popular open-source Python library used for data visualization. It provides a wide range

of functions for creating static, interactive, and animated visualizations of data.

**7.Sklearn**

Scikit-learn (sklearn) is a popular open-source Python library used for machine learning. It provides a

wide range of functions for data preprocessing, model selection, model training, and model evaluation

**5.**  **ALTERNATE MODELS and THEIR DISADVANTAGES**

### Logistic Regression

Logistic Regression is a machine learning algorithm used for classification problems. It is a type of

supervised learning algorithm that is used to predict the probability of a categorical outcome based

on one or more predictor variables.

**Disadvantages:**

Limited capturing of nuances: Logistic regression can be limited in capturing nuances in the

sentiment expressed in text. It is designed to predict binary outcomes, which can lead to a loss of

information when predicting sentiment scores.

Limited ability to handle unstructured data: Logistic regression is designed for structured data, and

may not be the best choice for unstructured text data that contains many different types of

information.

### Random Forest (RF):

It is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression

problems. It builds decision trees on different samples and takes their majority vote for classification

and average in case of regression. One of the most important features of the Random Forest

Algorithm is that it can handle the data set containing continuous variables as in the case of

regression and categorical variables as in the case of classification.

**Disadvantages:**

Limited flexibility: Random Forest is a relatively inflexible algorithm and may not be able to capture

complex relationships between variables or features.

Risk of overfitting: Random Forest may be prone to overfitting, particularly when the number of

decision trees or features is large. This can lead to poor generalization performance on new da

### Naïve bayes

Naive Bayes is a probabilistic machine learning algorithm that is widely used for sentiment analysis

and other NLP tasks. The algorithm is based on Bayes' theorem, which states that the probability of

a hypothesis (such as a sentiment label) given some evidence (such as a set of features or words in

a text) is proportional to the product of the probability of the evidence given the hypothesis and the

prior probability of the hypothesis.

**Advantages:**

1. **Simplicity:** Naive Bayes is a simple algorithm that is relatively easy to implement and understand. It requires minimal computational resources, making it a good choice for real-time applications.
2. **Efficiency:** Naive Bayes is a fast algorithm that can handle large amounts of data. It can be trained on large datasets in a relatively short amount of time, which makes it a popular choice for sentiment analysis and other NLP tasks.
3. **Good performance:** Despite its simplicity, Naive Bayes can perform well on certain types of text, such as short reviews or tweets. It has been shown to achieve high accuracy in sentiment analysis tasks, especially when the text is well-formatted and contains a limited number of features.
4. **Robustness to noise:** Naive Bayes is robust to noisy data, which means that it can handle text with misspellings, grammatical errors, and other types of noise.
5. **Low risk of overfitting:** Naive Bayes is a low variance algorithm, which means that it is less likely to overfit the training data than more complex algorithms. This can be an advantage in cases where the training data is limited.

### Vader

VADER (Valence Aware Dictionary and Sentiment Reasoner) is a rule-based sentiment analysis

model that is designed to work specifically with social media text. The model is based on a lexicon (a

list of words and their associated sentiment scores) that has been manually annotated by human

raters to reflect the sentiment of words in the context of social media.

**Advantages:**

1. **Designed for social media:** VADER is specifically designed to work with social media text, which can contain a lot of slang, acronyms, and other informal language that may be difficult for other sentiment analysis models to interpret.
2. **Accurate for short texts:** VADER has been shown to perform well on short texts such as tweets, which are common in social media. This is because the model is able to use features such as capitalization, punctuation, and word order to infer the sentiment of the text.
3. **Transparent and interpretable:** Because VADER is a rule-based model that uses a pre-defined lexicon of words with associated sentiment scores, it is possible to understand how the model arrived at its sentiment scores and to modify the lexicon to customize the model for specific domains or languages.
4. **Handles figurative language:** VADER is able to handle the nuances and complexities of social media text, including sarcasm, irony, and other types of figurative language that may be difficult for other sentiment analysis models to interpret.
5. **Fast and efficient:** VADER is a relatively fast and efficient model that can analyze large amounts of text in a short amount of time, making it well-suited for real-time applications.

### Roberta

RoBERTa (Robustly Optimized BERT approach) is a natural language processing (NLP) model

developed by Facebook AI and released in 2019. RoBERTa is based on the BERT (Bidirectional

Encoder Representations from Transformers) model, but it makes several improvements to the

architecture and training procedure to achieve state-of-the-art performance on a variety of NLP tasks,

including sentiment analysis.

**Disadvantages:**

1. **Computational resources:** RoBERTa is a large and complex model, with over 355 million parameters. This means that it requires a significant amount of computational resources to train and use, which can make it difficult or impractical to use in some settings.
2. **Limited transferability:** RoBERTa is trained on a large corpus of text, which may not be

representative of all languages, dialects, or domains. This can limit the model's ability to transfer its performance to new domains or languages.

1. **High inference time:** RoBERTa requires significant computational resources for inference, particularly when making predictions for large amounts of text.

#### 

### FlowChart

### Project Architecture

A flowchart is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task. The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows.

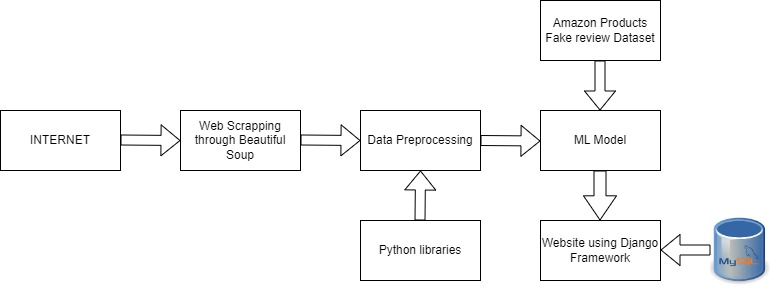


Fig: Basic Architecture

* 1. **Overall Architecture**

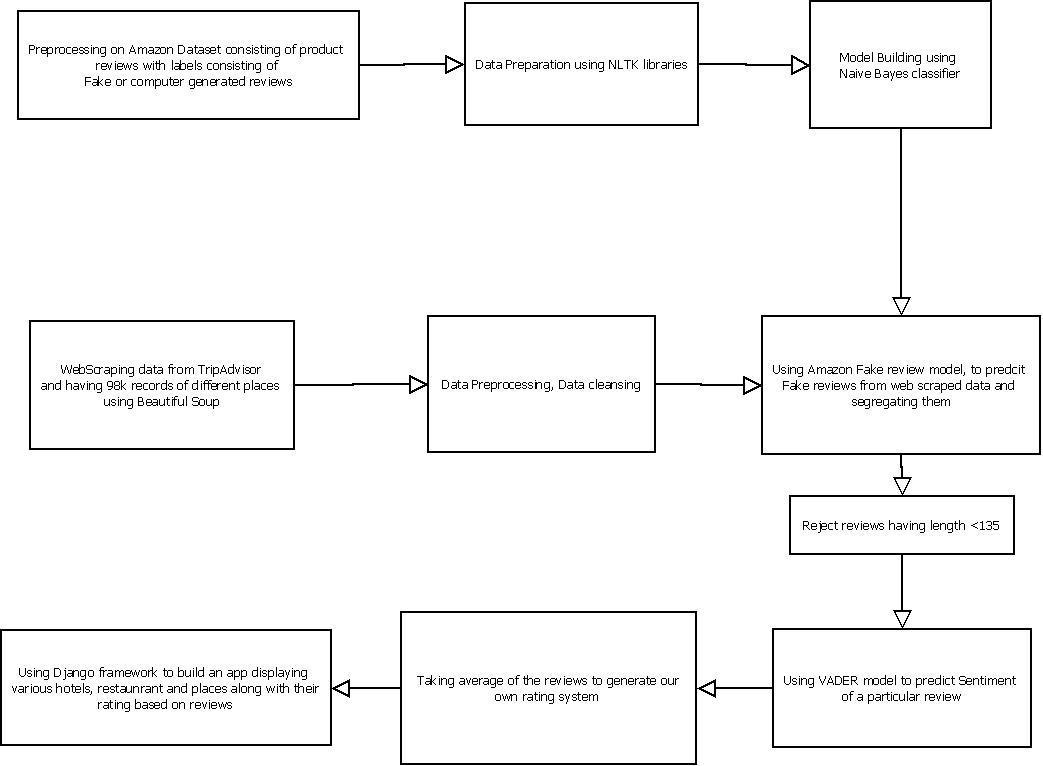
****

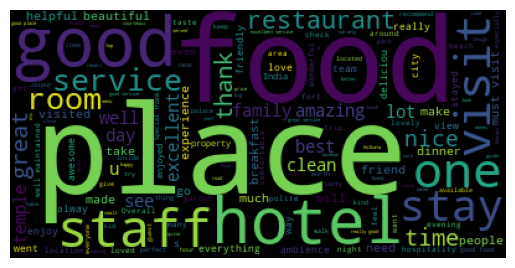
Fig: Overall Architecture

### 

# 6.3 VISUALIZATIONS

**Analysis using Matplotlib :**

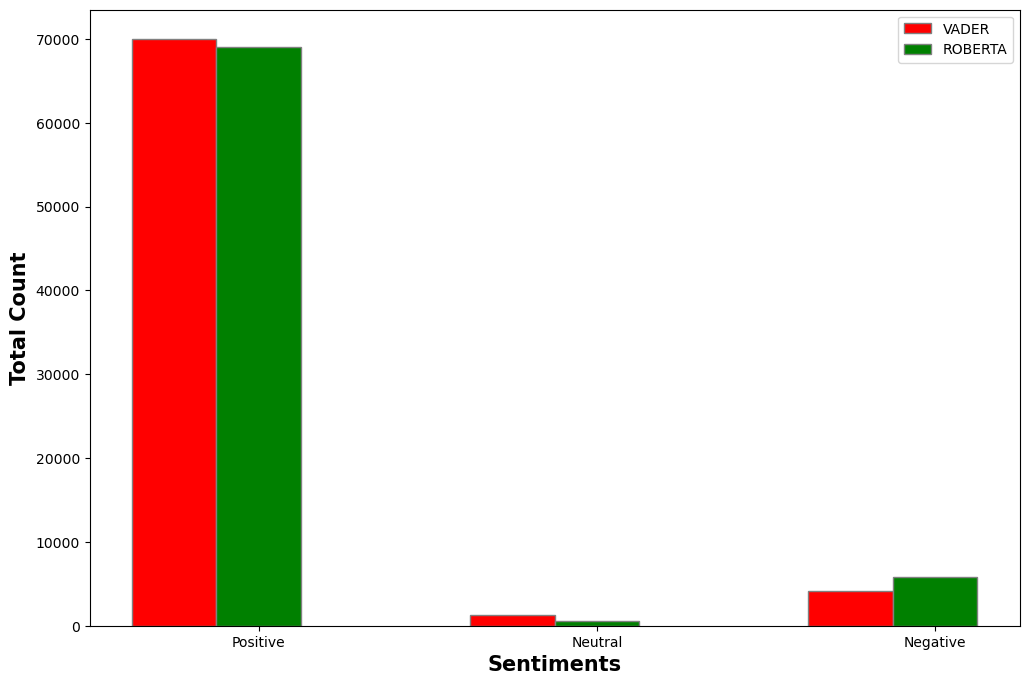
### Word Cloud

A word cloud, also known as a tag cloud or a weighted list, is a visual representation of a text dataset in which the size of each word corresponds to its frequency or importance in the dataset

They allow users to quickly identify the most prominent themes or ideas in a text dataset, and can provide a quick summary of the most significant words or concepts in the data.

From the above picture we can observe that ‘place’, ‘good’, ‘food’, ‘hotel’ etc., are some of the name in our reviews which is occurring multiple times

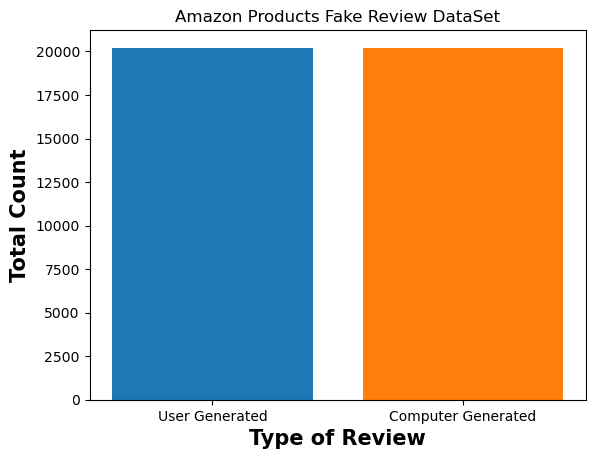
1. **Total count of reviews**



This graph depicts the count of reviews their sentiment according to VADER and ROBERTA model, from this we can observe that VADER and ROBERTA both are equally good at detecting sentiment with great accuracy.

ROBERTA did a better job at identifying sentiments compared to VADER model.

1. **Amazon products fake review data set**

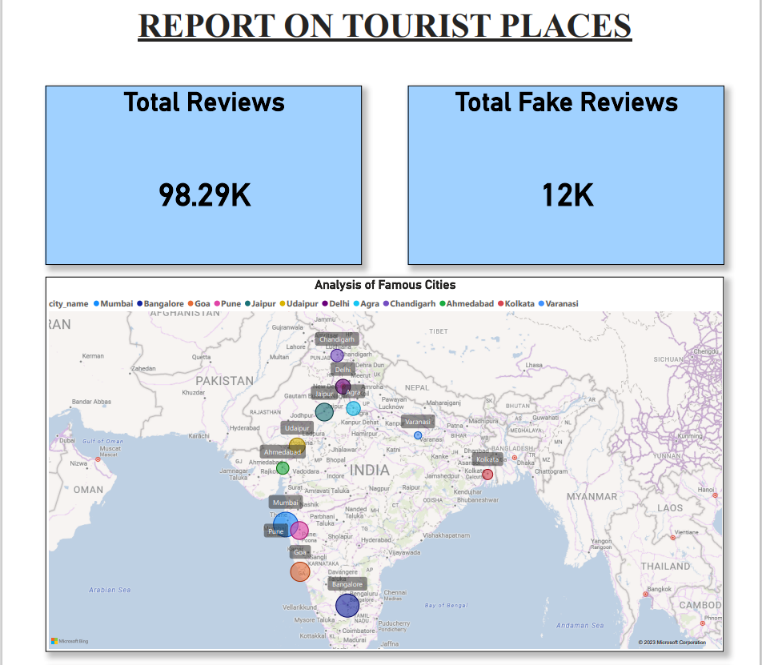


This is the Amazon review dataset which was used to train a model to detect fake reviews.

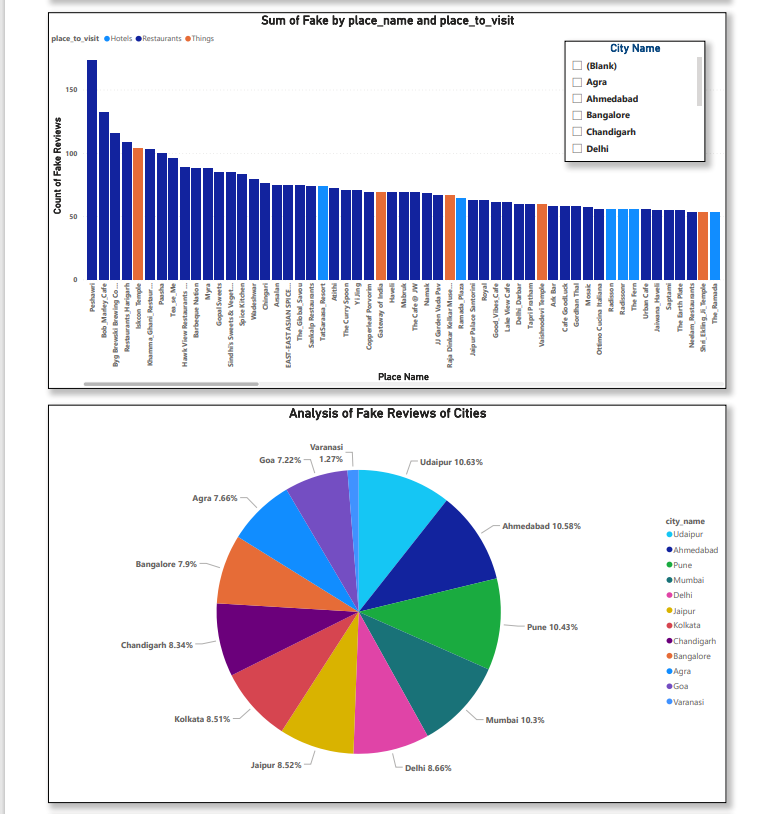
In this we can observe that the count of total User Generated and Computer-Generated reviews dataset are equal, which can be used to train a model as the dataset is balanced dataset.

**Analysis using PowerBI :**

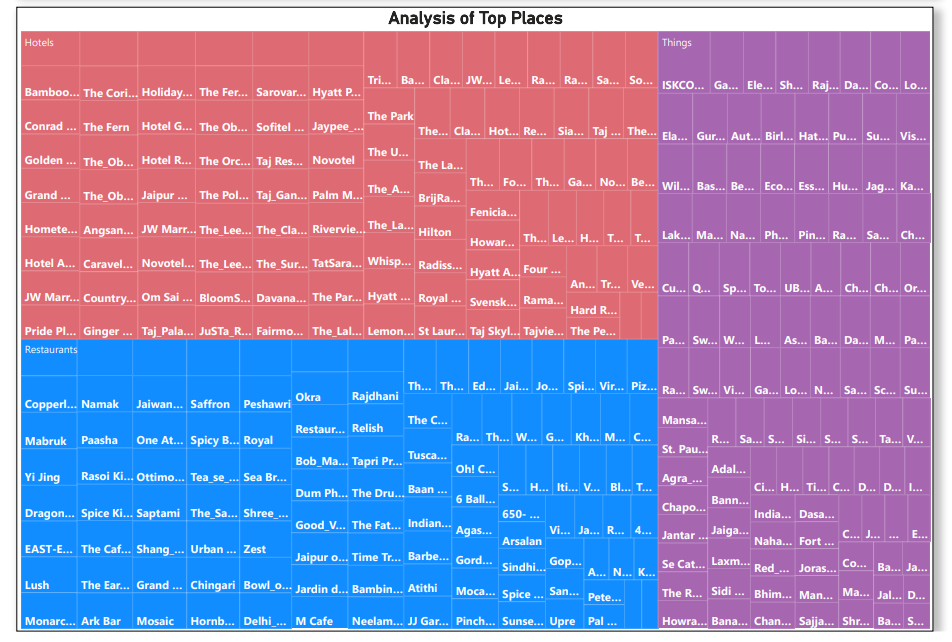
|  |
| --- |
| 1. The cards display the total reviews and fake reviews identified by our model out of the total reviews present in the dataset.  2. In the map we can see the city which has the highest ratings according the size of the bubbles which is Bangalore and Mumbai according to the analysis. |



|  |
| --- |
| 1.The bar graph shows count of total reviews by place name and place to visit like a hotel, restaurant, or site to visit. From the graph we can conclude that the restaurants category has the most number of fake reviews.  2.In the pie chart we can see the percentage of fake reviews in each city from the data in which Ahmedabad has the most amount of fake reviews amongst the cities and Varanasi has least amount of fake reviews. |



|  |
| --- |
| In the tree graph we have analyzed top hotels restaurants and sites amongst all the cities by the ratings generated by the model. Here as many of the places have equal rating the area of the places is same in the graph. |



# WEB-SCRAPPING

**Web-Scrapping Data from Trip Advisor**

from bs4 import BeautifulSoup

import requests

import numpy as np

import pandas as pd

counter = 1

def writer(city\_name, place\_to\_visit, place\_name, reviews):

    ''' This method is created to create output excel file with ease '''

    global counter

    hotel\_dict = {'city\_name':city\_name, 'place\_to\_visit': place\_to\_visit, 'place\_name': place\_name, 'reviews': reviews}

    df=pd.DataFrame(hotel\_dict)

    folder\_path = f'C:\Modules\Major Project\Web\_Scrapping\\{city\_name}\\{place\_to\_visit}\\{counter}.xlsx'

    df.to\_excel(folder\_path, index=False, header=True)

    print('This files has',len(reviews),'rows')

    print(f'File : {counter}.xlsx written to path')

    counter = counter+1

**>The Lalit Hotel**

# Defining user

user\_agent = {'User-agent': 'Mozilla/5.0'}

url\_first = 'https://www.tripadvisor.in/Hotel\_Review-g304554-d299124-Reviews'

url\_last = 'The\_Lalit\_Mumbai-Mumbai\_Maharashtra.html'

reviews\_lalit = []

#This loop was necessary to go to next page on trip advisor

for x in range(10,310,10):

    #This url\_mid will change va

    url\_mid = '-or' + str(x) + '-'

    url = url\_first + url\_mid + url\_last

    html\_text = requests.get( url , headers = user\_agent).text

    soup = BeautifulSoup(html\_text , 'html.parser')

    review\_div = soup.find\_all('div' , class\_ = 'fIrGe \_T')

    for r in review\_div:

        review\_text = r.find('q' , class\_ = 'QewHA H4 \_a')

        if review\_text != None:

            #print(review\_text.span.text)

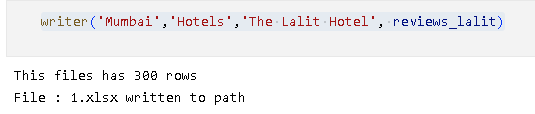
            #print()

            reviews\_lalit.append(review\_text.span.text)

#print(reviews\_lalit)

Above code is just an example of the code for web scraping reviews from The Lalit Hotel, similarly for different places there are different code.

writer('Mumbai','Hotels','The Lalit Hotel', reviews\_lalit)



1. **DATA PREPROCESSING IN PySpark**

# Importing and setting up Spark context

from pyspark.sql import SparkSession

spark = SparkSession.builder\

        .master("local")\

        .appName("Kaggle")\

        .config('spark.ui.port', '4050')\

        .getOrCreate()

#Importing necessary Libraries

from pyspark.ml.feature import IDF, Tokenizer , StopWordsRemover

import nltk

nltk.download('wordnet')

nltk.download('omw-1.4')

from nltk.stem import WordNetLemmatizer

from pyspark.sql.functions import udf

from pyspark.sql.functions import lower

from pyspark.sql.types import ArrayType, StringType

from pyspark.sql.functions import regexp\_replace

from pyspark.sql.functions import concat\_ws

# Removing all the character other than Upper case and Lower Case Alphabets

df = df.withColumn("reviews", regexp\_replace("reviews", "[^a-zA-Z]", " "))

#Converting uppercase letter to lower case

df = df.withColumn("lower\_reviews", lower(df["reviews"]))

# Tokenizing each sentence into words of list.

tokenizer = Tokenizer(inputCol="lower\_reviews", outputCol="tokenize\_reviews")

df = tokenizer.transform(df)

# Removing stop words from all the list of words.

stopwords\_remover = StopWordsRemover(inputCol="tokenize\_reviews", outputCol="filtered\_words")

df = stopwords\_remover.transform(df)

#Lemmatizing each word

wordnet = WordNetLemmatizer()

lemmatize\_udf = udf(lambda tokens: [wordnet.lemmatize(token) for token in tokens], ArrayType(StringType()))

df = df.withColumn("lemmatize\_reviews", lemmatize\_udf(df["filtered\_words"]))

# Joining each word list into a complete sentence

df = df.withColumn("reviews\_cleaned", concat\_ws(" ", "reviews"))

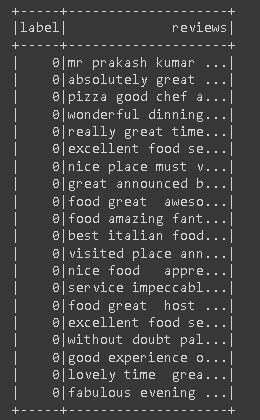
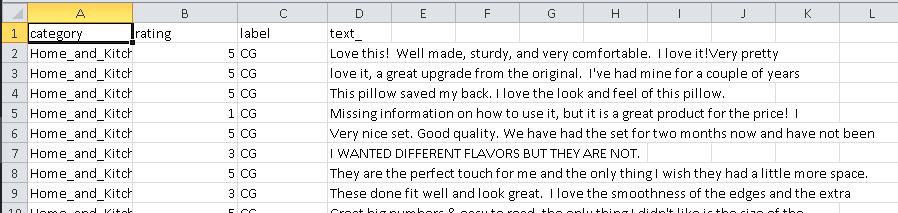


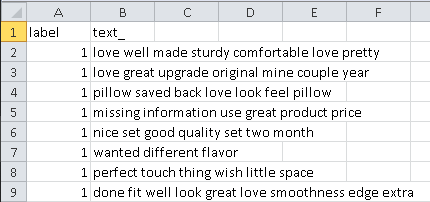
Fig: **Preprocessed Data for modelling**

**Amazon Original Dataset**

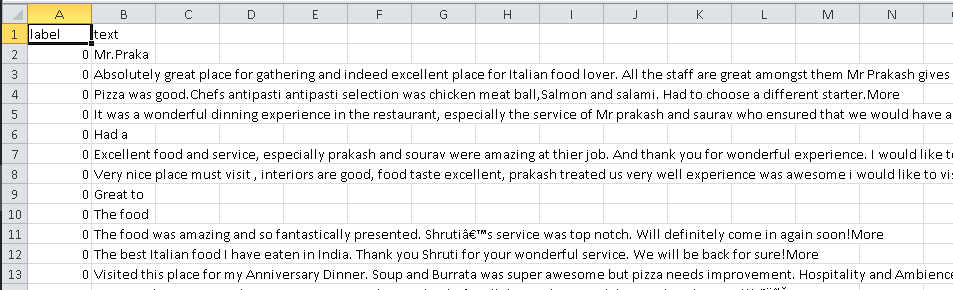


Using Python, we have cleaned the above dataset into the below processed csv file, where CG represents Computer Generated reviews and UG represents User Generated reviews.

**Amazon Preprocessed Dataset**



**Web Scraped Preprocessed Dataset**



**Combining Amazon Preprocessed Dataset + Web Scraped Preprocessed Dataset into one dataset to train our model.**

1. **MODEL CREATION**

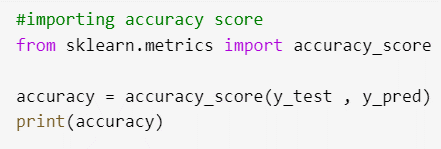
**1.Naive Bayes Classifier**

# Training model using Naive Bayes Classifier

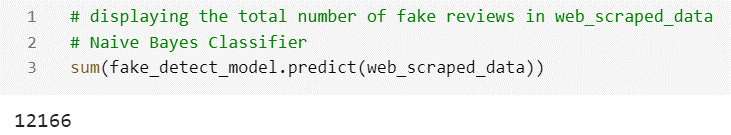
from sklearn.naive\_bayes import MultinomialNB

fake\_detect\_model = MultinomialNB().fit(X\_train , y\_train)

y\_pred = fake\_detect\_model.predict(X\_test)







**2.Logistic Regression**

|  |
| --- |
| # Training a model based on Logistic Regression  from sklearn.linear\_model import LogisticRegression  classifier = LogisticRegression(random\_state = 0) |

**3.Random Forest Classifier**

|  |
| --- |
|  |

**Sentiment Analysis Model Creation:**

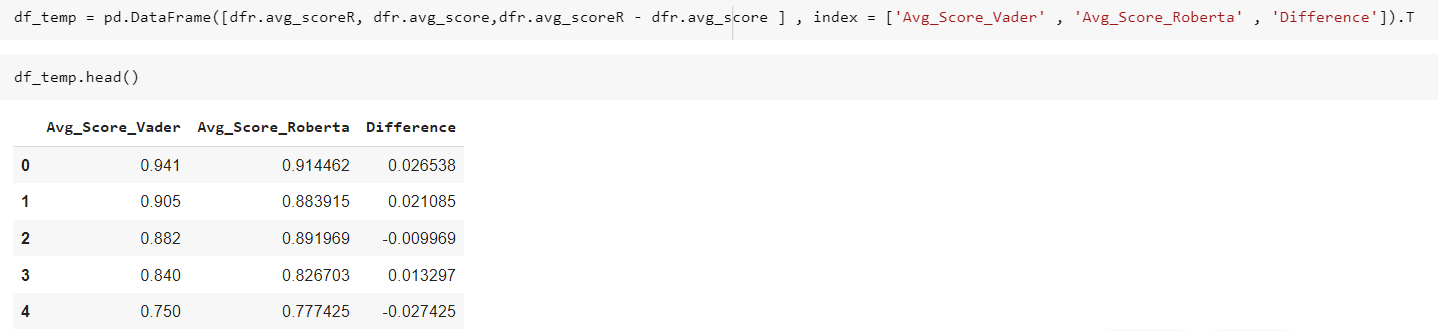
**1. Vader Model**

|  |
| --- |
|  |

**2. RoBERTa Model**

|  |
| --- |
|  |

**Difference Between Vader and Roberta Model Score**



**Conclusion obtained From the Difference of Vader and Roberta Model Average Score:**

As we can observe that there was not much of difference in the average value between VADER and ROBERTA model, ROBERTA model works better for individual sentences and returns sentiment based on that sentence but collective score i.e. While taking average not much of significance was observed so for saving computational power and in certain scenario VADER model is better than ROBERTA mode.

**10. APP CREATION (Django Framework)**

**What is Django?**

Django is a high-level web framework written in Python that follows the Model-View-Controller (MVC) architectural pattern. It was created to help developers build web applications quickly and efficiently, with a focus on reusability, modularity, and rapid development. Django provides many built-in features such as an Object-Relational Mapping (ORM), URL routing, a template engine, and a built-in admin interface, among others. It also has a strong emphasis on security, scalability, and community support.

**Why should data scientists use Django?**

There are several reasons why one might choose to use Django for web development:

1. **Rapid development:** Django provides a lot of built-in functionality that helps developers create web applications quickly and efficiently. This can be especially helpful for teams that need to deliver projects on tight deadlines.
2. **Scalability:** Django is designed to be scalable, making it a good choice for applications that are expected to grow over time. It can handle high levels of traffic and can be easily scaled horizontally or vertically.
3. **Security:** Django has built-in security features that help protect web applications from common security threats, such as Cross-Site Scripting (XSS) and Cross-Site Request Forgery (CSRF). This can save developers a lot of time and effort when it comes to securing their applications.
4. **Community support:** Django has a large and active community of developers who contribute to the project and provide support to other developers. This means that there are many resources available for learning and problem-solving, and it can be easy to find help when needed.
5. **Versatility:** Django can be used to build a wide range of web applications, from simple websites to complex web applications. It is also highly customizable, which means that developers can tailor it to meet the specific needs of their projects.

**Type this command to install Django:**

* pip install django
* pip install django-binary-database-file



**Steps for building a web application using Django:**

1. **Install Django:**

To get started with Django, you need to install it on your system. You can do this using pip **pip install django**.

1. **Create a Django project:**

Once you have installed Django, you can create a new Django project using the **django-admin startproject** command. This will create a new directory with the name of your project and a few files that Django needs to run.

1. **Create a Django app:**

A Django project can contain one or more Django apps. We can create a new Django app using the **python manage.py startapp** command.

1. **Define models**:

In Django, models represent the data that your web application will work with. You can define models using Python classes that inherit from the **django.db.models.Model** class.

1. **Selecting Database**

As we know by default django framework have dbsqlite3 database support but django provide us support for many databases support as we have used **MySQL** in our **Tourist Place Recommendation Using Sentiment Analysis.**

1. **Create database tables:**

Once you have defined your models, you can create the corresponding database tables using the **python manage.py makemigrations** and **python manage.py migrate** commands.

1. **Create views**:

In Django, views handle requests and generate responses. You can define views using Python functions that take a request object as input and return a response object.

1. **Define URL patterns:**

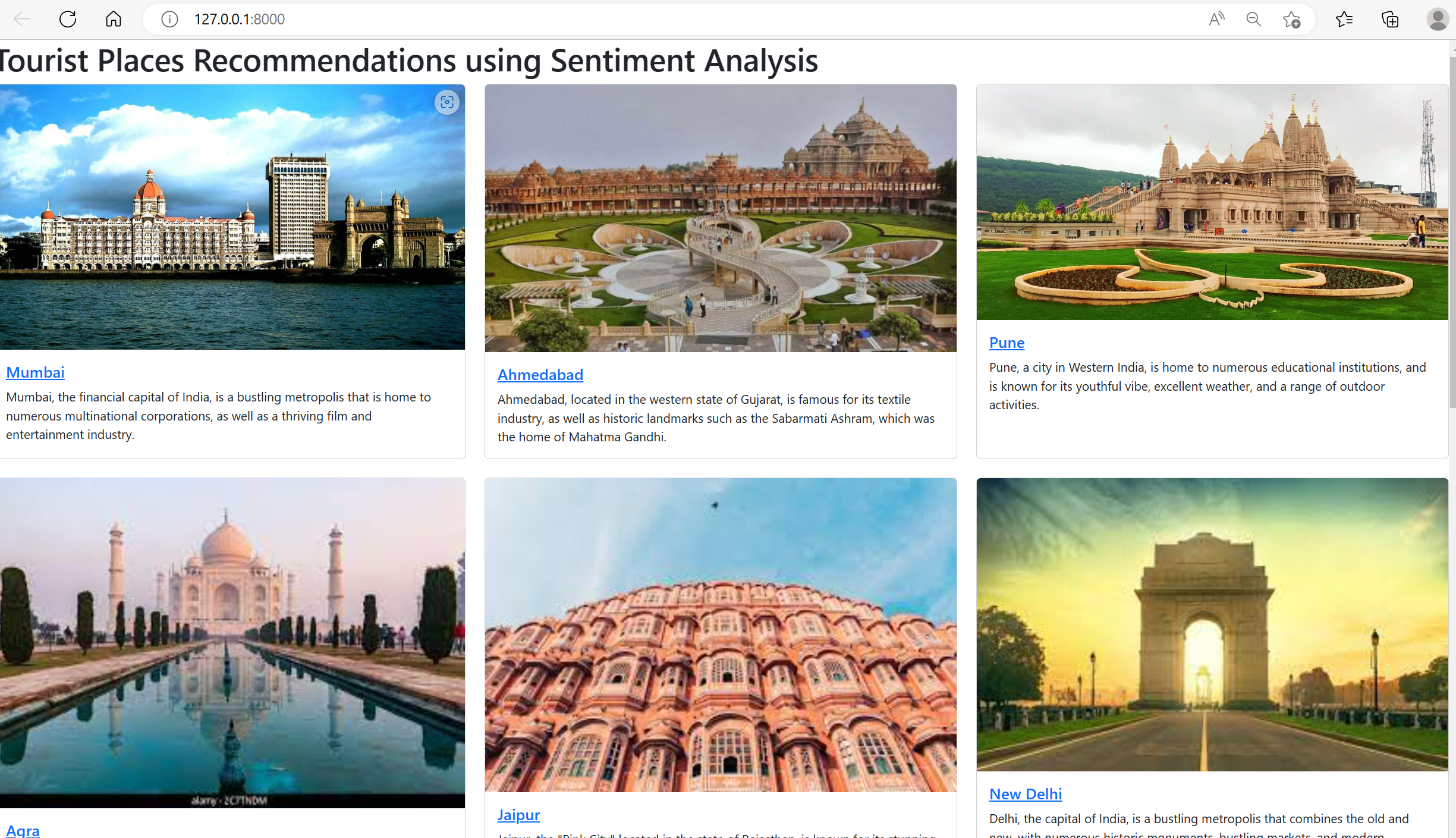
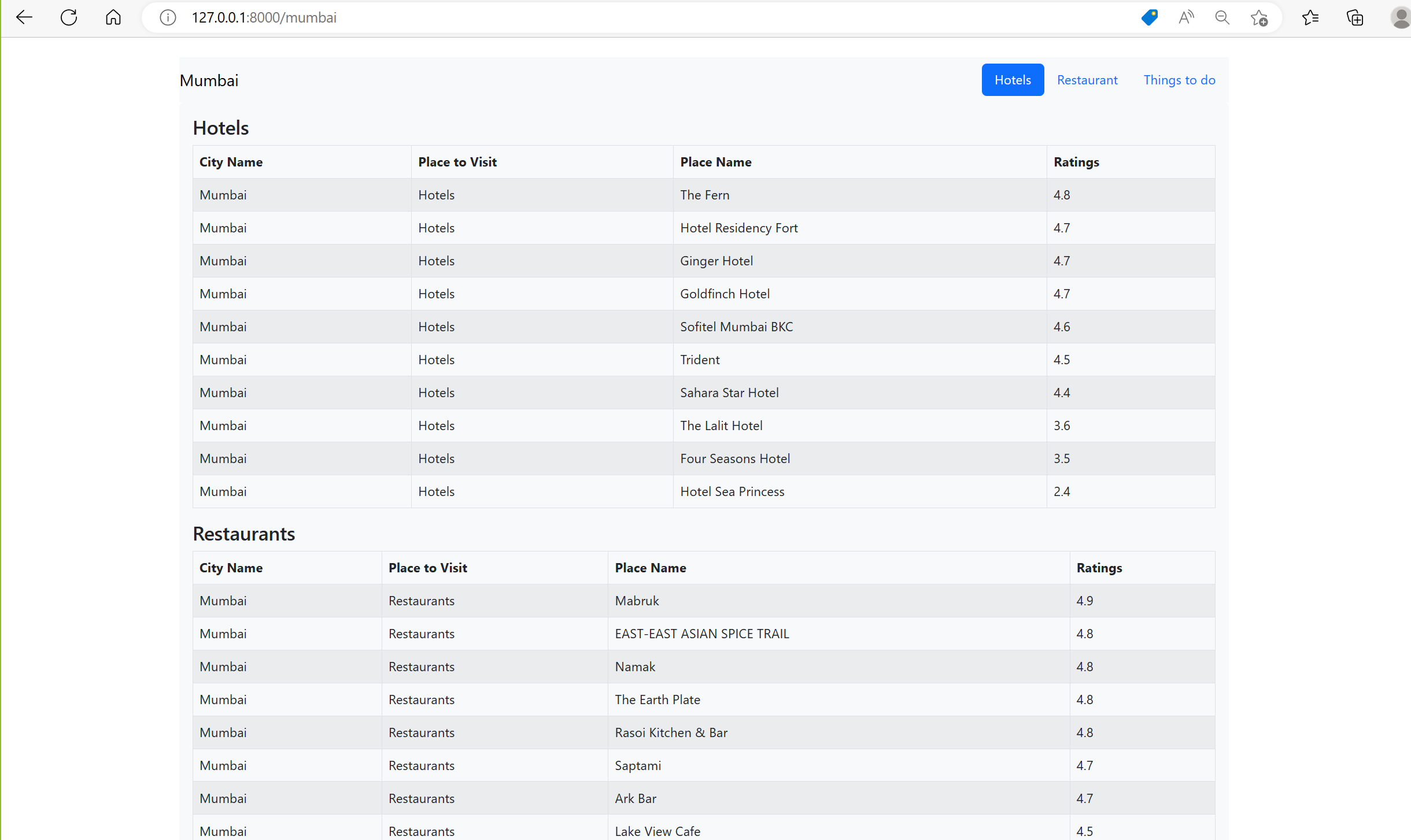
In Django, URL patterns map URLs to views. You can define URL patterns using regular expressions in a file called **urls.py**.

1. **Create templates:**

In Django, templates define the HTML that your web application will display. You can create templates using HTML and a templating language that Django provides.

1. **Serve the application**:

Finally, you can serve your Django application using the **python manage.py runserver** command. This will start a local development server that you can use to test your application.

# 

**11.CONCLUSION & FUTURE SCOPE**

**Conclusion:**

In conclusion, Tourist Places Recommendations using Sentiment Analysis is a promising application

of machine learning and natural language processing that can help people make better travel

decisions based on the sentiments expressed in online reviews. By analyzing large amounts of text

data from various sources, including social media and travel review websites, sentiment analysis

algorithms can extract insights and patterns about people's opinions and preferences regarding

different tourist destinations. This information can then be used to generate personalized

recommendations for users based on their interests and preferences. Although there are some

limitations to using sentiment analysis, such as the potential for inaccuracies and bias in the data, it

has shown to be a useful tool for making data-driven recommendations in a variety of industries. By

combining sentiment analysis with other machine learning algorithms and technologies such as

Django, numpy, pandas, and scikit-learn, developers can create powerful and efficient applications

that provide valuable insights to users.

**Future Scope:**

The future scope for Tourist Places Recommendations using Sentiment Analysis is vast and exciting,

with many potential opportunities for growth and development. Here are some possible areas where

this technology can be further expanded:

1. Integration with social media platforms: Currently, most sentiment analysis algorithms focus on reviews from travel websites. In the future, it may be possible to integrate social media data into these algorithms, allowing them to analyze sentiment from platforms like Twitter and Instagram.
2. Improved accuracy: One of the challenges of sentiment analysis is achieving high accuracy. As the technology continues to develop, there will be opportunities to refine the algorithms and improve their ability to accurately capture and analyze sentiment.
3. Personalization: Personalization is a key feature of tourist recommendations. By combining sentiment analysis with other machine learning algorithms, it may be possible to create even more personalized recommendations based on users' individual preferences and past travel behavior.
4. Multi-language support: Currently, most sentiment analysis algorithms only support a few languages. In the future, it may be possible to expand support to more languages, making it easier to analyze sentiment in a global context.
5. Real-time recommendations: As the technology becomes faster and more efficient, it may be possible to provide real-time recommendations to travelers as they are planning their trips, allowing them to adjust their plans based on the latest sentiment analysis data.

**12. REFERENCES**

## Dataset:

[**https://www.tripadvisor.in/**](https://www.tripadvisor.in/)

## Models:

### LOGISTIC REGRESSION

### <https://www.javatpoint.com/logistic-regression-in-machine-learning>

### NAÏVE BAYES

[**https://www.analyticsvidhya.com/blog/2021/07/performing-sentiment-analysis-with-naive- bayes-classifier/**](https://www.analyticsvidhya.com/blog/2021/07/performing-sentiment-analysis-with-naive-%20%20bayes-classifier/)

### RANDOM FOREST

### <https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/>

### VADER

### <https://www.analyticsvidhya.com/blog/2021/06/vader-for-sentiment-analysis/>

### ROBERTA

[**https://huggingface.co/docs/transformers/model\_doc/roberta**](https://huggingface.co/docs/transformers/model_doc/roberta)