



CENTER FOR DEVELOPMENT OF
ADVANCED COMPUTING



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

Table of Context

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

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

2. 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.

3. 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

2. 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.

4. LIBRARIES USED

1. 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.

2. 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.

3. 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

5.1 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.

5.2 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 data.

5.3 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.

5.4 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.

5.5 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.
3. **High inference time:** RoBERTa requires significant computational resources for inference, particularly when making predictions for large amounts of text.

6. FlowChart

6.1 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.

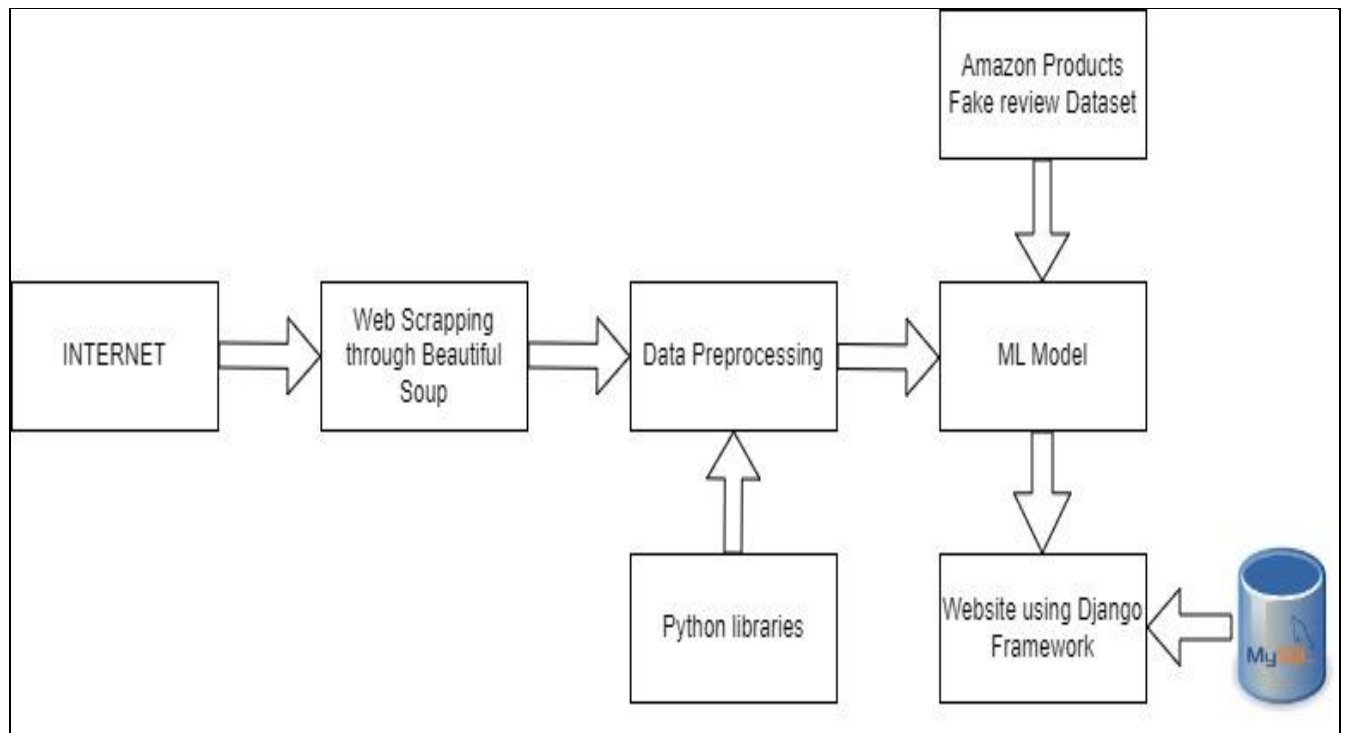


Fig: Basic Architecture

6.2 Overall Architecture

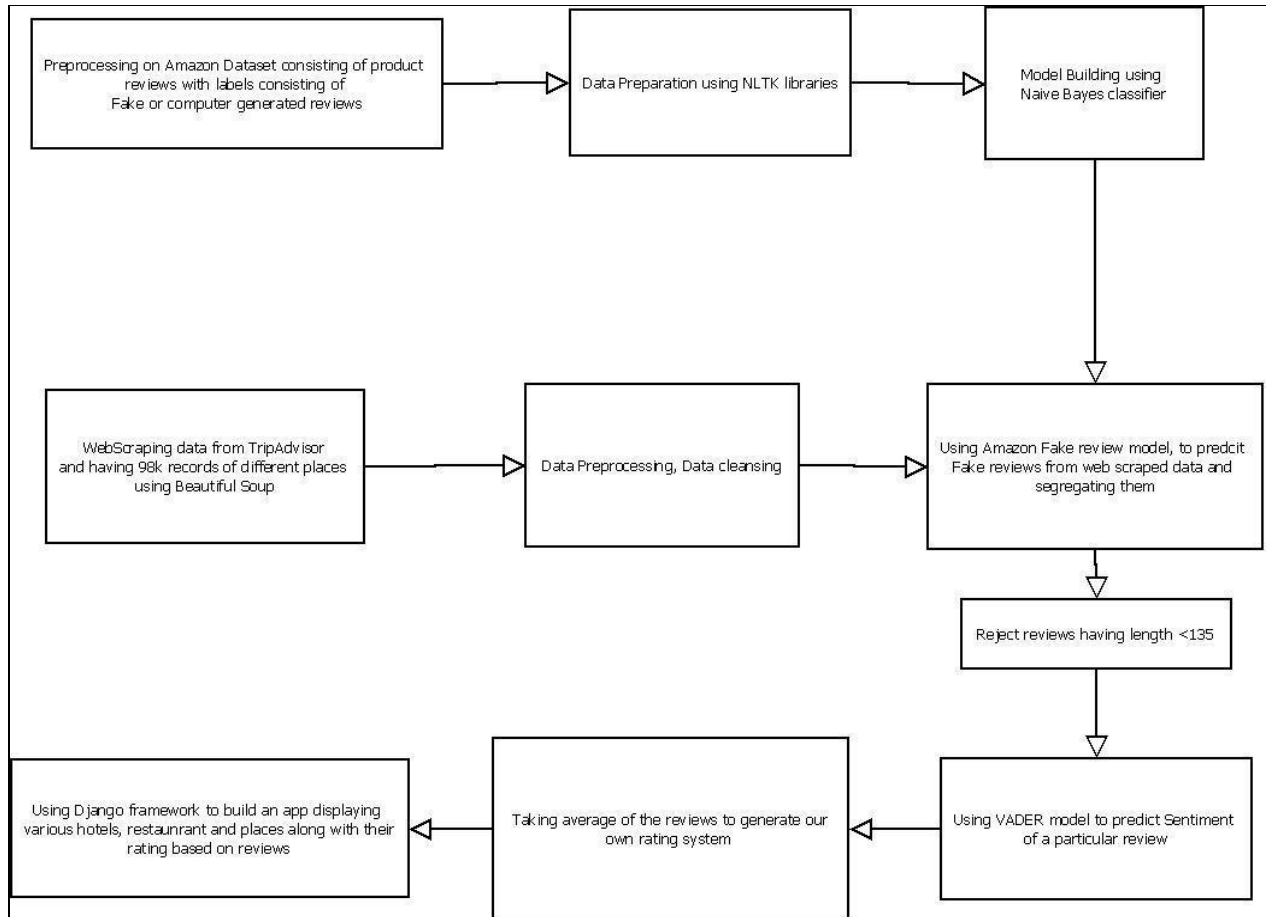
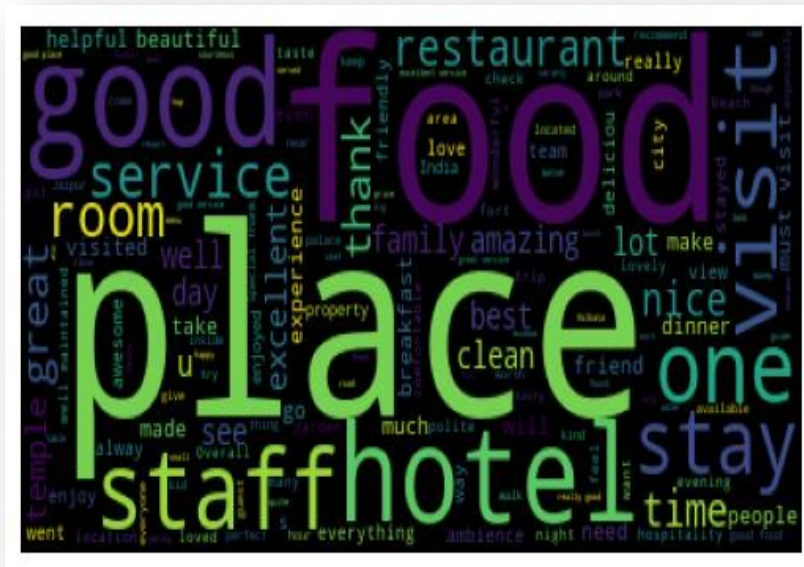


Fig: Overall Architecture

6.3 VISUALIZATIONS

Analysis using Matplotlib :

1. Word Cloud

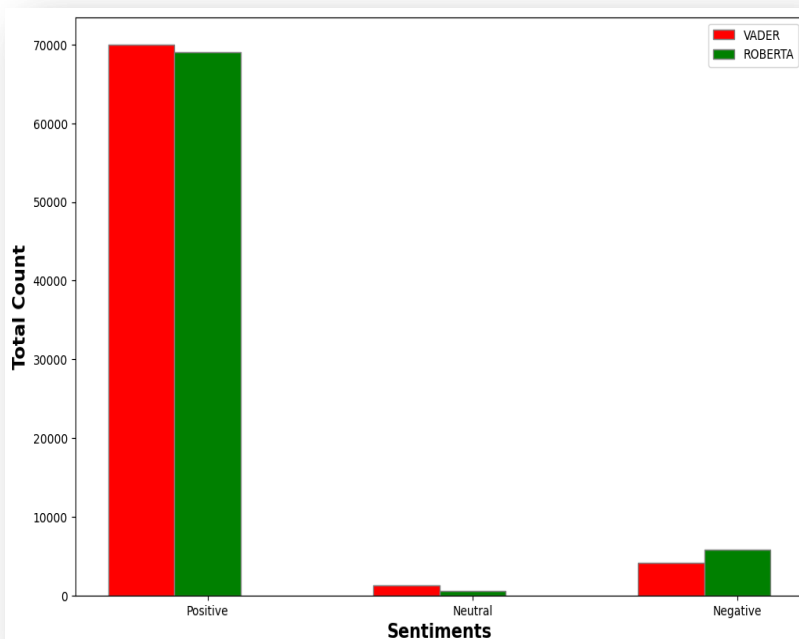


A word cloud, also known as a tag cloud or a weighted list, is a visual representation of a text dataset in which the size of each word corresponds to its frequency or importance in the dataset

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

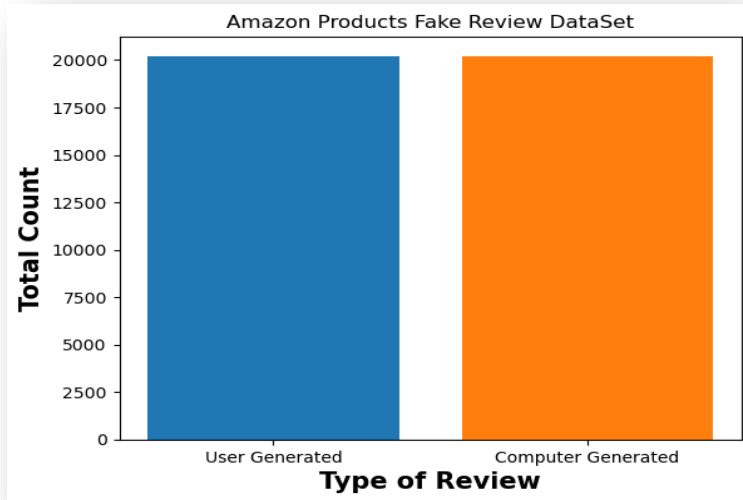
2. Total count of reviews



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.

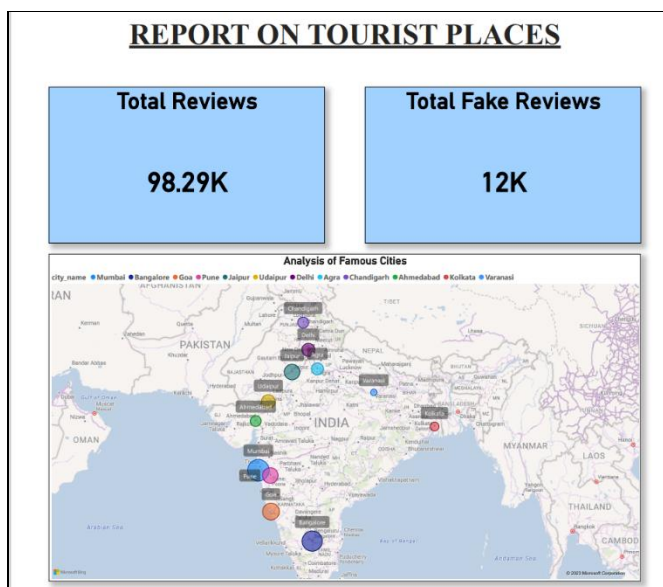
3. Amazon products fake review data set



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

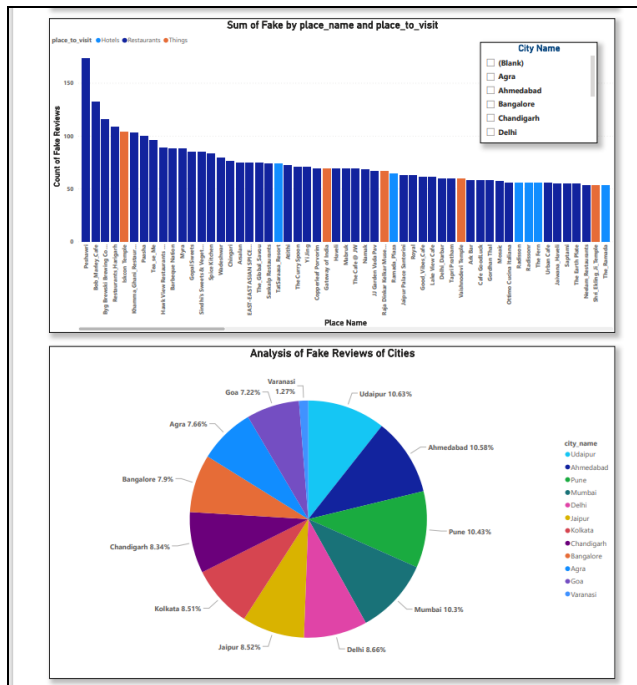
In this we can observe that the count of total User Generated and Computer-Generated reviews dataset are equal, which can be used to train a model as the dataset is balanced dataset.

Analysis using PowerBI :



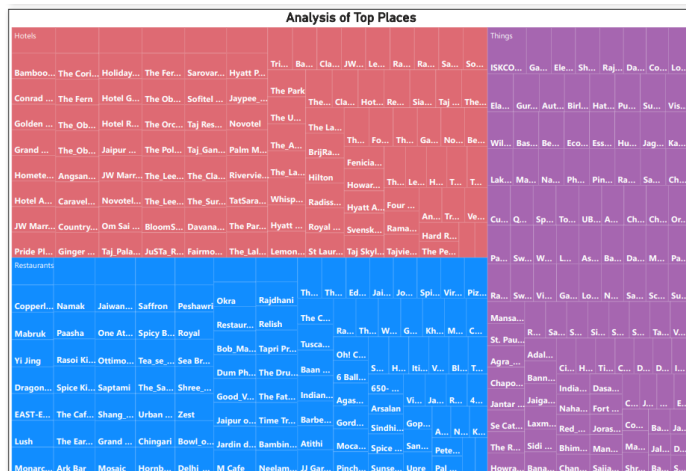
1. The cards display the total reviews and fake reviews identified by our model out of the total reviews present in the dataset.

2. In the map we can see the city which has the highest ratings according to 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.

7. 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_ = 'flrGe_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)
```

```
This files has 300 rows
File : 1.xlsx written to path
```

8. 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"))
```

```
+-----+-----+
|label|      reviews|
+-----+-----+
|0|mr prakash kumar ...|
|0|absolutely great ...|
|0|pizza good chef a...|
|0|wonderful dinning...|
|0|really great time...|
|0|excellent food se...|
|0|nice place must v...|
|0|great announced b...|
|0|food great aweso...|
|0|food amazing fant...|
```

Fig: Preprocessed Data for modelling

Amazon Original Dataset

	A	B	C	D	E	F	G	H	I	J	K	L
1	category	rating	label	text_								
2	Home_and_Kitch	5	CG	Love this! Well made, sturdy, and very comfortable. I love it!Very pretty								
3	Home_and_Kitch	5	CG	love it, a great upgrade from the original. I've had mine for a couple of years								
4	Home_and_Kitch	5	CG	This pillow saved my back. I love the look and feel of this pillow.								
5	Home_and_Kitch	1	CG	Missing information on how to use it, but it is a great product for the price! I								
6	Home_and_Kitch	5	CG	Very nice set. Good quality. We have had the set for two months now and have not been								
7	Home_and_Kitch	3	CG	I WANTED DIFFERENT FLAVORS BUT THEY ARE NOT.								
8	Home_and_Kitch	5	CG	They are the perfect touch for me and the only thing I wish they had a little more space.								
9	Home_and_Kitch	3	CG	These done fit well and look great. I love the smoothness of the edges and the extra								
10	Home_and_Kitch	5	CG	Great big numbers & easy to read, the only thing I didn't like is the size of the								

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

	A	B	C	D	E	F	
1	label	text_					
2	1	love well made sturdy comfortable love pretty					
3	1	love great upgrade original mine couple year					
4	1	pillow saved back love look feel pillow					
5	1	missing information use great product price					
6	1	nice set good quality set two month					
7	1	wanted different flavor					
8	1	perfect touch thing wish little space					
9	1	done fit well look great love smoothness edge extra					

Web Scraped Preprocessed Dataset

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	
1	label	text													
2	0	Mr.Praka													
3	0	Absolutely great place for gathering and indeed excellent place for Italian food lover. All the staff are great amongst them Mr Prakash gives													
4	0	Pizza was good.Chefs antipasti antipasti selection was chicken meat ball,Salmon and salami. Had to choose a different starter.More													
5	0	It was a wonderful dinning experience in the restaurant, especially the service of Mr prakash and saurav who ensured that we would have a													
6	0	Had a													
7	0	Excellent food and service, especially prakash and sourav were amazing at thier job. And thank you for wonderful experience. I would like to													
8	0	Very nice place must visit , interiors are good, food taste excellent, prakash treated us very well experience was awesome i would like to vi:													
9	0	Great to													
10	0	The food													
11	0	The food was amazing and so fantastically presented. Shruti's service was top notch. Will definitely come in again soon!More													
12	0	The best Italian food I have eaten in India. Thank you Shruti for your wonderful service. We will be back for sure!More													
13	0	Visited this place for my Anniversary Dinner. Soup and Burrata was super awesome but pizza needs improvement. Hospitality and Ambience													

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

9. 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)

# importing accuracy score
from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_test , y_pred)
print(accuracy)
```

```
0.8445653517991839
```

```
1 # displaying the total number of fake reviews in web_scraped_data
2 # Naive Bayes Classifier
3 sum(fake_detect_model.predict(web_scraped_data))
```

```
12166
```

2. Logistic Regression

```
# Training a model based on Logistic Regression
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)

# importing accuracy score
from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_test , y_pred)
print(accuracy)
```

```
0.8648448126622975
```

```
# displaying the total number of fake reviews in web_scraped_data
# Logistic Regression
sum(classifier.predict(web_scraped_data))
```

```
3307
```

3. Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier

# creating a RF classifier
clf = RandomForestClassifier(n_estimators = 100)

# importing accuracy score
from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_test , y_pred)
print(accuracy)
```

```
0.8439470755533572
```

```
# displaying the total number of fake reviews in web_scraped_data
# Random Forest Classifier
sum(clf.predict(web_scraped_data))
```

```
8649
```

Sentiment Analysis Model Creation:

1. Vader Model

```
pip install vaderSentiment
```

```
#import libraries
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

#To get the progress bar
from tqdm import tqdm, trange

def sentiment_scores(sentence):

    sia_obj = SentimentIntensityAnalyzer() #creating object

    sentiment_dict = sia_obj.polarity_scores(sentence) #

    return sentiment_dict

sentiment_data = [] #creating list

for i in trange(len(df)):

    sentiment_data.append(sentiment_scores(df['reviews'])(i))['compound'])
    # we are generating compound scores of each sentence

#displaying sorted df
sorted_df
```

	city_name	place_to_visit	place_name	avg_score
0	Agra	Hotels	Holiday_Inn	0.914
1	Agra	Hotels	Bansi_Home_Stay	0.892
2	Agra	Hotels	Jaypee_Palace_Hotel	0.884
3	Agra	Hotels	Hotel_Taj_Resorts	0.827
4	Agra	Hotels	The_Grand_Imperial	0.785
...
327	Varanasi	Restaurant	Brown_Bread_Bakery	0.456
328	Varanasi	Things	Assi_Ghat	0.695
329	Varanasi	Things	Dasaswamedh_Ghat	0.668
330	Varanasi	Things	Banaras_Ghats	0.665
331	Varanasi	Things	Sarnath	0.648

2. RoBERTa Model

```
from transformers import AutoTokenizer
from transformers import AutoModelForSequenceClassification
from scipy.special import softmax

MODEL = f"cardiffnlp/twitter-roberta-base-sentiment"
tokenizer = AutoTokenizer.from_pretrained(MODEL)
model = AutoModelForSequenceClassification.from_pretrained(MODEL)

# Run for RoBERTa Model
sentiment_data = []

for i in trange(len(df)):

    encoded_text = tokenizer(df['reviews'])(i), return_tensors='pt', max_length=512, truncation=True)
    output = model(**encoded_text)
    scores = output[0][0].detach().numpy()
    scores = softmax(scores)
    scores_dict = {
        'roberta_neg': scores[0],
        'roberta_neu': scores[1],
        'roberta_pos': scores[2]
    }

    compound = (scores_dict['roberta_neg'] + (scores_dict['roberta_neu'] * 2) + scores_dict['roberta_pos'] * 3) -2

    sentiment_data.append(compound)

#displaying sorted df
sorted_df
```

	city_name	place_to_visit	place_name	score_V	avg_score	avg_scoreR
0	Agra	Hotels	Holiday_Inn	0.8126	0.914462	0.941
1	Agra	Hotels	Jaypee_Palace_Hotel	0.9864	0.883915	0.905
2	Agra	Hotels	Bansi_Home_Stay	0.8860	0.891969	0.882
3	Agra	Hotels	Hotel_Taj_Resorts	0.9686	0.826703	0.840
4	Agra	Hotels	Howard_Plaza	0.9724	0.777425	0.750
...
327	Varanasi	Restaurant	Brown_Bread_Bakery	0.0828	0.455673	0.383
328	Varanasi	Things	Assi_Ghat	0.0000	0.694993	0.735
329	Varanasi	Things	Sarnath	0.8313	0.647928	0.710
330	Varanasi	Things	Banaras_Ghats	0.9778	0.665100	0.598
331	Varanasi	Things	Dasaswamedh_Ghat	0.0000	0.667587	0.569

Difference Between Vader and Roberta Model Score

```
df_temp = pd.DataFrame([dfr.avg_scoreR, dfr.avg_score,dfr.avg_scoreR - dfr.avg_score ], index = ['Avg_Score_Vader' , 'Avg_Score_Roberta' , 'Difference']).T
```

```
df_temp.head()
```

	Avg_Score_Vader	Avg_Score_Roberta	Difference
0	0.941	0.914462	0.026538
1	0.905	0.883915	0.021085
2	0.882	0.891969	-0.009969
3	0.840	0.826703	0.013297
4	0.750	0.777425	-0.027425

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**.

2. 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.

3. 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.

4. 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.

5. 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**.

6. 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.

7. 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.

8. Define URL patterns:

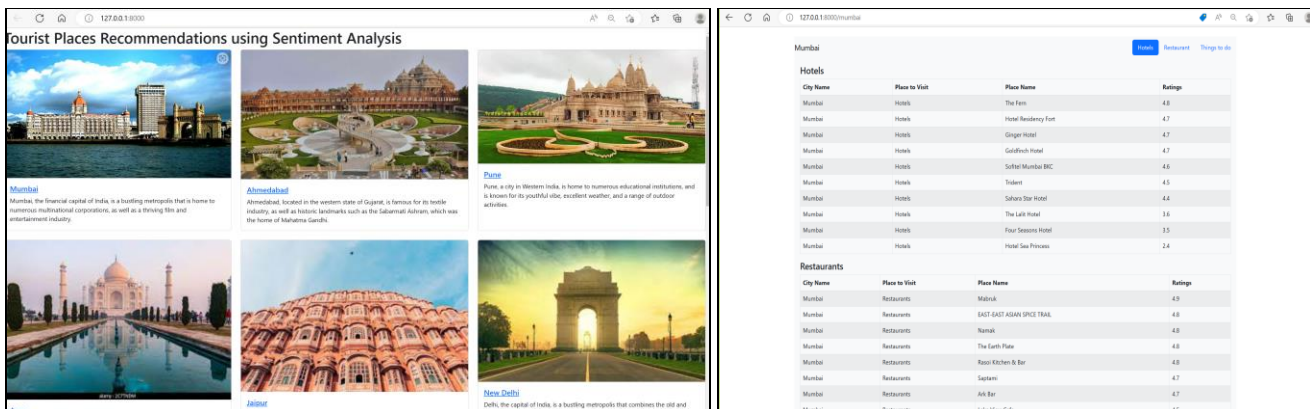
In Django, URL patterns map URLs to views. You can define URL patterns using regular expressions in a file called **urls.py**.

9. 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.

10. 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.



City Name	Place to Visit	Place Name	Ratings
Mumbai	Hotels	The Fern	4.9
Mumbai	Hotels	Hotel Rameshwar Fort	4.7
Mumbai	Hotels	Singh Hotel	4.7
Mumbai	Hotels	Goldfish Hotel	4.7
Mumbai	Hotels	Safari Mumbai B&C	4.6
Mumbai	Hotels	Talwar	4.5
Mumbai	Hotels	Sahara Star Hotel	4.4
Mumbai	Hotels	The Earth Hotel	3.6
Mumbai	Hotels	Four Seasons Hotel	3.5
Mumbai	Hotels	Hotel Sea Princess	2.4
Mumbai	Restaurants	Mahesh	4.9
Mumbai	Restaurants	SAIT EAST AGAIN SPICE TRAIL	4.9
Mumbai	Restaurants	Nomads	4.9
Mumbai	Restaurants	The Earth Place	4.9
Mumbai	Restaurants	Royal Kitchen & Bar	4.9
Mumbai	Restaurants	Saphere	4.7
Mumbai	Restaurants	Ark Bar	4.7
Mumbai	Restaurants	Lake View Cafe	4.5

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/>

- **Models:**

1. LOGISTIC REGRESSION

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

2. NAÏVE BAYES

<https://www.analyticsvidhya.com/blog/2021/07/performing-sentiment-analysis-with-naive-bayes-classifier/>

3. RANDOM FOREST

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

4. VADER

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

5. ROBERTA

https://huggingface.co/docs/transformers/model_doc/roberta