# Praxis Business School

CAPSTONE PROJECT REPORT

## Data-Driven Approach to Enhance the Indian Hotel Industry: Sentiment Analysis & Recommendations for Improved Customer Experience

4.5 Excellent ●●●●◐ 6,090 reviews

#9 of 98 hotels in Boston

| | |
|---|---|
| ●●●●◐ | Location |
| ●●●●● | Cleanliness |
| ●●●●● | Service |
| ●●●●◐ | Value |

Presented by: Supervisor:

-Akash Dey -Subhasis Dasgupta

-Arka Dhali

-Ipsheetha Nath

-Pallavi Mazumdar

-Soumya Roy

-Subhrajyoti Basak

-Tanushree Mandal

A report submitted in fulfilment of the requirements

for the certificate of PGPDS

in the

July, 2022 Batch

Data Science

27th April, 2023

## **ACKNOWLEDGEMENT**

We are highly grateful to Prof. Dr. Subhasis Dasgupta, Head of Machine Learning & Analytics, for his expert guidance and continuous encouragement throughout the whole project to reach at a significant standpoint. We would like to convey our thanks to Prof. Charanpreet Singh, Prof. Dr. Prithwis Mukherjee, Prof. Dr. Sourav Saha, Founder, Director and Academic Dean of this institute and all other professors of Data Science Department of this institute who taught us each and every course of this program.

## Contents

# **ABSTRACT**

One of the crucial components for any hotel organisation to comprehend the performance of a hotel is review research. The reviewer may discuss the quality of the meal, the accommodations, and other characteristics. Understanding the topics based on review and the rating helps to identify the hotels' condition. Once it is known, the hotel organization can focus on specific service parameters which are not being liked by the customers.

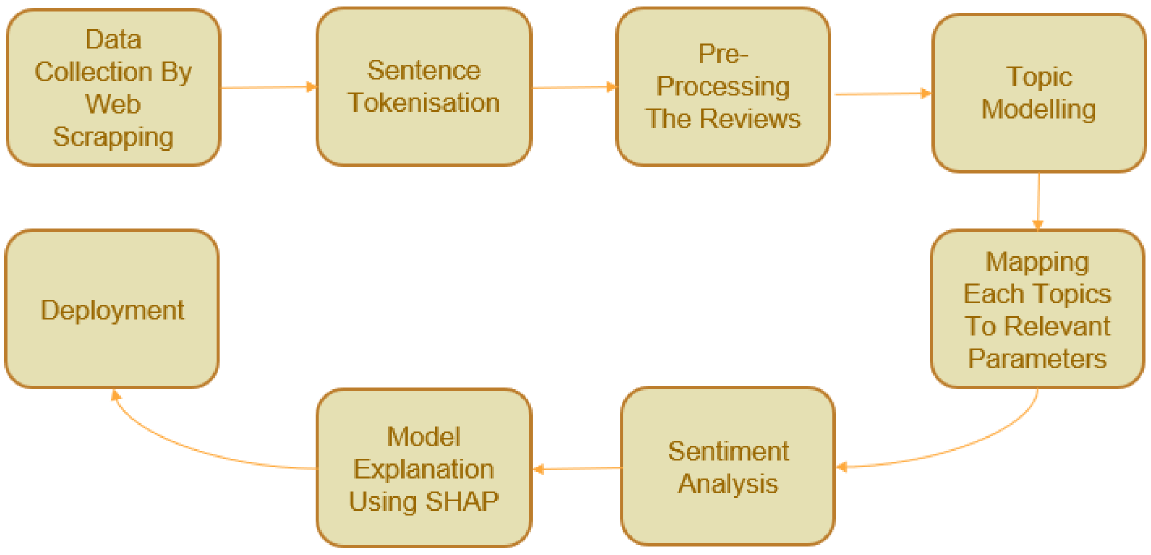
So, in this project we have tried to build a Topic Model for hotel Reviews and Ratings in order to achieve to do the Sentiment Analysis of the customers, so that it can help the hotels to focus on the service parameters to improve and attract more visitors.

## OBJECTIVE

Hotel Industry is one of the growing sectors in today's market. Owners can approach us to understand the reasons behind declining customer experience in terms of different service parameters.

Our analysis can help identify which service parameters are critical in improving customer experience. By addressing these parameters, owners can potentially improve customer satisfaction and loyalty.

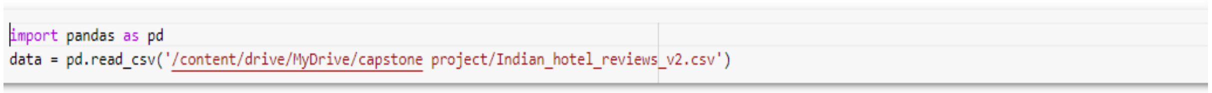
## METHODOLOGY


Data Collection By Web Scrapping
Sentence Tokenisation
Pre-Processing The Reviews
Topic Modelling
Mapping Each Topics To Relevant Parameters
Sentiment Analysis
Model Explanation Using SHAP
Deployment

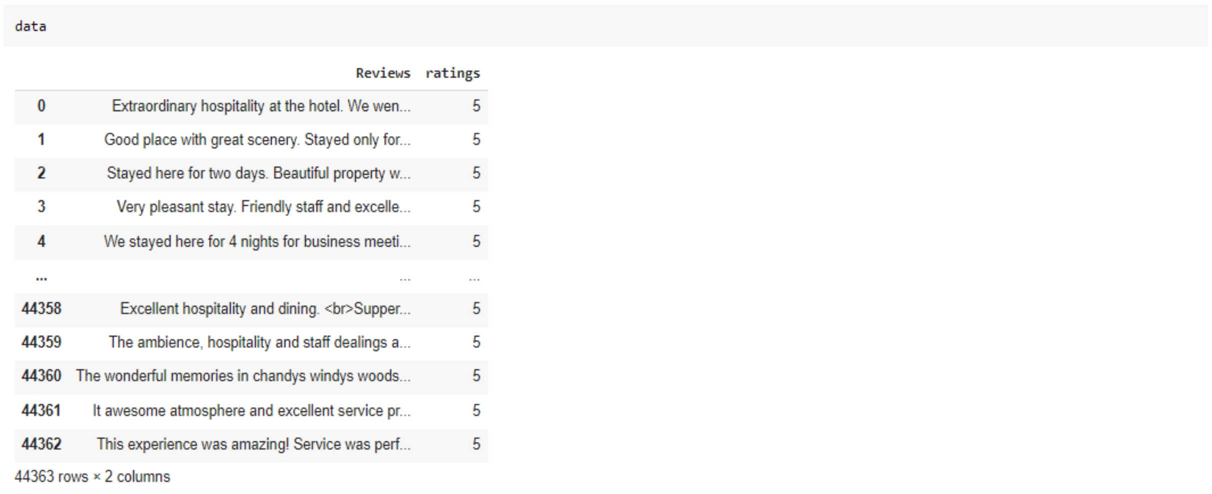
# DATA COLLECTION

## Hotel Reviews Scraping

Hotel reviews scraping from Trip Advisor website was done using the Selenium library of Python. Reviews of almost all 28 states in India across 185 hotels were extracted. For each hotel almost 300 good reviews and near about 100 bad reviews were being scraped. Finally, in we ended up scraping 44363 reviews.



```
import pandas as pd
data = pd.read_csv('/content/drive/MyDrive/capstone project/Indian_hotel_reviews_v2.csv')
```



```
data
```

```
                                                  Reviews  ratings
0          Extraordinary hospitality at the hotel. We wen...        5
1          Good place with great scenery. Stayed only for...        5
2          Stayed here for two days. Beautiful property w...        5
3          Very pleasant stay. Friendly staff and excelle...        5
4          We stayed here for 4 nights for business meeti...        5
...                                                     ...      ...
44358      Excellent hospitality and dining. <br>Supper...          5
44359      The ambience, hospitality and staff dealings a...        5
44360      The wonderful memories in chandys windys woods...        5
44361      It awesome atmosphere and excellent service pr...        5
44362      This experience was amazing! Service was perf...         5
44363 rows × 2 columns
```

## **PRE-PROCESSING**

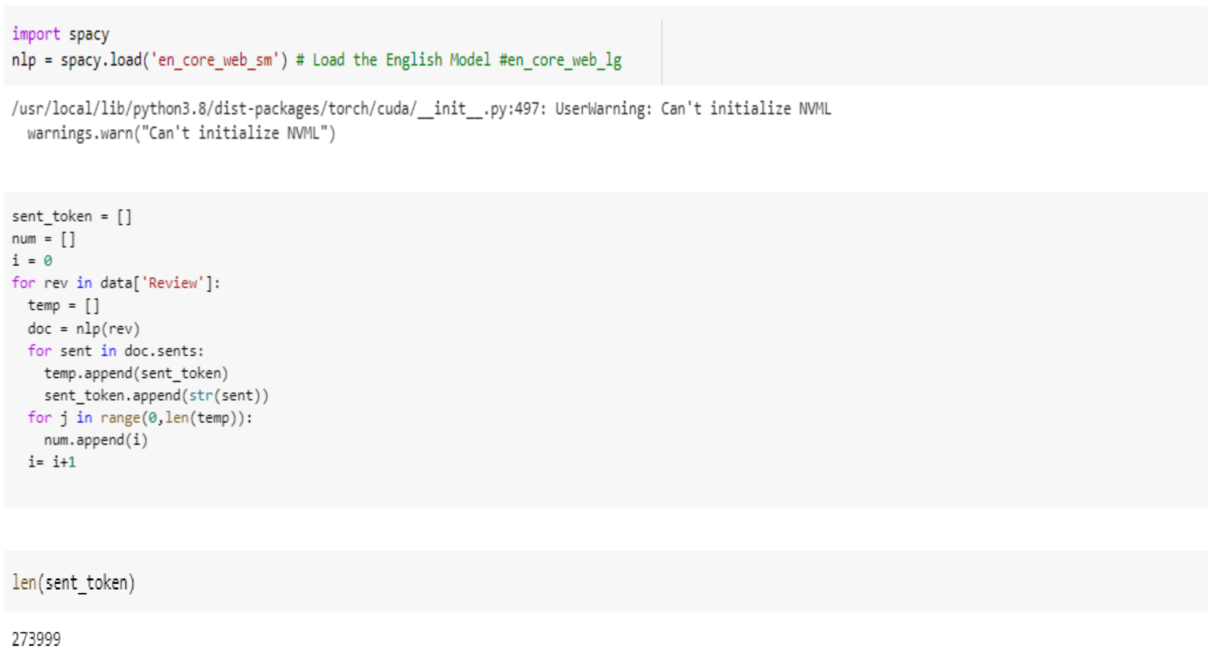
### Sentence Tokenization

The scraped reviews were stored in csv file. To proceed with pre-processing, each review was broken into sentence tokens using the en_core_web_sm from Spacy library.

Spacy is a free, open-source library for NLP in Python. It is designed to make it easy to build systems for information extraction or general-purpose natural language processing.

en_core_web_sm is English pipeline optimized for CPU. Components include tok2vec, tagger, parser, senter, ner, attribute_ruler, lemmatizer.

In the below mentioned image we imported the spacy library, did sentence tokenization, and counted the number of sentence tokens, which were created out of 44363 reviews that is 273999 tokens.



```
import spacy
nlp = spacy.load('en_core_web_sm') # Load the English Model #en_core_web_lg
```

```
/usr/local/lib/python3.8/dist-packages/torch/cuda/__init__.py:497: UserWarning: Can't initialize NVML
  warnings.warn("Can't initialize NVML")
```

```
sent_token = []
num = []
i = 0
for rev in data['Review']:
  temp = []
  doc = nlp(rev)
  for sent in doc.sents:
    temp.append(sent_token)
    sent_token.append(str(sent))
  for j in range(0,len(temp)):
    num.append(i)
  i= i+1
```

```
len(sent_token)
```

```
273999
```

### Remove Punctuations

We imported the string library to remove punctuations by building “remove punctuation” function. Python String module contains some constants, utility function, and classes for string manipulation. It’s a built-in module and we have to import it before using any of its constants and classes.

```
# now processing the data
#library that contains punctuation
import string

#defining the function to remove punctuation
def remove_punctuation(text):
    punctuationfree="".join([i for i in text if i not in string.punctuation])
    return punctuationfree
#storing the puntuation free text
new_data['sent_token']= new_data['sent_token'].apply(lambda x:remove_punctuation(x))
new_data.head()
```

| | sent_index | sent_token |
|---|---|---|
| 0 | 0 | Extraordinary hospitality at the hotel |
| 1 | 0 | We went as a family with my parents |
| 2 | 0 | Everyone really enjoyed at the resort |
| 3 | 0 | There is a good play area and evening at that ... |
| 4 | 0 | Great resort to spend time and stay at Ooty |

## Lowering the Text

We have transformed all the sentence tokens to lower-case.

```
# lowering the text
new_data['sent_token']= new_data['sent_token'].apply(lambda x: x.lower())
```

```
# creating the list of tokens
import re
list_of_tokens = [re.findall("[a-z']+",rev) for rev in new_data['sent_token']]
```

## Removing Stopwords

We have imported the NLTK library for removal of Stopwords. It is a string-oriented library which allows us to carry on with various customizations on text data.

```python
# now removing the stopwords
no_stopwords = []
import nltk
nltk.download('stopwords')
stopwords = nltk.corpus.stopwords.words('english')
for i in range(0,273999):
  temp = []
  for words in list_of_tokens[i]:
    if words not in stopwords:
      temp.append(words)
  no_stopwords.append(temp)
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

## Lemmatization of words

Lemmatization of words i.e., bringing the words to its base form using WordNetLemmatizer of NLTK corpus.

```python
# now lematizing the words
nltk.download('omw-1.4')
nltk.download('wordnet')
lemmatized = []
from nltk.stem import WordNetLemmatizer
#defining the object for Lemmatization
wordnet_lemmatizer = WordNetLemmatizer()
for i in range(0,273999):
  temp = []
  for words in no_stopwords[i]:
    temp.append(wordnet_lemmatizer.lemmatize(words))
  lemmatized.append(temp)
```

```
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

# Identifying Service Parameters of Hotel Industry

**Implementation of Topic Model - LDA**

A common topic modelling technique to extract themes from a corpus is Latent Dirichlet Allocation (LDA).

The term "Latent" denotes something that is concealed or undiscovered. In the instance of LDA topic modelling, we attempt to find the concealed topics.

A probability distribution known as the Dirichlet process has a range that is made up of other probability distributions. The Dirichlet model describes the pattern of words that are similar to one another, frequently appear together, and repeat together.

When using topic modelling, it calculates the likelihood that certain terms that are scattered throughout the manuscript will appear again. Because of this, LDA is a type of generative probabilistic model that can construct data points and estimate probabilities. Probabilities are generated by LDA for the words that are used to create themes, which are then categorised into documents.

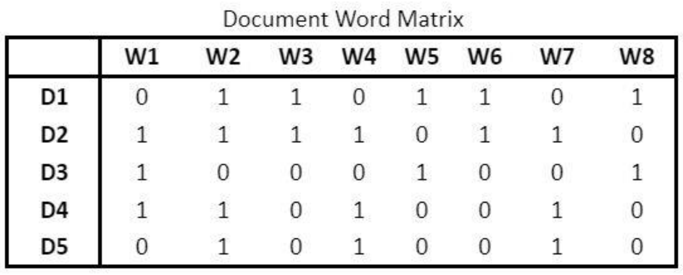
The LDA bases two critical hypotheses on:

1. Documents have a variety of themes

2. Topics contain a variety of tokens (or words).

The corpus, or collection of documents, which can be represented as a document-word (or document term matrix), also known as DTM, is subject to the aforementioned presumptions.

After pre-processing, we obtain the document word matrix shown below in Fig.1. The five documents are designated as D1, D2, D3, D4, and D5, and the words are designated as Ws; let's say there are 8 distinct words from W1 to W8. As a result, the matrix has the following dimensions: 5 * 8 (five rows and eight columns):

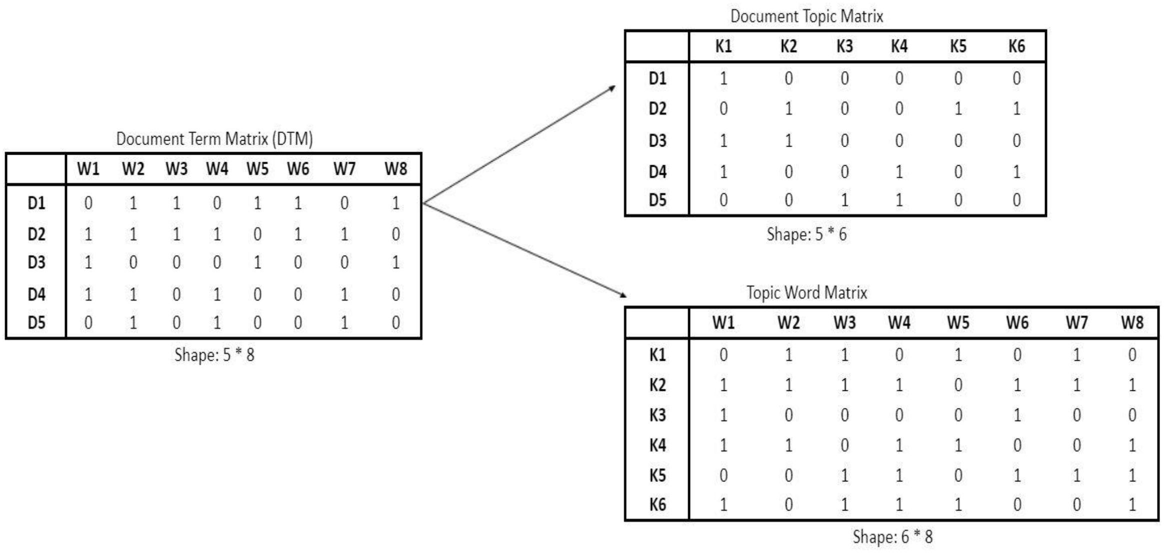


Document Word Matrix

| | W1 | W2 | W3 | W4 | W5 | W6 | W7 | W8 |
|---|---|---|---|---|---|---|---|---|
| **D1** | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| **D2** | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| **D3** | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| **D4** | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| **D5** | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |

Fig. 1

The document-word matrix described above is transformed by LDA into the document term matrix and the topic word matrix, as illustrated below in Fig.2:



Document Term Matrix (DTM)

|  | W1 | W2 | W3 | W4 | W5 | W6 | W7 | W8 |
|---|---|---|---|---|---|---|---|---|
| D1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| D2 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| D3 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| D4 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| D5 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |

Shape: 5 * 8

Document Topic Matrix

|  | K1 | K2 | K3 | K4 | K5 | K6 |
|---|---|---|---|---|---|---|
| D1 | 1 | 0 | 0 | 0 | 0 | 0 |
| D2 | 0 | 1 | 0 | 0 | 1 | 1 |
| D3 | 1 | 1 | 0 | 0 | 0 | 0 |
| D4 | 1 | 0 | 0 | 1 | 0 | 1 |
| D5 | 0 | 0 | 1 | 1 | 0 | 0 |

Shape: 5 * 6

Topic Word Matrix

|  | W1 | W2 | W3 | W4 | W5 | W6 | W7 | W8 |
|---|---|---|---|---|---|---|---|---|
| K1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| K2 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| K3 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| K4 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| K5 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| K6 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |

Shape: 6 * 8

Fig.2

The LDA model has two parameters that control the distributions:

1. Alpha (ɑ) controls per-document topic distribution, and
2. Beta (□) controls per topic word distribution

As LDA assumes that documents are a mixture of topics and topics are a mixture of words so LDA backtracks from the document level to identify which topics would have generated these documents and which words would have generated those topics.

In the first iteration, topics are randomly assigned to each word in the document. The topics are represented by the letter k. So, in our corpus, the words in the documents will be associated with some random topics like below:

- D1 = (w1 (k5), w2 (k3), w3 (k1), w4 (k2), w5 (k5), w6 (k4), w7 (k7), w8(k1))
- D2 = (w`1(k2), w`2 (k4), w`3 (k2), w`4 (k1), w`5 (k2), w`6 (k1), w`7 (k5), w`8(k3), w`9 (k7), w`10(k1))
- D3 = (w“1(k3), w“2 (k1), w“3 (k5), w“4 (k3), w“5 (k4), w“6(k1),…, w“13 (k1), w“14(k3), w“15 (k2))
- D4 = (w“`1(k4), w“`2 (k5), w“`3 (k3), w“`4 (k6), w“`5 (k5), w“`6 (k3) …, w“`10 (k3), w“`11 (k7), w“`12 (k1))
- D5 = (w““1 (k1), w““2 (k7), w““3 (k2), w““4 (k8), w““5 (k1), w““6(k8) …, w““32(k3), w““33(k6), w““34 (k5))

This gives the output as Documents with the composition of Topics and Topics composing of words:

The documents are the mixture of the topics:

- D1 = k5 + k3 + k1 + k2 + k5 + k4 + k7+ k1
- D2 = k2 + k4 + k2 + k1 + k5 + k2 + k1+ k5 + k3 + k7 + k1
- D3 = k4 + k5 + k3 + k6 + k5 + k3 + … + k3+ k7 + k1
- D3 = k1 + k7 + k2 + k8 + k1 + k8 + … + k3+ k6 + k5

The topics are the mixture of the words:

- K1 = w3 + w8 + w`4 + w`6 + w’10 + w“2 + w“6 + … + w“13 + w“`12 + w““1 + w““5
- K2 = w4 + w`1 + w`3 + w“15 + …. + w““3 + …
- K3 = w2 + w’8 + w“1 + w“4 + w“14 + w“`3 + w“`6 + … + w“`10 + w““32 + …

Similarly, LDA will give the word combinations for other topics.

Each document (D) and word (w) will be iterated through by LDA. What would that entail? It does this by calculating p1 and p2 probability for each topic (k), where:

- P1: percentage of words currently assigned to the topic (k) in the document (D)
- P2: The percentage of assignments related to the topic(k) across all documents derived from this word w. In other words, p2 represents the percentage of documents where the term (w) is also associated with the topic (k).

The formula for p1 and p2 is:

P1 = proportion (subject k / document D)

P2 = proportion (word w / topic k) are the formulas for p1 and p2, respectively.

Using these probabilities, p1 and p2, LDA now calculates a new probability, (p1*p2), and by using this product probability, LDA determines the new subject, which is the most pertinent topic for the present word.

Now, the LDA is performed for a large number of iterations for the step of choosing the new topic ‘k’ until a steady-state is obtained. The convergence point of LDA is obtained where it gives the most optimized representation of the document-term matrix and topic-word matrix.

Latent Dirichlet Allocation (LDA) does two tasks: it finds the topics from the corpus, and at the same time, assigns these topics to the document present within the same corpus. The below schematic diagram Fig.3 summarizes the process of LDA well:

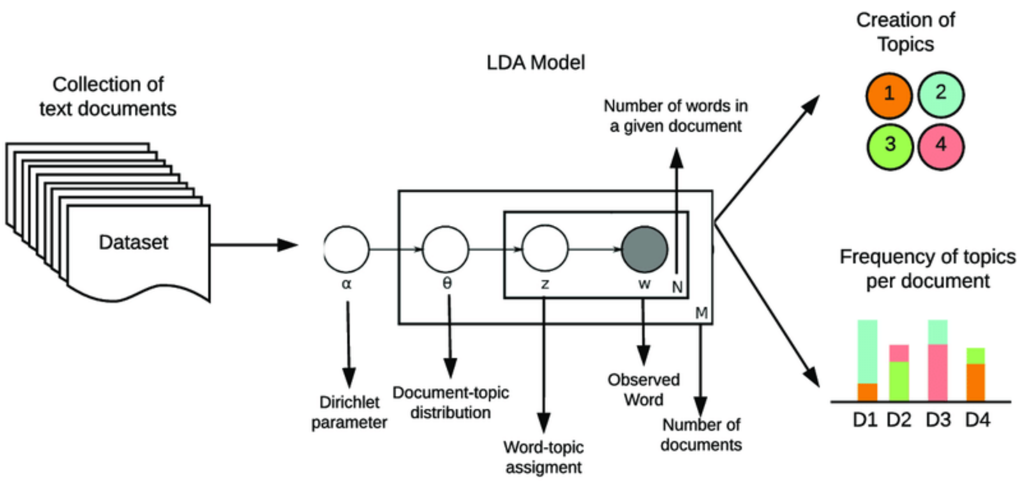


Fig.3

To summarize:

- M: is the total documents in the corpus
- N: is the number of words in the document
- w: is the Word in a document
- z: is the latent topic assigned to a word
- theta ($\theta$): is the topic distribution
- LDA models parameters: Alpha (α) and Beta (β )

From the above explanation we see that in LDA model we supply two hyperparameters to detect the optimum number of topics. In our case we got the optimum value of alpha and beta by using Bayesian Optimization of the bayes_opt library wherein the Coherence Score is the highest. The procedure is as follows:

```
from bayes_opt import BayesianOptimization, UtilityFunction
from gensim.models import CoherenceModel
import warnings
warnings.filterwarnings('ignore')
```

```
def black_box_function(k,a,b):
    lda_model = gensim.models.ldamulticore.LdaMulticore(corpus=corpora,num_topics= int(k),iterations=1000,
                                                         id2word=dictionary,alpha = a,eta = b,random_state=123)
    coherence_model_lda = CoherenceModel(model=lda_model, texts=lemmatized, dictionary=dictionary, coherence='c_v')
    return coherence_model_lda.get_coherence()
```

```
pbounds = {'k' : (2,20), 'a' : (0.001,1), 'b':(0.001,1)}
```

The optimum Coherence score obtained is as follows:

```
optimizer = BayesianOptimization(f = black_box_function,
                                 pbounds = pbounds, verbose = 2,
                                 random_state = 4)
optimizer.maximize(init_points = 5, n_iter = 25)
```

```
|   iter    |  target   |     a     |     b     |     k     |
-------------------------------------------------------------
| 1         | 0.4516    | 0.9671    | 0.5477    | 19.51     |
| 2         | 0.4815    | 0.7151    | 0.698     | 5.89      |
| 3         | 0.4897    | 0.9763    | 0.007224  | 6.554     |
| 4         | 0.4697    | 0.4354    | 0.7796    | 5.558     |
| 5         | 0.4178    | 0.8631    | 0.9834    | 4.949     |
| 6         | 0.4745    | 0.2404    | 0.2575    | 6.403     |
| 7         | 0.4926    | 1.0       | 0.7385    | 6.779     |
| 8         | 0.4891    | 1.0       | 0.1703    | 7.593     |
| 9         | 0.4806    | 0.3756    | 0.9896    | 7.791     |
| 10        | 0.4736    | 0.9644    | 0.01761   | 8.976     |
| 11        | 0.4425    | 0.05412   | 0.9585    | 11.57     |
| 12        | 0.4711    | 0.05537   | 0.01517   | 7.827     |
| 13        | 0.4742    | 0.001     | 0.001     | 15.67     |
| 14        | 0.4679    | 1.0       | 1.0       | 14.85     |
| 15        | 0.4838    | 1.0       | 1.0       | 7.457     |
| 16        | 0.4413    | 0.001     | 1.0       | 17.0      |
| 17        | 0.2772    | 0.01811   | 0.007044  | 2.017     |
| 18        | 0.4773    | 0.9919    | 0.04829   | 13.17     |
| 19        | 0.48      | 0.001     | 0.001     | 13.98     |
| 20        | 0.4663    | 0.9977    | 0.01885   | 14.75     |
| 21        | 0.4444    | 0.05366   | 0.01348   | 20.0      |
| 22        | 0.4655    | 0.001     | 1.0       | 13.7      |
| 23        | 0.4856    | 1.0       | 0.001     | 11.13     |
| 24        | 0.4643    | 1.0       | 1.0       | 10.05     |
| 25        | 0.4844    | 0.001     | 0.001     | 10.17     |
| 26        | 0.444     | 1.0       | 0.001     | 17.35     |
| 27        | 0.4908    | 0.001     | 0.001     | 12.41     |
| 28        | 0.4759    | 0.001     | 1.0       | 6.454     |
| 29        | 0.4228    | 0.001     | 1.0       | 9.052     |
| 30        | 0.4626    | 1.0       | 0.001     | 10.18     |
=============================================================
```

We stored the final model trained on the optimum value of the hyperparameters based on the Coherence Score which is as follows:

```
print(optimizer.max)
```

```
{'target': 0.49261367897796177, 'params': {'a': 1.0, 'b': 0.7385413244062596, 'k': 6.779065830463273}}
```

```
lm_topic_6 = gensim.models.ldamulticore.LdaMulticore(corpus=corpora,num_topics= 6,iterations=1000,
                                                     id2word=dictionary,alpha = 1,eta = 0.7385413244062596,random_state=123)
```

```
lm_topic_6.save('lm_topic_6')
```

Now, let us throw some light on the concept of Bayesian Optimization and Coherence Score.

## Bayesian Optimization

Bayesian Optimization is a process of finding out the set of best hyperparameters, but in contrast to random or grid search, it keeps track of past evaluation results which it uses to form a probabilistic model mapping hyperparameters to a probability of a score on the objective function:

$$P(\,score \mid hyperparameters)$$

This model, denoted as P(y | x) in the literature, is referred to as a "surrogate" for the objective function. The surrogate function is significantly simpler to optimise than the objective function, and Bayesian approaches operate by picking hyperparameters that perform best on the surrogate function to determine the next set of hyperparameters to assess on the actual objective function. In other terms, it takes the actions listed below:

1. Create a substitute probability model for the goal function.
2. Determine which hyperparameters the surrogate responds to most favourably.
3. Use the true goal function when these hyperparameters are applied.
4. Update the surrogate model to take into account the new findings.

Until the maximum number of iterations or time is reached, repeat steps 2-4.

In comparison to grid search and random search, Bayesian approaches can uncover better model settings in fewer iterations by considering hyperparameters that seem more promising given prior outcomes.

Look at the figure below for a nice illustration of what happens in Bayesian optimisation. After two evaluations, the surrogate model's initial estimate is shown in Fig.4 in black with related uncertainty shown in grey. The surrogate model's representation of the actual objective function, shown in red, is obviously inaccurate:

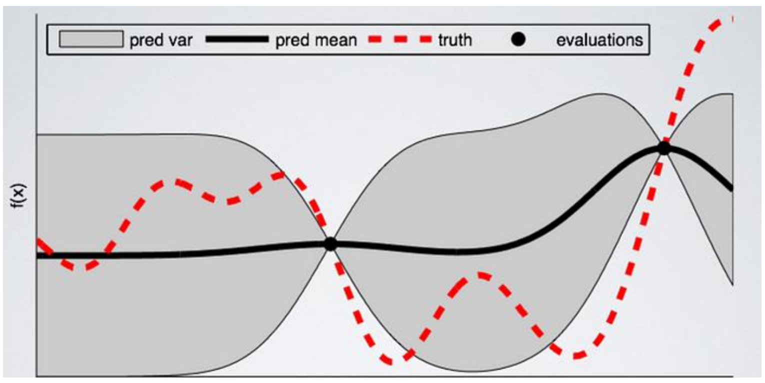


Fig.4

The Fig.5 shows the surrogate function after 8 evaluations. Now the surrogate almost exactly matches the true function. Therefore, if the algorithm selects the hyperparameters that maximize the surrogate, they will likely yield very good results on the true objective function.

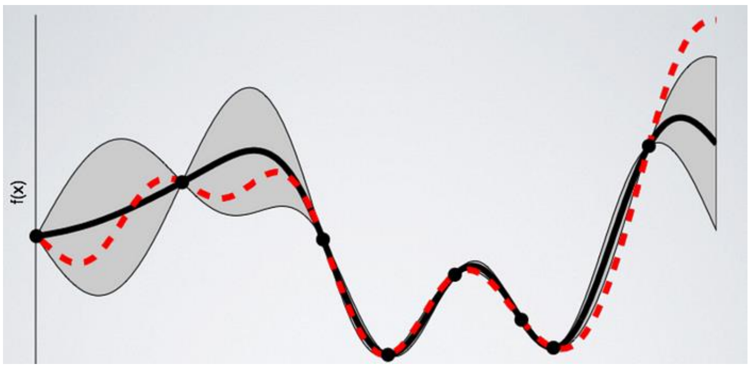
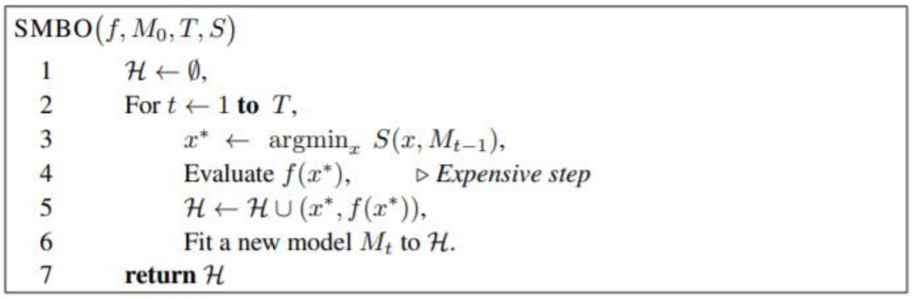


Fig.5

The Fig.6 shows the pseudo-code for SMBO which stands for Sequential Model-Based Optimization, which is another name of Bayesian Optimization. It is “sequential” because the hyperparameters are added to update the surrogate model one by one; it is “model-based” because it approximates the true objective function with a surrogate model that is cheaper to evaluate.



```
SMBO($f, M_0, T, S$)
1     $\mathcal{H} \leftarrow \emptyset$,
2     For $t \leftarrow 1$ **to** $T$,
3         $x^* \leftarrow \text{argmin}_x \; S(x, M_{t-1})$,
4         Evaluate $f(x^*)$,        ▷ *Expensive step*
5         $\mathcal{H} \leftarrow \mathcal{H} \cup (x^*, f(x^*))$,
6         Fit a new model $M_t$ to $\mathcal{H}$.
7     **return** $\mathcal{H}$
```

The explanation of the pseudo-code:

H : Observation History of (hyperparameter, score) pair

T : Max Number of Iterations

f : True Objective Function

M : Surrogate Function, which is updated whenever a new sample is added

S : Acquisition Function

x* : The Next Chosen Hyperparameter to Evaluate

The explanation of the loop is as follows:

- First, it is initiating a surrogate model and an acquisition function.
- In Line 3 of the loop in the pseudo-code, for each iteration, finds the hyperparameter x* where the acquisition function is maximized. The acquisition function is a function of the surrogate model, meaning that it is built using the

surrogate model instead of the true objective function. Here, the pseudo-code shows x* is obtained when the acquisition function is minimized. To maximize or to minimize all depends on how the acquisition function is defined. If we are using the most common acquisition function— Expected Improvement, then we should maximize it.

- Line 4: Obtain the objective function score of x* to see how this point actually performs.
- Line 5: Include the (hyperparameter x*, true objective function score) in the history of other samples.
- Line 6: Train the surrogate model using the latest history of samples.

This process is repeated until the max number of iterations is reached.

## The Most Common Acquisition Function: Expected Improvement

p(y|x): the surrogate model. y is the true objective function score and x is the hyperparameter

y*: the minimum observed true objective function score so far

y: new scores

Expected Improvement is built on top of the surrogate model, meaning that different surrogate models would result in different ways of optimizing this acquisition function.

## The Most Common Surrogate Model: Gaussian Process Model

The Gaussian Process model, which is straightforward and straightforward to optimise, is utilised as the surrogate model. P(y|x) is directly modelled by Gaussian Process. To create the multivariate Gaussian distributions, the history of (hyperparameter, true objective function score) is used as (x, y).

The new score must be lower than the current minimum score ($y<y^*$) in order to maximise the Expected Improvement result for the Gaussian Process model. This allows max(y* — y, 0) to be a significant positive integer.

The next input to the real objective function would be the set of hyperparameters that, when calculated using the multivariate Gaussian distributions, produce the highest Expected Improvement.

## Coherence Model

**What is Coherence?**

If several claims or facts concur, this is referred to as coherence. A cohesive set of facts can therefore be interpreted within a context that includes all or almost all of the facts.

**Evaluation metric – Topic Coherence**

In order to assess the coherence between topics implied by a model, the idea of topic coherence incorporates multiple measures into a framework. By calculating the degree of semantic similarity between high scoring terms in the topic, Topic Coherence measures the score of a single topic.

**Coherence Measures**

C_v measure is based on a sliding window, one-set segmentation of the top words and an indirect confirmation measure that uses normalized pointwise mutual information (NPMI) and the cosine similarity in determining the optimal number of topics.

The chart below in Fig.7 outlines the coherence score, C_v, for the number of topics across two validation sets, and a fixed alpha = 0.01 and beta = 0.1.

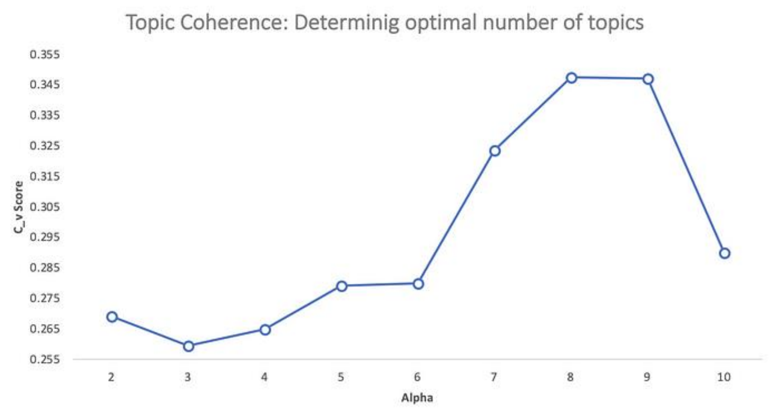


Fig.7

Initially, coherence score seems to keep increasing with the number of topics, it may make better sense to pick the model that gave the highest CV before flattening out or a major drop. We want to select the optimal alpha and beta parameters; we choose the values that yielded maximum C_v score.

Now, we further dig into some of the concepts used in Coherence Score.

## Normalized Pointwise Mutual Information

Namely, consider the name ‘Las Vegas’: it is not that frequent to read only ‘Las’ or ‘Vegas’ (in English corpora of course). The only way we see them is in the bigram Las Vegas, hence it is likely for them to form a unique concept. On the other hand, if we think of ‘New York’, it is easy to see that the word ‘New’ will probably occur very frequently in different contexts. How can we assess that the co-occurrence with York is meaningful and not as vague as ‘new dog, new cat’?

The answer lies in the Pointwise Mutual Information (PMI) criterion. The idea of PMI is that we want to quantify the likelihood of co-occurrence of two words, taking into account the fact that it might be caused by the frequency of the single words. Hence, the algorithm computes the (log) probability of co-occurrence scaled by the product of the single probability of occurrence as follows:

$$PMI(a,b) = \log\left(\frac{P(a,b)}{P(\mathrm{a})P(\mathrm{b})}\right)$$

Now, knowing that, when ‘a’ and ‘b’ are independent, their joint probability is equal to the product of their marginal probabilities, when the ratio equals 1 (hence the log equals 0), it means that the two words together don’t form a unique concept: they co-occur by chance.

On the other hand, if either one of the words (or even both of them) has a low probability of occurrence if singularly considered, but its joint probability together with the other word is high, it means that the two are likely to express a unique concept.

From above its follows that the definition for Normalized Pointwise Mutual Information is:

$$npmi \equiv \frac{pmi}{-\log p(x,y)} = \frac{\log[p(x)p(y)]}{\log p(x,y)} - 1.$$

The when there are:

- no co-occurrences, $\log p(x,y) \to -\infty$, so *nmpi* is -1,
- co-occurrences at random, $\log p(x,y) = \log[p(x)p(y)]$, so *nmpi* is 0,
- complete co-occurrences, $\log p(x,y) = \log p(x) = \log p(y)$, so *nmpi* is 1.

## Cosine Similarity

The cosine similarity index calculates how similar two vectors in an inner product space are to one another. It establishes whether two vectors are roughly pointing in the same direction by calculating the cosine of the angle between them. In text analysis, it is frequently used to gauge document similarity.

The mathematical equation of Cosine similarity between two non-zero vectors is:

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

The value of cosine similarity is restricted to the range between 0 and 1. The cosine of the angle between the two non-zero vectors A and B is measured by the similarity metric.

Let's say there was a 90-degree angle between the two vectors. The cosine similarity will then have a value of 0, indicating that the two vectors are perpendicular or orthogonal to one another. The angle between the two vectors, A and B, decreases as the cosine similarity measurement approaches 1.

Cosine similarity is a metric used in NLP to assess how similar documents are, regardless of size. It does this mathematically by computing the cosine of the angle formed by two vectors that are projected onto a multidimensional space.

The benefit of the cosine similarity is that, even though the two comparable documents are separated by a large Euclidean distance, there may still be a small angle between them. Higher similarity is associated with smaller angles.

A scenario that involves identifying the similarity between pairs of a document is a good use case for the utilization of cosine similarity as a quantification of the measurement of similarity between two objects.

Quantification of the similarity between two documents can be obtained by converting the words or phrases into a vectorized form of representation.

The vector representations of the documents can then be used within the cosine similarity formula to obtain a quantification of similarity.

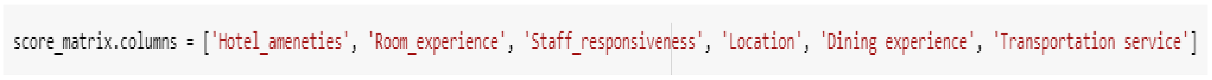
In the scenario described above, the cosine similarity of 1 implies that the two documents are exactly alike. A cosine similarity of 0 would conclude that there are no similarities between the two documents.

Post the LDA topic modelling, we went through the words belonging to each of the topics, the sentence tokens along with the respective reviews in order to assign names to the topics connecting with the service parameters of Hotel Industry. Our service parameters based on the analysis are as follows:

1. Hotel_ameneties
2. Room_experience
3. Staff_responsiveness

4. Location
5. Dining experience
6. Transportation service

The below code shows the same:



```
score_matrix.columns = ['Hotel_ameneties', 'Room_experience', 'Staff_responsiveness', 'Location', 'Dining experience', 'Transportation service']
```

# Sentiment Analysis – BERT

## BERT MODEL

In the realm of Natural Language Processing (NLP), sentiment analysis is a crucial activity. It is employed to comprehend the feelings of the clientele or populace. In the case of hotel reviews, we are attempting to examine the customers' feelings based on the hotel's service standards, whether they are favourable or bad.

This also helps the hotel industry to know more about their services and work on the feedback to further improve it.

### What is BERT?

BERT stands for Bidirectional Encoder Representations from Transformers and it is a state-of-the-art machine learning model used for NLP tasks. Jacob Devlin and his colleagues developed BERT at Google in 2018. Devlin and his colleagues trained the BERT on English Wikipedia (2,500M words) and BooksCorpus (800M words) and achieved the best accuracies for some of the NLP tasks in 2018.

Transformer architecture has encoder and decoder stack, hence called encoder-decoder architecture whereas BERT is just an encoder stack of transformer architecture. There are two variants, BERT-base and BERT-large, which differ in architecture complexity. The base model has 12 layers in the encoder whereas the Large has 24 layers.

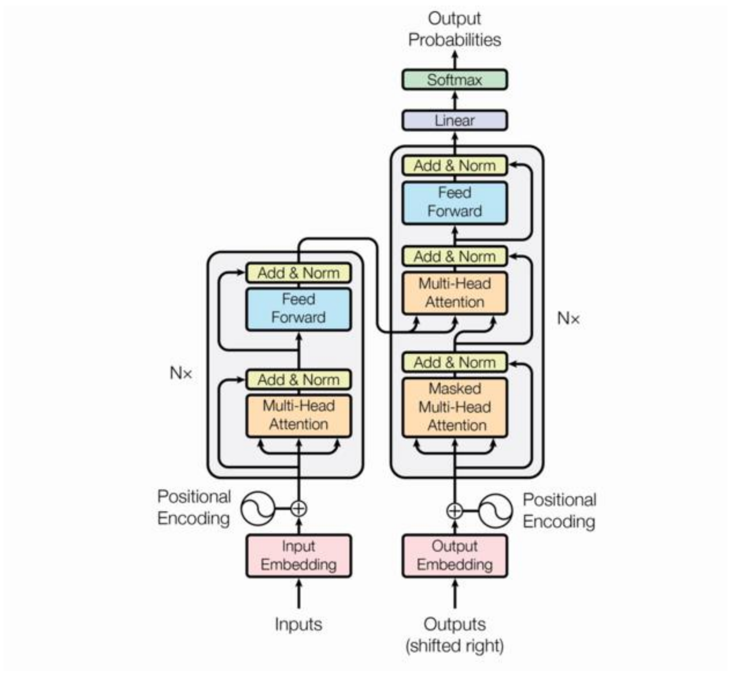
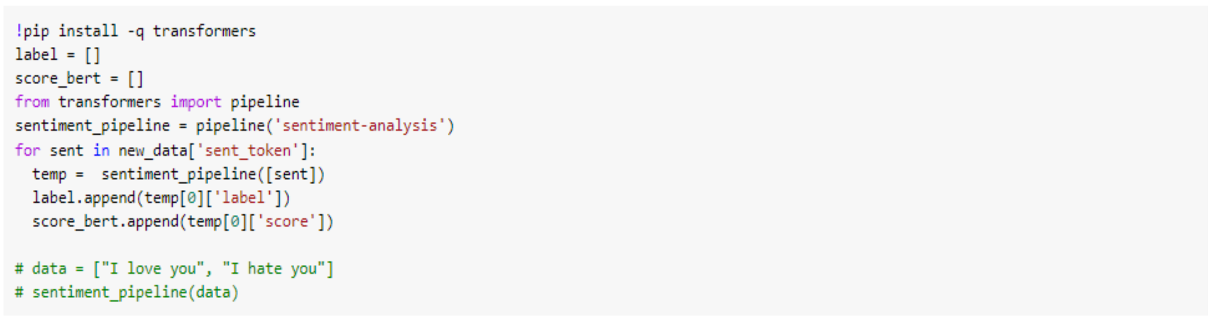


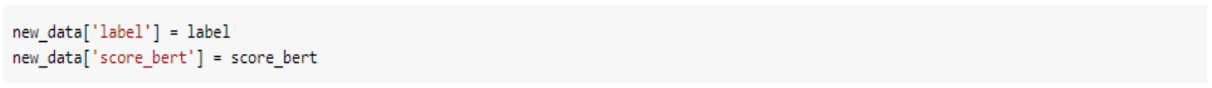
Fig.8

BERT was developed using a huge text corpus for training, which allows the architecture or model to learn the heterogeneity in data patterns and perform effectively across a variety of NLP applications. Being bidirectional, BERT gathers knowledge from both the left and the right sides of the context of a token during training.

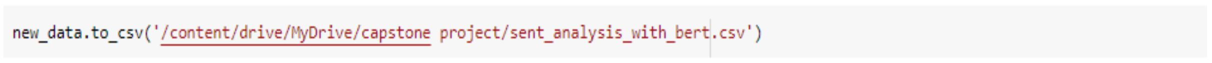
The below mentioned codes takes you to the data frame post training the BERT model:



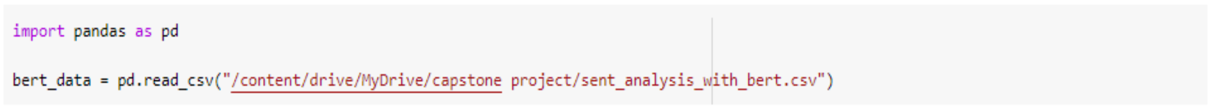
```
!pip install -q transformers
label = []
score_bert = []
from transformers import pipeline
sentiment_pipeline = pipeline('sentiment-analysis')
for sent in new_data['sent_token']:
  temp =  sentiment_pipeline([sent])
  label.append(temp[0]['label'])
  score_bert.append(temp[0]['score'])

# data = ["I love you", "I hate you"]
# sentiment_pipeline(data)
```



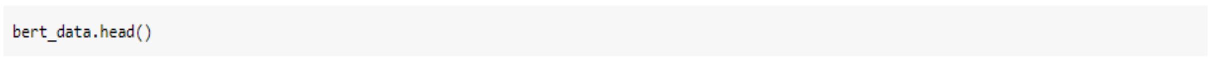
```
new_data['label'] = label
new_data['score_bert'] = score_bert
```



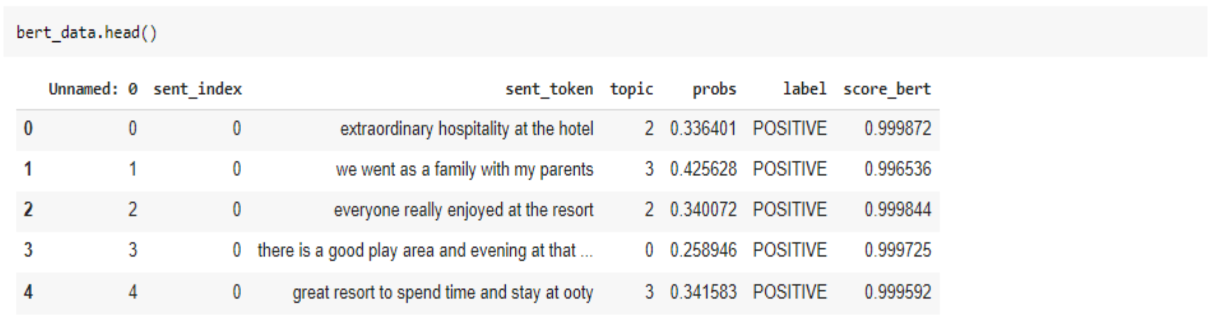
```
new_data.to_csv('/content/drive/MyDrive/capstone project/sent_analysis_with_bert.csv')
```



```
import pandas as pd

bert_data = pd.read_csv("/content/drive/MyDrive/capstone project/sent_analysis_with_bert.csv")
```



```
bert_data.head()
```



```
bert_data.head()
```

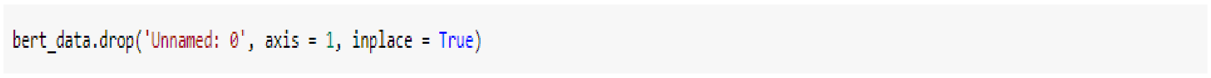
| | Unnamed: 0 | sent_index | sent_token | topic | probs | label | score_bert |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | extraordinary hospitality at the hotel | 2 | 0.336401 | POSITIVE | 0.999872 |
| 1 | 1 | 0 | we went as a family with my parents | 3 | 0.425628 | POSITIVE | 0.996536 |
| 2 | 2 | 0 | everyone really enjoyed at the resort | 2 | 0.340072 | POSITIVE | 0.999844 |
| 3 | 3 | 0 | there is a good play area and evening at that ... | 0 | 0.258946 | POSITIVE | 0.999725 |
| 4 | 4 | 0 | great resort to spend time and stay at ooty | 3 | 0.341583 | POSITIVE | 0.999592 |

In the above data frame, we got the following features post BERT training model:

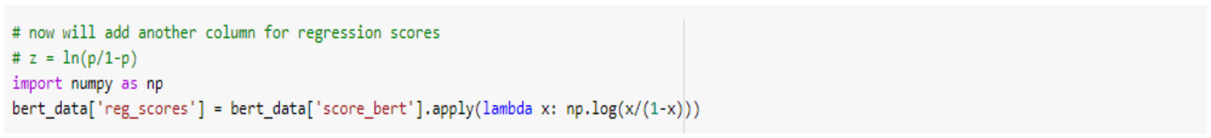
- Sentence index
- The sentence tokens from each review
- The topic to which the sentence tokens has been assigned
- The respective probability of the sentence tokens being in that topic,
- The BERT model  analysis of the sentence token being "POSITIVE" or "NEGATIVE"
- The respective probability scores.

# POST – SENTIMENT ANALYSIS STEPS

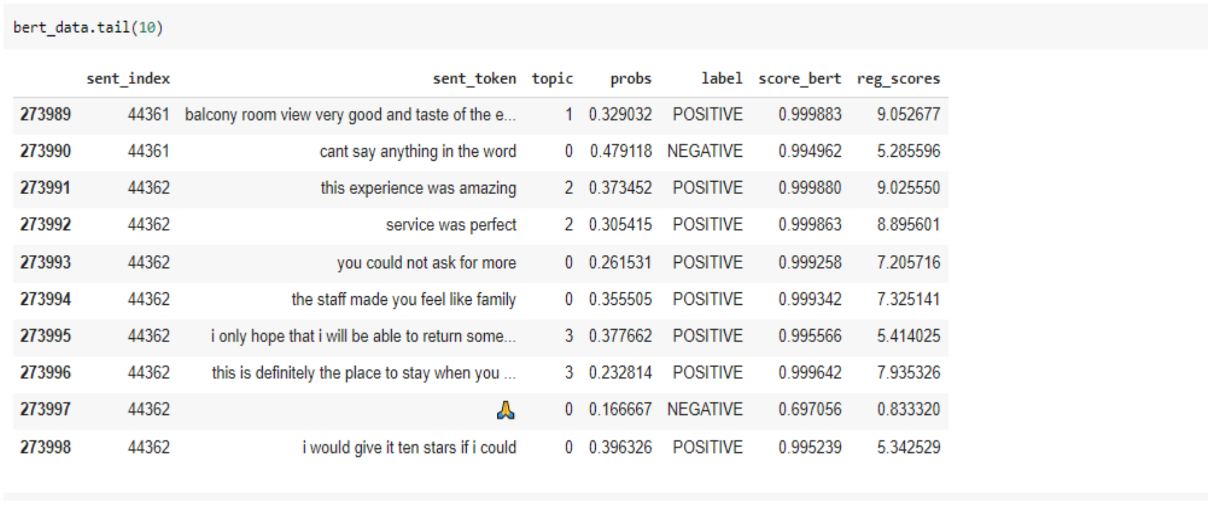
Bert is a Deep Learning Model, which gives the labels in either positive or negative and uses the sigmoid function to gives result in probability. Ratings cannot be defined properly through probability scores, so, we converted the probability scores to regression scores by doing log of the bert_scores i.e., log(x/(1-x) in order to define the neutral sentence tokens more appropriately through regression scores.



```
bert_data.drop('Unnamed: 0', axis = 1, inplace = True)
```



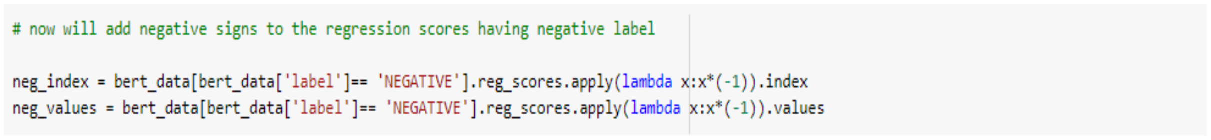
```
# now will add another column for regression scores
# z = ln(p/1-p)
import numpy as np
bert_data['reg_scores'] = bert_data['score_bert'].apply(lambda x: np.log(x/(1-x)))
```



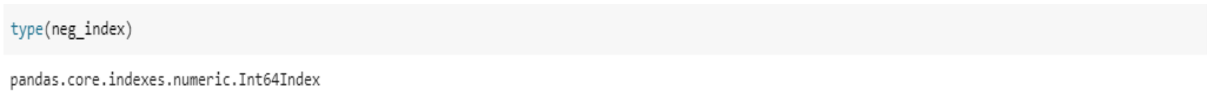
```
bert_data.tail(10)
```

| | sent_index | sent_token | topic | probs | label | score_bert | reg_scores |
|---|---|---|---|---|---|---|---|
| 273989 | 44361 | balcony room view very good and taste of the e... | 1 | 0.329032 | POSITIVE | 0.999883 | 9.052677 |
| 273990 | 44361 | cant say anything in the word | 0 | 0.479118 | NEGATIVE | 0.994962 | 5.285596 |
| 273991 | 44362 | this experience was amazing | 2 | 0.373452 | POSITIVE | 0.999880 | 9.025550 |
| 273992 | 44362 | service was perfect | 2 | 0.305415 | POSITIVE | 0.999863 | 8.895601 |
| 273993 | 44362 | you could not ask for more | 0 | 0.261531 | POSITIVE | 0.999258 | 7.205716 |
| 273994 | 44362 | the staff made you feel like family | 0 | 0.355505 | POSITIVE | 0.999342 | 7.325141 |
| 273995 | 44362 | i only hope that i will be able to return some... | 3 | 0.377662 | POSITIVE | 0.995566 | 5.414025 |
| 273996 | 44362 | this is definitely the place to stay when you ... | 3 | 0.232814 | POSITIVE | 0.999642 | 7.935326 |
| 273997 | 44362 | 🙏 | 0 | 0.166667 | NEGATIVE | 0.697056 | 0.833320 |
| 273998 | 44362 | i would give it ten stars if i could | 0 | 0.396326 | POSITIVE | 0.995239 | 5.342529 |

The sentence tokens for which the labels are negative, their regression scores have been multiplied by -1.

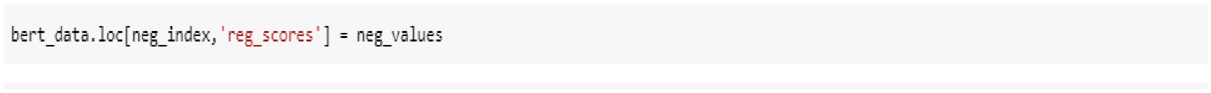


```
# now will add negative signs to the regression scores having negative label

neg_index = bert_data[bert_data['label']== 'NEGATIVE'].reg_scores.apply(lambda x:x*(-1)).index
neg_values = bert_data[bert_data['label']== 'NEGATIVE'].reg_scores.apply(lambda x:x*(-1)).values
```



```
type(neg_index)
```

```
pandas.core.indexes.numeric.Int64Index
```



```
bert_data.loc[neg_index,'reg_scores'] = neg_values
```

```
bert_data[bert_data['reg_scores'] == bert_data['reg_scores'].min() ]
```

```
       sent_index                         sent_token  topic     probs     label  score_bert  reg_scores
70881       11249  totally disappointed and a definite letdown      0  0.386491  NEGATIVE    0.999826   -8.659107
```

A new data frame has been created where we are storing only the sentence index, topics and the regression scores. Now, if two or more sentence tokens from one review belong to the same topic, we have taken the mean of those regression scores.

```
# creating another data frame storing the sent_index, topic, scores

temp = bert_data[['sent_index', 'topic', 'reg_scores']]
```

```
temp2 = temp.groupby(['sent_index', 'topic']).mean()
```

```
temp2
```

```
                   reg_scores
sent_index  topic
         0      0    6.898367
                2    8.846902
                3    7.485648
                5    1.586616
         1      1    8.785276
       ...    ...         ...
     44361      1    9.052677
                2    9.072217
     44362      0    4.760017
                2    8.960575
                3    6.674676
143513 rows × 1 columns
```

The above data frame was converted to pivot table in order to store the review wise, topic wise regression scores.

```
score_matrix = pd.pivot_table(data=df_pivot, index='sent_index', columns='topic', values='reg_scores', fill_value=0)
```

```
score_matrix.columns = ['Hotel_ameneties', 'Room_experience', 'Staff_responsiveness',
                        'Location', 'Dining experience', 'Transportation service']
```

```
score_matrix.tail()
```

| sent_index | Hotel_ameneties | Room_experience | Staff_responsiveness | Location | Dining experience | Transportation service |
|---|---|---|---|---|---|---|
| **44358** | 0.000000 | 6.605692 | 8.607698 | 0.000000 | 8.596513 | 0.0 |
| **44359** | 0.000000 | 0.000000 | 8.692202 | 0.000000 | 0.000000 | 0.0 |
| **44360** | 0.000000 | 0.000000 | 8.263286 | 5.587883 | 0.000000 | 0.0 |
| **44361** | -5.285596 | 9.052677 | 9.072217 | 0.000000 | 0.000000 | 0.0 |
| **44362** | 4.760017 | 0.000000 | 8.960575 | 6.674676 | 0.000000 | 0.0 |

In the above pivot table, we observe that the regression scores of some of the reviews for some of the Hotel service parameters turns out to be 0, which cannot be actually true. Since a customer making visit to a hotel will definitely have some opinion for the mentioned service parameters, so we are estimating the 0 value regression scores using the Iterative Imputer, Bayesian Ridge.

```
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
from sklearn.linear_model import LinearRegression

# lr = LinearRegression()
# imputation with Bayesian Ridge which also comes with default
imp = IterativeImputer(missing_values= 0, max_iter=10, verbose=2, imputation_order='roman',random_state=0)
X_br=imp.fit_transform(score_matrix)
X_br
```

```
# simpleimputer with Bayesian Ridge
sm_imp_br = pd.DataFrame(X_br, columns = ['Hotel_ameneties', 'Room_experience', 'Staff_responsiveness',
                                          'Location', 'Dining experience', 'Transportation service'])
```

```
sm_imp_br.tail(20)
```

| | Hotel_ameneties | Room_experience | Staff_responsiveness | Location | Dining experience | Transportation service |
|---|---|---|---|---|---|---|
| **44343** | 4.929043 | 6.082725 | 8.936465 | 5.413013 | 7.046925 | 3.813577 |
| **44344** | 6.949794 | 8.820985 | 9.345813 | 7.104495 | 8.973408 | 5.420721 |
| **44345** | 5.461898 | 6.884208 | 8.513015 | 7.449536 | 7.196591 | 4.424485 |
| **44346** | 6.789341 | 9.003978 | 7.807496 | 8.798617 | 8.152850 | 5.446214 |
| **44347** | 7.937997 | 7.750592 | 8.983475 | 6.604187 | 8.247464 | 5.287475 |
| **44348** | 1.474260 | 4.200196 | 7.273865 | 5.987189 | 7.636171 | 2.533178 |
| **44349** | -8.221877 | 2.273540 | 3.821853 | 8.502917 | 1.143449 | -5.034103 |
| **44350** | 4.999220 | 6.173070 | 8.955460 | 5.576564 | 7.113666 | 3.886618 |
| **44351** | 3.893825 | 6.687356 | 5.953942 | 4.613787 | 0.690907 | 6.060989 |
| **44352** | 5.377567 | 6.682818 | 8.089914 | 7.437486 | 7.306660 | 4.334485 |
| **44353** | 5.727069 | 6.334018 | 8.231076 | 6.126762 | 9.035511 | 4.367103 |
| **44354** | 8.935558 | 8.283552 | 9.028354 | 6.866587 | 8.278125 | 5.708846 |
| **44355** | 4.927237 | 6.090047 | 8.857212 | 5.520796 | 7.038473 | 3.823786 |
| **44356** | 6.748763 | 8.501155 | 9.119997 | 8.854669 | 8.458502 | 5.518166 |

We have added another column in the score matrix for the Ratings of the reviews from our dataset and with respect to the added Ratings in the review Label column, the scores with rating as >=4 are labelled as “Good” review whereas the scores with rating <=3 are labelled as “Bad” review.

```
def map_function(x):
  if x >=4:
    return 'good'
  else:
    return 'bad'
```

```
sm_imp_br['Review_label'] = sm_imp_br['Review_label'].map(map_function)
```

```
sm_imp_br['Review_label'].value_counts()
```

```
good     35789
bad       8574
Name: Review_label, dtype: int64
```

After dropping the rating column from our data frame, we trained Random Forest classifier with “Hotel_amenities”, “Room_experience”, “Staff_responsiveness”, “Location”, “Dining experience”, “Transportation service” as independent variables and “Review_label” as the Target variable.

We found out the best set of hyperparameters using Bayesian optimization and we trained our model as follows:

```
# from sklearn.ensemble import RandomForestClassifier
rf2 = RandomForestClassifier(n_estimators= 200, min_samples_split= 8,min_samples_leaf = 6,class_weight= 'balanced', bootstrap =True)
rf2.fit(X, y)
```

```
RandomForestClassifier
RandomForestClassifier(class_weight='balanced', min_samples_leaf=6,
                       min_samples_split=8, n_estimators=200)
```

The hyperparameters used are as follows:

min_samples_split = 8

min_samples_leaf = 6

class_weight = ‘balanced’

n_estimators = 200

```
y.value_counts()
```

```
good    35789
bad      8574
Name: Review_label, dtype: int64
```

Since, our dataset was highly imbalance with 35789 good labels and 8574 bad labels. So, to handle the imbalance during training, we specified the hyperparameter class_weight as 'balanced'. The "balanced" mode uses the values of the target to automatically adjust weights inversely proportional to class frequencies in the input data as:

n_samples / (n_classes * np.bincount(y))

n_samples = total number of datapoints in the training set

n_classes = 2 (Binary Classification)

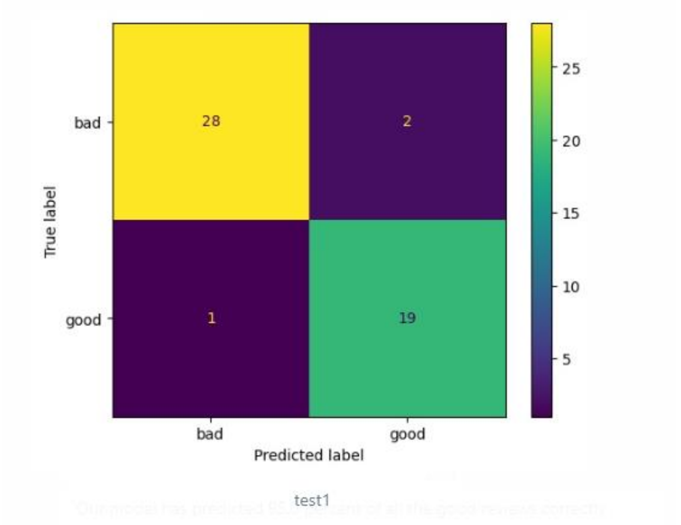
np.bincount(y) = the number of datapoints for each label.

For example, a datapoint having label as good, will be given a weightage of (44363)/(2*35789) = 0.62

Whereas, a datapoint having label as bad, will be given a weightage of 2.58.

## Checking Model Performance on Unseen Data

### TEST 1

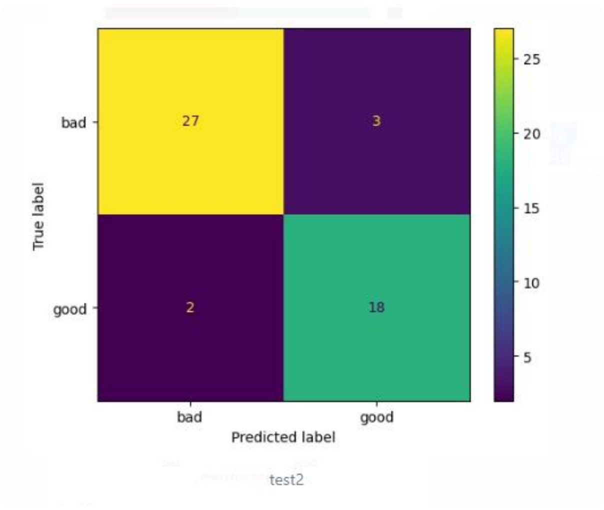


Our model has predicted 95.0 percent of all the good reviews correctly.

Our model has predicted 93.33 percent of all the bad reviews correctly.

Overall Accuracy is 94%.

**TEST 2**

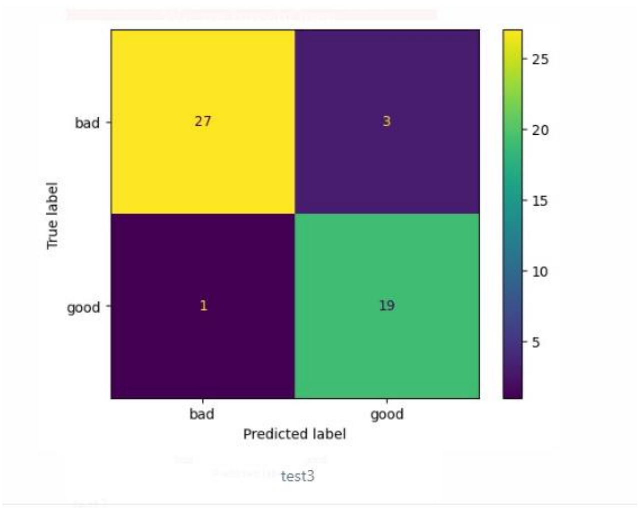


Our model has predicted 90.0 percent of all the good reviews correctly.

Our model has predicted 90.0 percent of all the bad reviews correctly.

Overall Accuracy is 90%.

**TEST 3**



Our model has predicted 95.0 percent of all the good reviews correctly.

Our model has predicted 90.0 percent of all the bad reviews correctly.

Overall Accuracy is 92%.

## MODEL EXPLANATION USING SHAP

The state-of-the-art technology for machine learning explainability is called SHAP, which stands for SHapley Additive exPlanations. It is a genius approach to reverse-engineer the output of any predictive algorithm, and it was initially published in 2017 by Lundberg and Lee.

What SHAP does is quantifying the contribution that each feature brings to the prediction made by the model. For example, we imagine a machine learning model that predicts the income of a person knowing age, gender and job of the person represented by Fig.9 below.

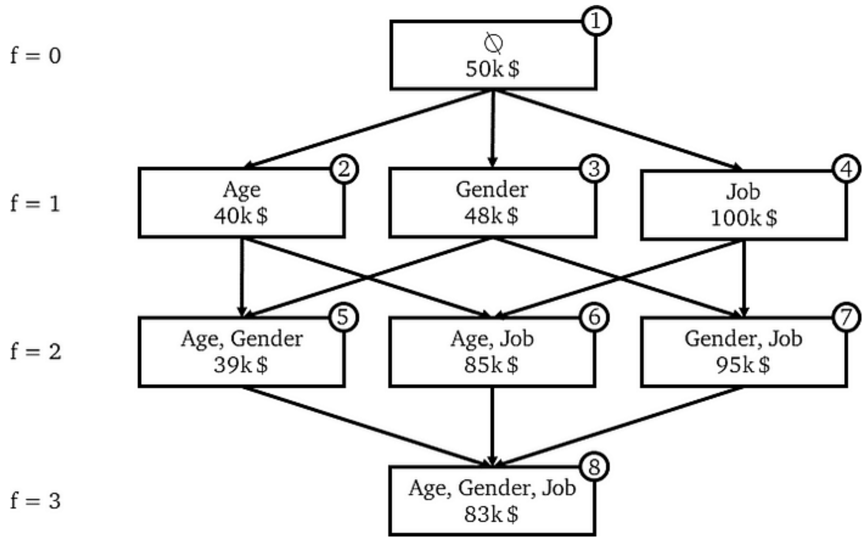


Fig.9

Each node represents a coalition of features. Each edge represents the inclusion of a feature not present in the previous coalition. SHAP requires to train a distinct predictive model for each distinct coalition in the power set, meaning 2 ^ F models.

We have 2 ^ F = 2 ^ 3 = 8 possible coalitions of features. These models are completely equivalent to each other in the sense what concerns the hyperparameters and their training data. The only thing that changes is the set of features.

As there are 8 possible coalition features, so we train 8 distinct models on the same training data. We can then take a new observation (let us call it $x_0$) and see what the 8 different models predict for the same observation $x_0$.

In the above Fig.9, Two nodes connected by an edge differ for just one feature, in the sense that the bottom one has exactly the same features of the upper one plus an additional feature that the upper one did not have. The gap between the predictions of two connected nodes can be imputed to the effect of that additional feature. This is called “marginal contribution” of a feature. Each edge represents the marginal contribution brought by a feature to a model.

Imagine that we are in node 1, which is the model with no features. This model will simply predict the average income of all the training observations (50k \$). If we move to node 2, which is a model with just one feature (Age), the prediction for $x_0$ is now 40k \$.

Thus, the marginal contribution brought by Age to the model containing only Age as a feature is -10k \$. In formula:

$$MC_{Age,\{Age\}}(x_0) = Predict_{\{Age\}}(x_0) - Predict_{\varnothing}(x_0) = 40k\$ - 50k\$ = -10k\$$$

To obtain the overall effect of Age on the final model (i.e. the SHAP value of Age for $x_0$) it is necessary to consider the marginal contribution of Age in all the models where Age is present. In our tree representation, this means to consider all the edges connecting two nodes such that:

- the upper one does not contain Age, and
- the bottom one contains Age.

In the following Fig.10, such edges have been highlighted in red.

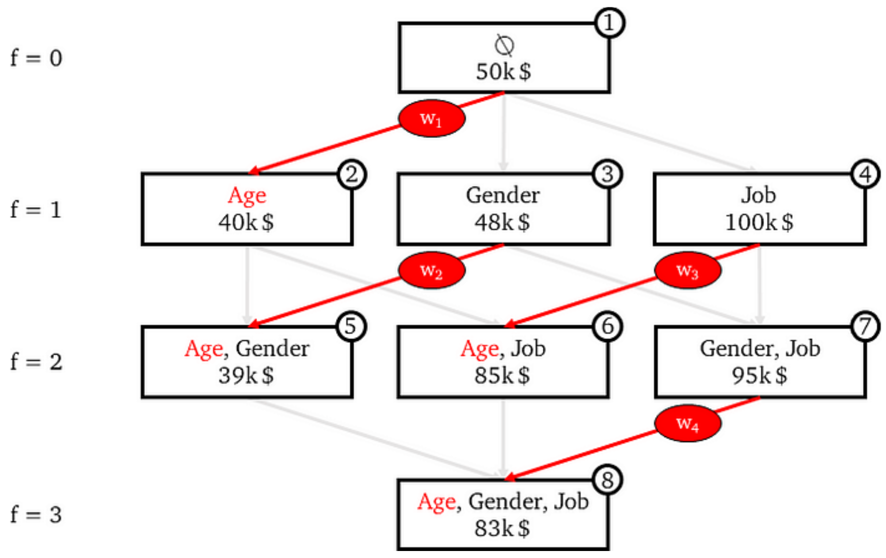


Fig.10

All these marginal contributions are then aggregated through a weighted average. In formula:

$$\begin{aligned} SHAP_{Age}(x_0) = \; & w_1 \times MC_{Age,\{Age\}}(x_0) + \\ & w_2 \times MC_{Age,\{Age,Gender\}}(x_0) + \\ & w_3 \times MC_{Age,\{Age,Job\}}(x_0) + \\ & w_4 \times MC_{Age,\{Age,Gender,Job\}}(x_0) \end{aligned}$$

where $w_1+w_2+w_3+w_4=1$.

**Determination of the weights:**

In other words, the sum of all the weights on the same "row" should equal to the sum of all the weights on any other "row".

In our example, this implies: $w_1 = w_2 + w_3 = w_4$.

All the weights of marginal contributions to f-feature-models should be equal to each other, for each f. In other words, all the edges on the same "row" should be equal to each other.

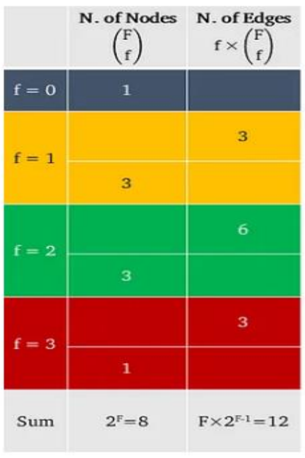
In our example, this means: $w_2 = w_3$.

the weight of a marginal contribution to a f-feature-model is the reciprocal of the number of possible marginal contributions to all the f-feature-models. The problem boils down to counting the number of possible f-feature-models, given f and knowing that the total number of features is F. This is simply the definition of binomial coefficient.

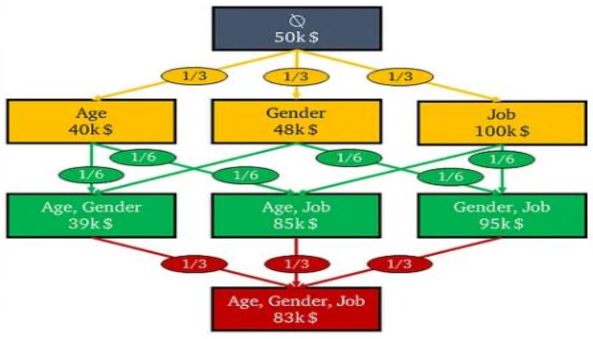
the number of edges in each "row" — is:

$$f \times \binom{F}{f}$$

It is enough to take the reciprocal of this and we have the weight of a marginal contribution to a f-feature-model.



| | N. of Nodes $\binom{F}{f}$ | N. of Edges $f \times \binom{F}{f}$ |
|---|---|---|
| f = 0 | 1 | |
| f = 1 | | 3 |
| | 3 | |
| f = 2 | | 6 |
| | 3 | |
| f = 3 | | 3 |
| | 1 | |
| Sum | $2^F=8$ | $F\times 2^{F-1}=12$ |



How weigths are obtained from the number of edges

Fig.11

Now, we have all the elements required for calculating the SHAP value of Age for $x_0$:

$$\begin{aligned} SHAP_{Age}(x_0) &= [(1\times\binom{3}{1})]^{-1}\times MC_{Age,\{Age\}}(x_0)\ + \\ &\quad [(2\times\binom{3}{2})]^{-1}\times MC_{Age,\{Age,Gender\}}(x_0)\ + \\ &\quad [(2\times\binom{3}{2})]^{-1}\times MC_{Age,\{Age,Job\}}(x_0)\ + \\ &\quad [(3\times\binom{3}{3})]^{-1}\times MC_{Age,\{Age,Gender,Job\}}(x_0)\ + \\ &= \frac{1}{3}\times(-10k\$)+\frac{1}{6}\times(-9k\$)+\frac{1}{6}\times(-15k\$)+\frac{1}{3}\times(-12k\$) \\ &= -11.33k\$ \end{aligned}$$

Generalizing to any feature and any F, we obtain the formula reported by Slundberg and Lee:

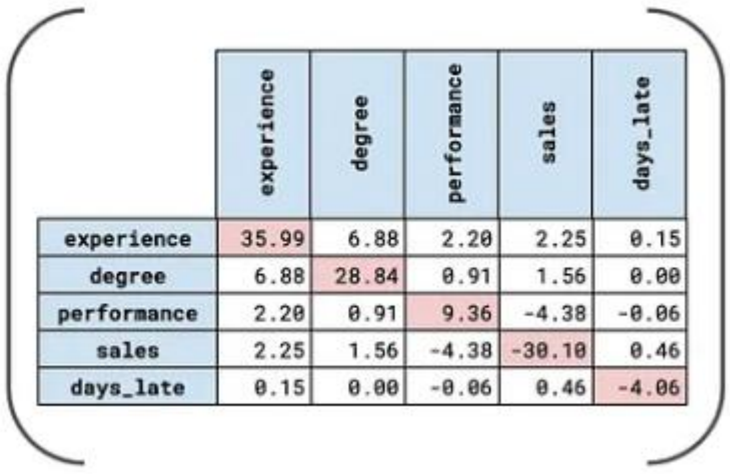
$$SHAP_{feature}(x) = \sum_{set:feature\in set} [|set|\times\binom{F}{|set|}]^{-1}[Predict_{set}(x) - Predict_{set\setminus feature}(x)]$$

**SHAP Interaction Effect:**

Individual predictions provided by a model are each explained by a set of SHAP values. It accomplishes this by outlining how each component affected the final prediction. The contributions are divided into their main and interaction effects in SHAP interaction values.

Shap interaction values return a multi-dimensional array, for example: if we have 1000 datapoints and 5 features then it will return a 1000matrices each of size 5x5.

The matrix tells us how much each factor contributed to the model's prediction when compared to the mean prediction. This is a similar interpretation to standard SHAP values except the contributions are broken down into main and interaction effects. The main effects are given on the diagonals and the interaction effects are given on the off-diagonals.

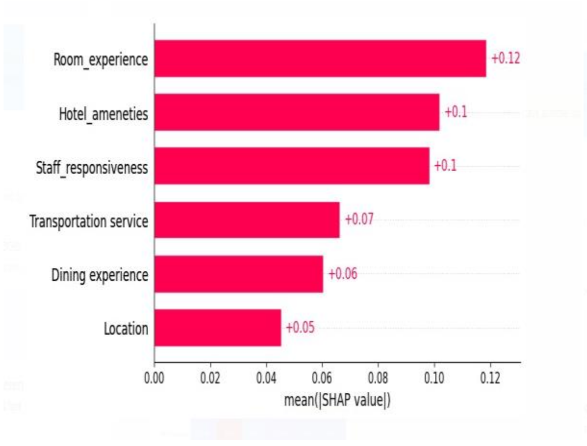


| | experience | degree | performance | sales | days_late |
|---|---|---|---|---|---|
| experience | 35.99 | 6.88 | 2.20 | 2.25 | 0.15 |
| degree | 6.88 | 28.84 | 0.91 | 1.56 | 0.00 |
| performance | 2.20 | 0.91 | 9.36 | -4.38 | -0.06 |
| sales | 2.25 | 1.56 | -4.38 | -30.10 | 0.46 |
| days_late | 0.15 | 0.00 | -0.06 | 0.46 | -4.06 |

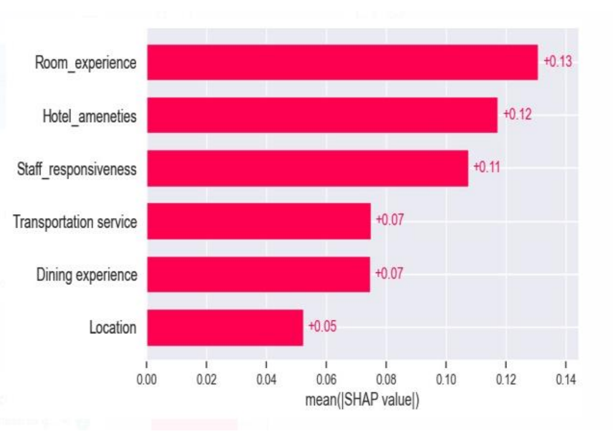
Fig.12

## State wise Analysis

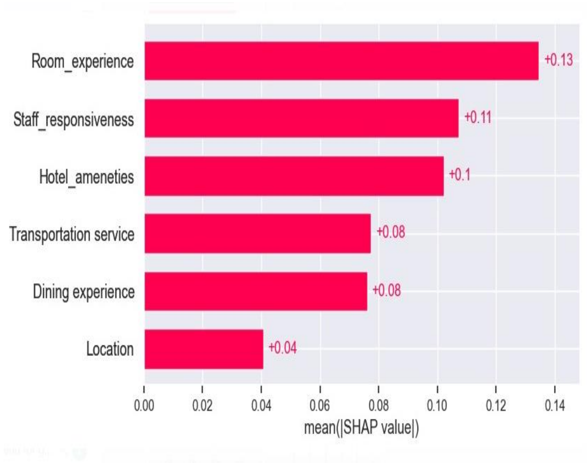
WEST BENGAL



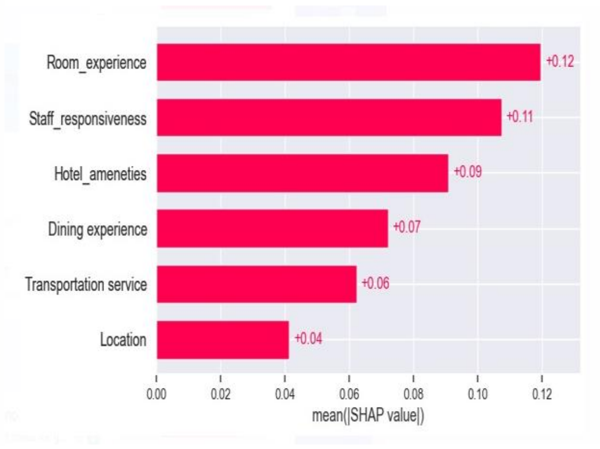
HIMACHAL PRADESH



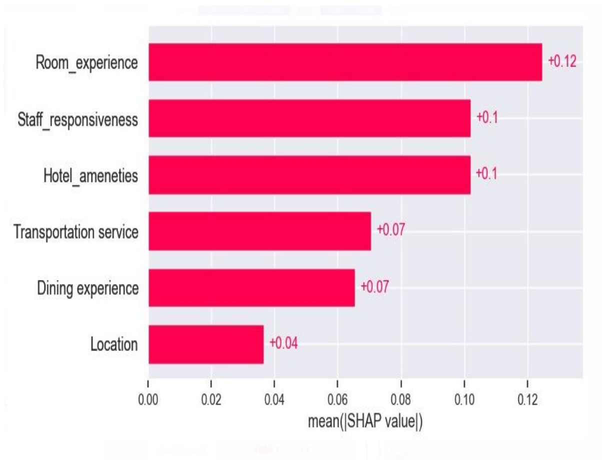
GOA



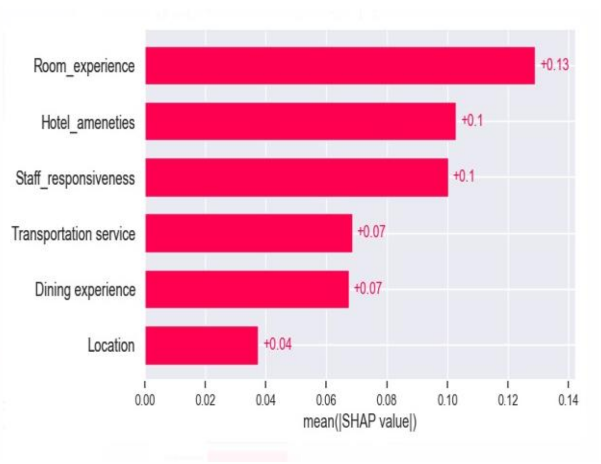
RAJASTHAN



SIKKIM



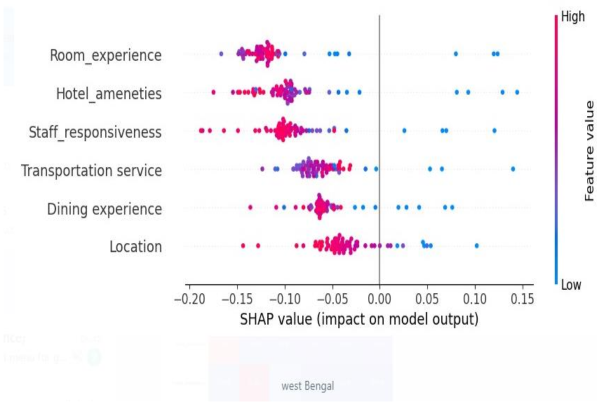
KARNATAKA



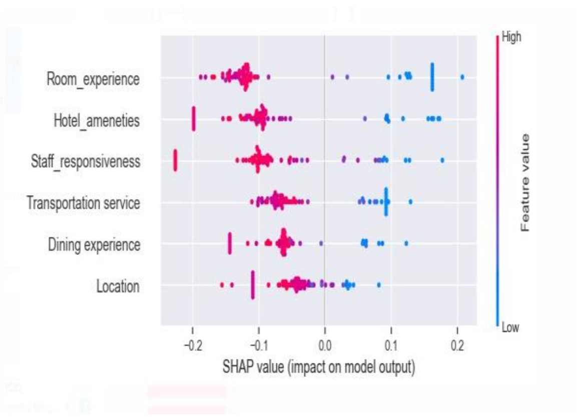
The above bar plots show the mean shap values over all the datapoints. Y-axis represents the hotel service parameters and X-axis represents the mean of the absolute shap values. From the above plots we can conclude the following:

- In all the states, the feature “Room Experience” contributes the most for the review being bad.
- We observe that in West Bengal, Himachal Pradesh and Karnataka, the second feature which contributes the most for the review being bad is the “Hotel_amenities”.
- Whereas in Goa, Rajasthan and Sikkim, the second feature which contributes the most for the review being bad is the “Staff_responsiveness”.
- In the state of Rajasthan, we see “Dining experience ” is in the 4th position playing a crucial role in making the reviews bad. Whereas in case of the other remaining states, we observe that “Transportation service” is playing a crucial role for the same. Since Rajasthan is an important tourist destination and many people go there to experience luxury, hence the customers’ expectation of good dining experience is high, so this feature is playing a crucial role being in the 4th position.

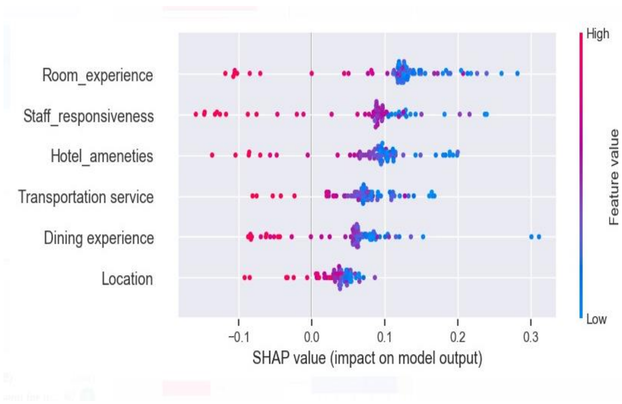
WEST BENGAL



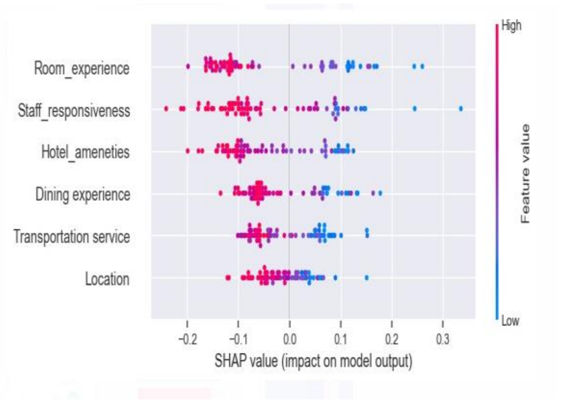
HIMACHAL PRADESH



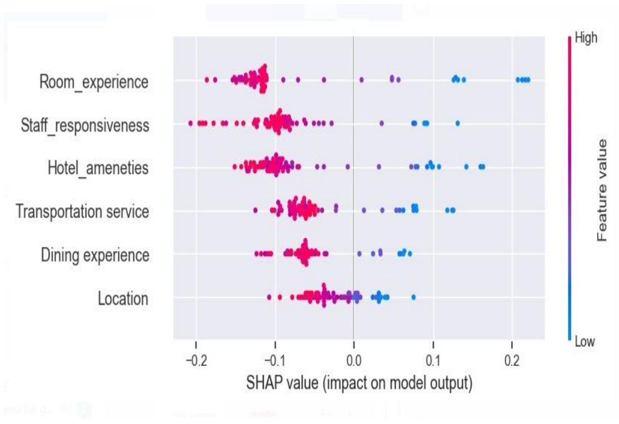
GOA

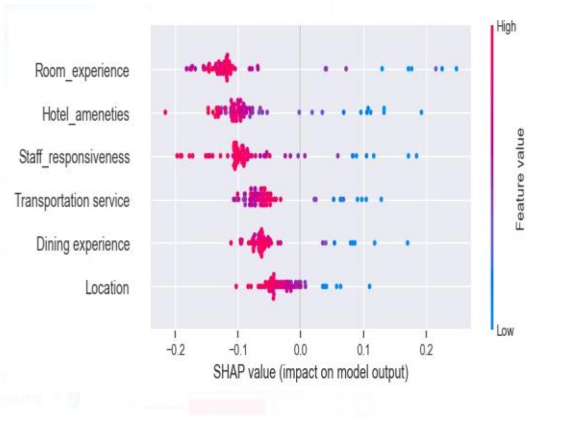


RAJASTHAN



SIKKIM KARNATAKA

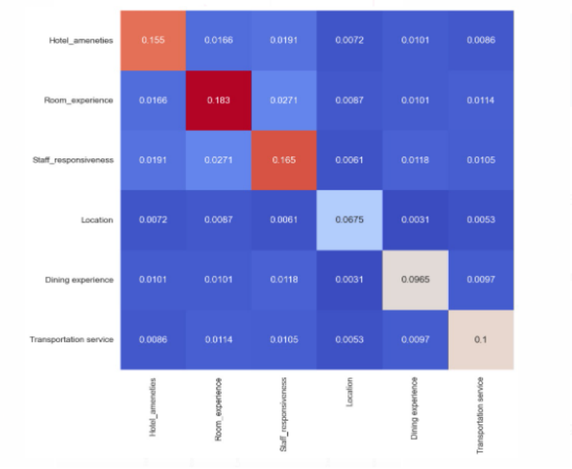


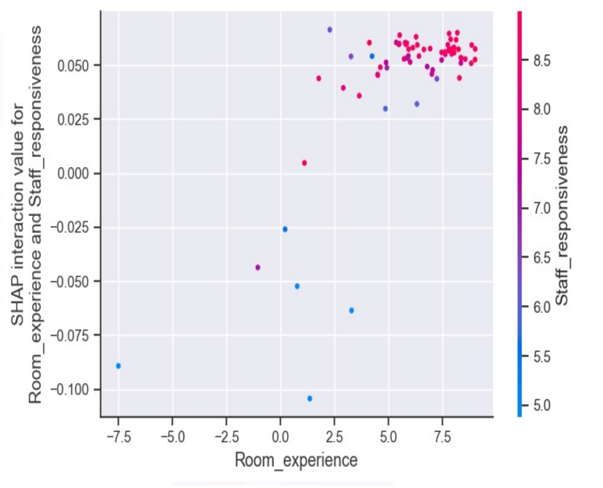


In the X-axis, we are plotting the model output(log-odds) with respect to the reviews being bad. Positive values on the X-axis means higher probability of the reviews being bad and Negative values on the X-axis means lower probability of the reviews being bad. From the above plots we can conclude the following:

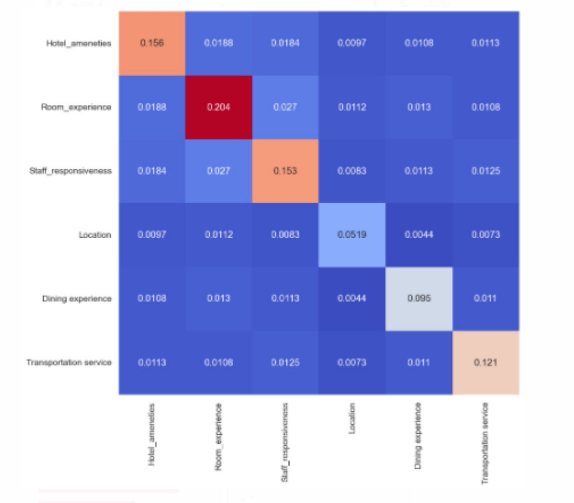
- In the case of state of Goa, we can see that in most of the reviews the overall scores of all the service parameters are on the lower side, these service parameters altogether are increasing the probability of the reviews being bad.
- However, in the case of other states, we can see, in most of the reviews, the scores of the service parameters are on the higher side and higher scores are decreasing the probability of the review being bad.
- In case of Himachal Pradesh, the service parameter “Staff Responsiveness” in some cases perform exceptionally well in lowering the probability of the review being bad, which is evident from the fact that it has brought the value of the log-odds drastically down to below 0.2. From this we can understand that if we can provide better staff responsiveness to the customers, this feature can exceed the room experience and hotel amenities in terms of importance.
- In case of West Bengal, we observe that the scores of the “Transportation Service” is on the lower side and should be worked upon to increase the same.

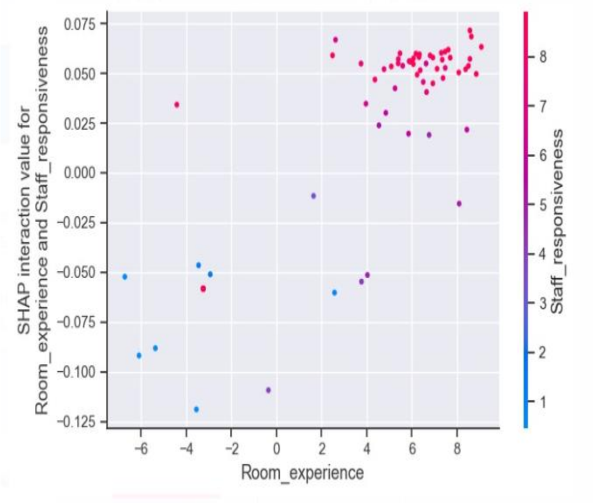
## WEST BENGAL



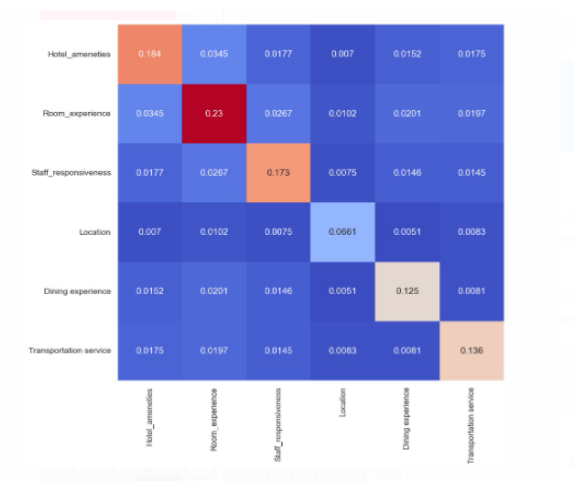


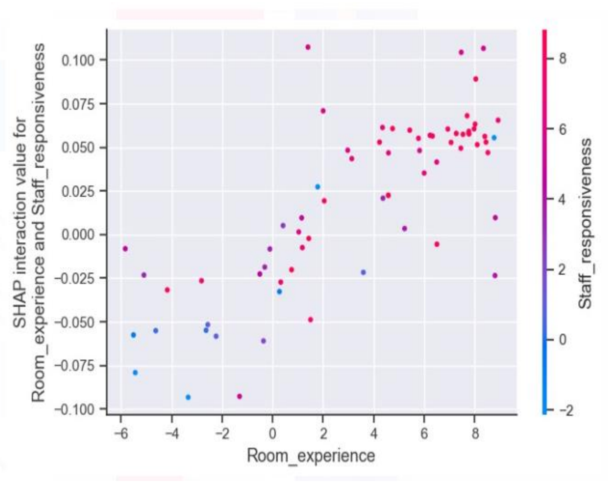
## HIMACHAL PRADESH



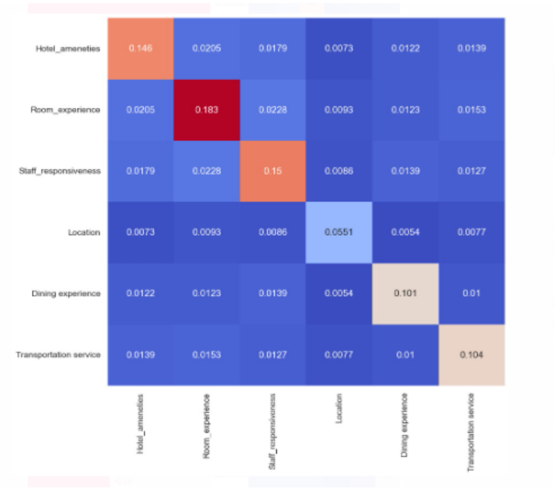


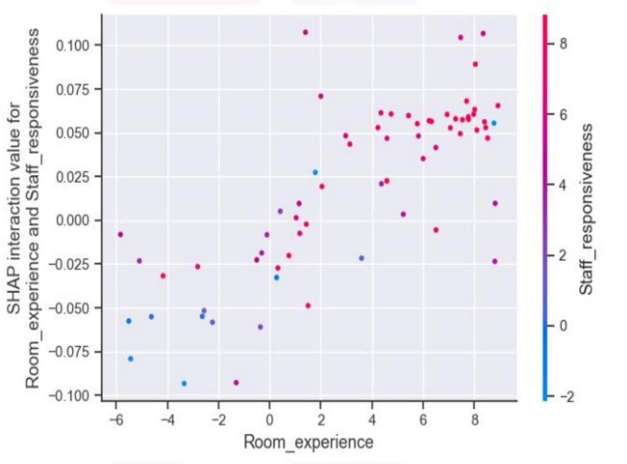
## GOA



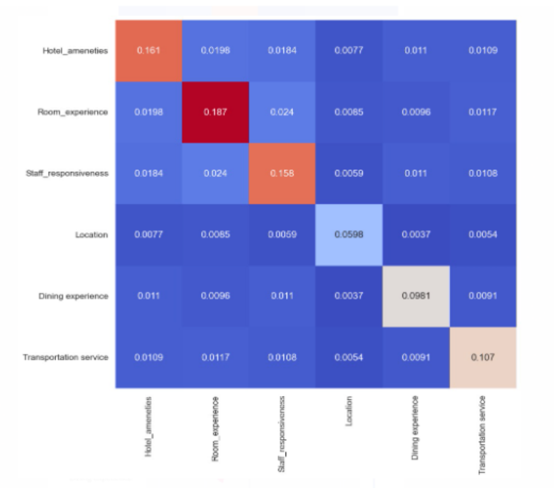


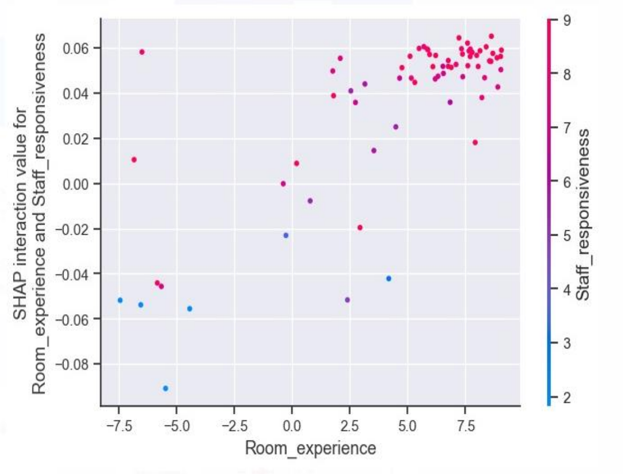
RAJASTHAN



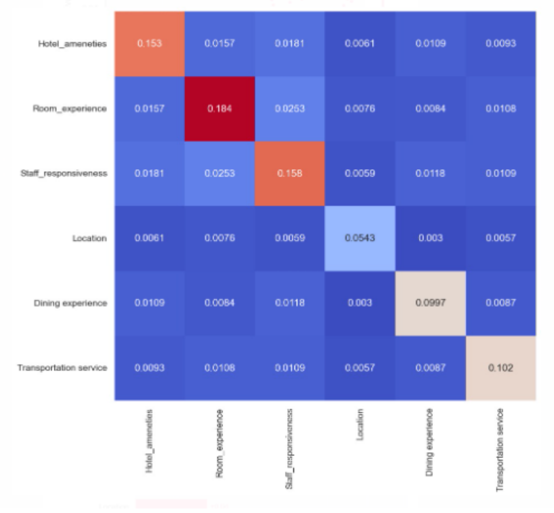


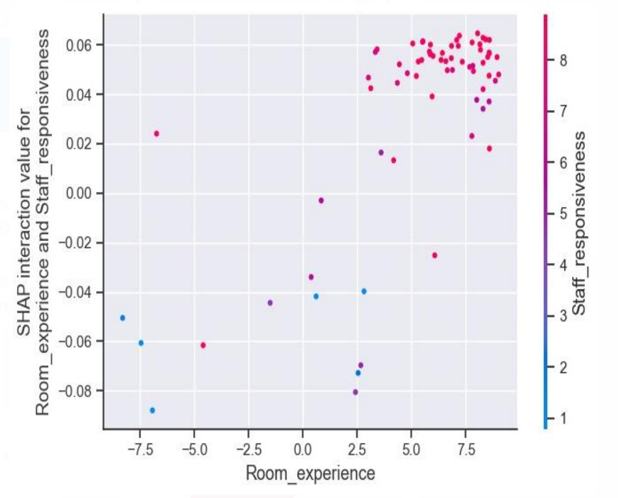
SIKKIM





KARNATAKA

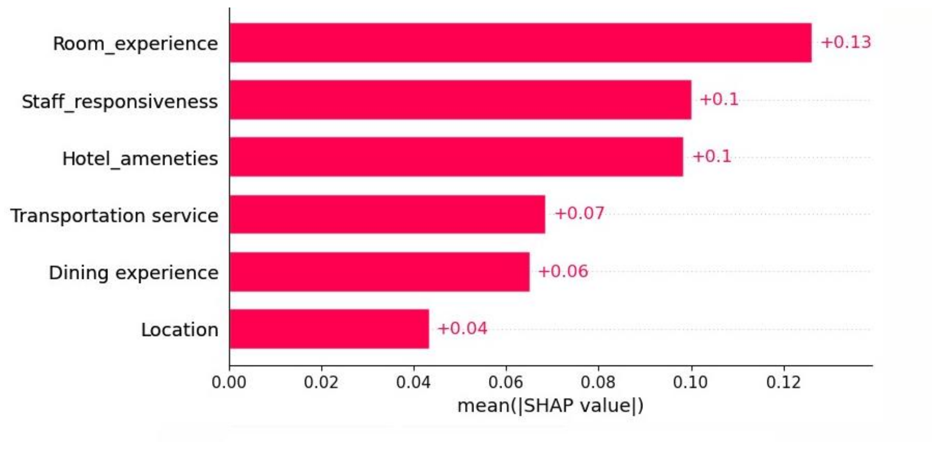




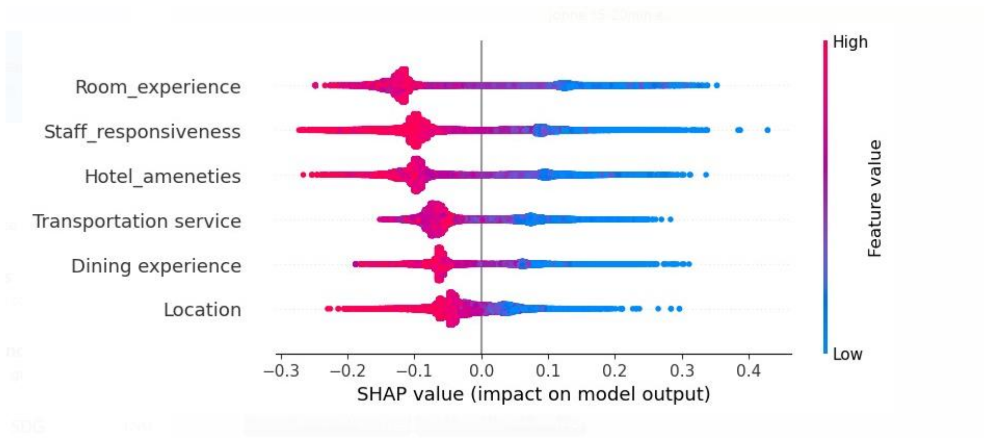
In the X-axis we have the actual feature scores of one of the interacting variable, while, the colours are representing the feature scores of the other interacting variable. The Y-axis represents the Shap interaction values(log-odds), positive values of the log-odds mean higher probability of the review being bad and vice-versa. We conclude the following from the above plots:

- From the above dependence plots, we observe that in case of Goa we can see that although the scores of room experience and hotel amenities are not good, their interaction effect is not affecting the reviews badly. So, we can conclude that for the tourists visiting Goa the overall combination of room experience and hotel amenities is not that important.
- While in case of the other states, we can see that although the scores of staff responsiveness and room experience are on the higher side but still their combined effect is slightly increasing the probability of the reviews being bad. Although this is counter intuitive, but this is happening may be due to higher prices of the rooms. Since higher prices comes with higher expectations from the customer.

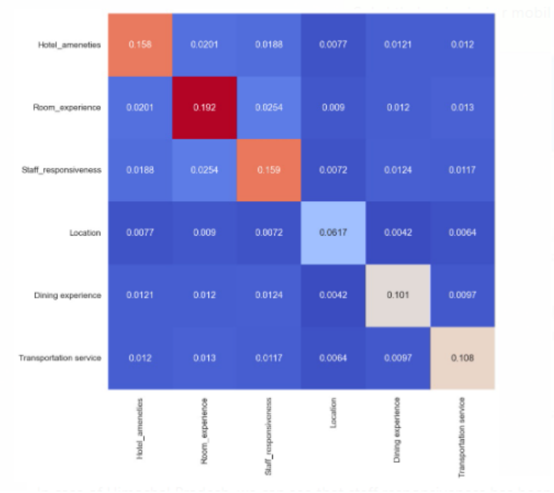
## ANALYSIS on ALL THE REVIEWS COLLECTED



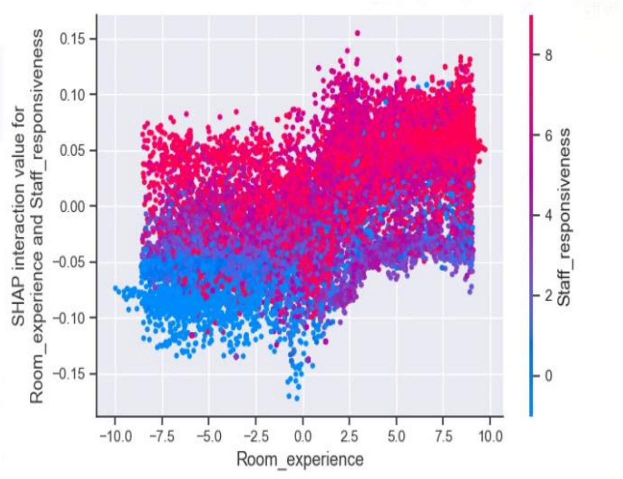
In the above bar plot we observe that “Room_experience” proves to be the most important service parameter for all the reviews which is in the same line as that of our state wise reviews analysis.



In the above beeswarm plot what we observe that in some cases lower scores of “Staff_responsiveness” is pushing up the value of log odds beyond 0.3 and even crossing 0.4. So, we can say that for some customers Staff_responsiveness is playing a crucial role in determining the quality of their experience.



| | Hotel_ameneties | Room_experience | Staff_responsiveness | Location | Dining experience | Transportation service |
|---|---|---|---|---|---|---|
| Hotel_ameneties | 0.158 | 0.0201 | 0.0188 | 0.0077 | 0.0121 | 0.012 |
| Room_experience | 0.0201 | 0.192 | 0.0254 | 0.009 | 0.012 | 0.013 |
| Staff_responsiveness | 0.0188 | 0.0254 | 0.159 | 0.0072 | 0.0124 | 0.0117 |
| Location | 0.0077 | 0.009 | 0.0072 | 0.0617 | 0.0042 | 0.0064 |
| Dining experience | 0.0121 | 0.012 | 0.0124 | 0.0042 | 0.101 | 0.0097 |
| Transportation service | 0.012 | 0.013 | 0.0117 | 0.0064 | 0.0097 | 0.108 |



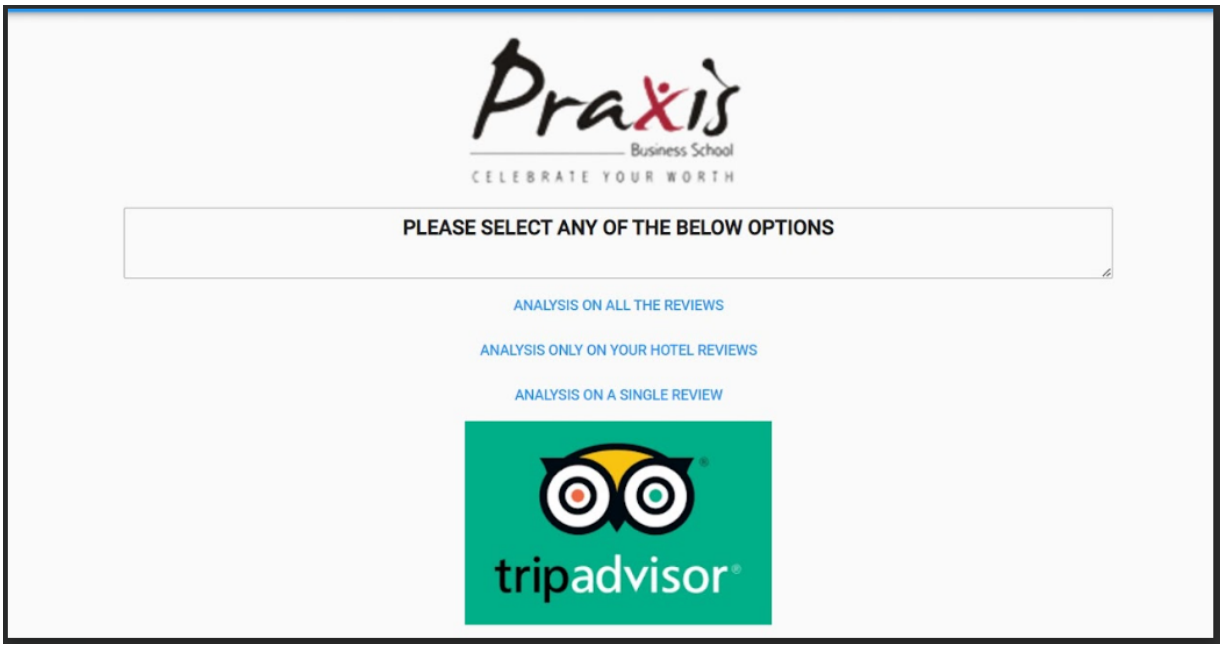
- From the dependence plot we can see that there are some datapoints for which the scores of both the interacting features are on the higher side but still they are marginally pushing up the values of the log odds or the probability of the reviews being bad. This might be due to the price of the rooms that is, they are not getting the service as expected. Since higher price comes with higher expectations.

- We also observe that datapoints have lower scores of room experience but higher value of staff responsiveness, for those customers the shap interaction value is mainly varying between 0 - 0.05 , so we can say that good staff experience has somehow overruled the effect of lower scores of room experience.

- Some datapoints are there where the scores of both the interacting features are on the lower side but still, they are not contributing much on increasing the probability of the reviews being bad. These are actually the value for money hotels where the expectations of the customers are on the lower side.

## DEPLOYMENT

We have used Anvil App editor for developing the User Interface. We have connected the front-end to the back-end using the Uplink which was generated by the Anvil App editor. The backend has been hosted in our local machine.



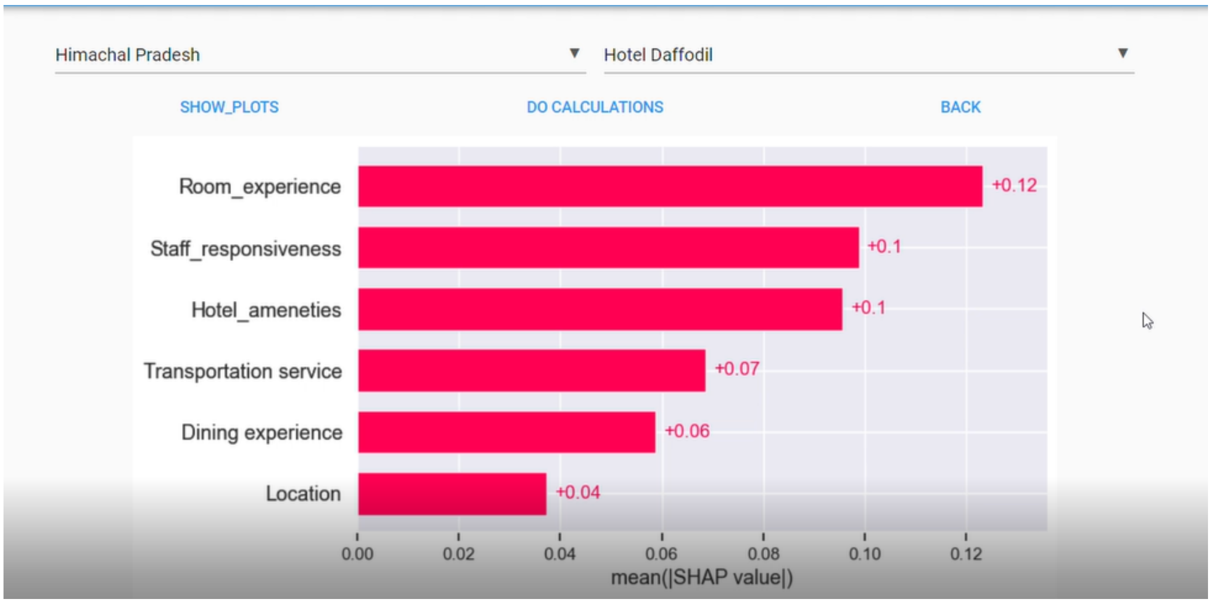
PLEASE SELECT ANY OF THE BELOW OPTIONS

ANALYSIS ON ALL THE REVIEWS

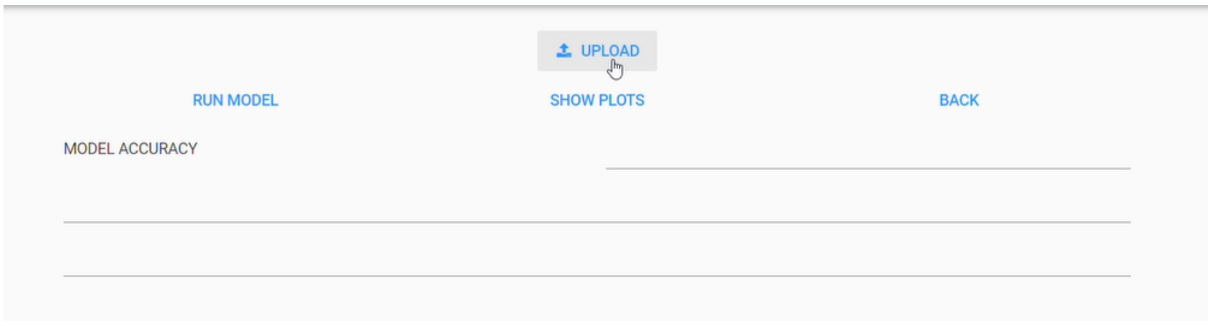
ANALYSIS ONLY ON YOUR HOTEL REVIEWS

ANALYSIS ON A SINGLE REVIEW

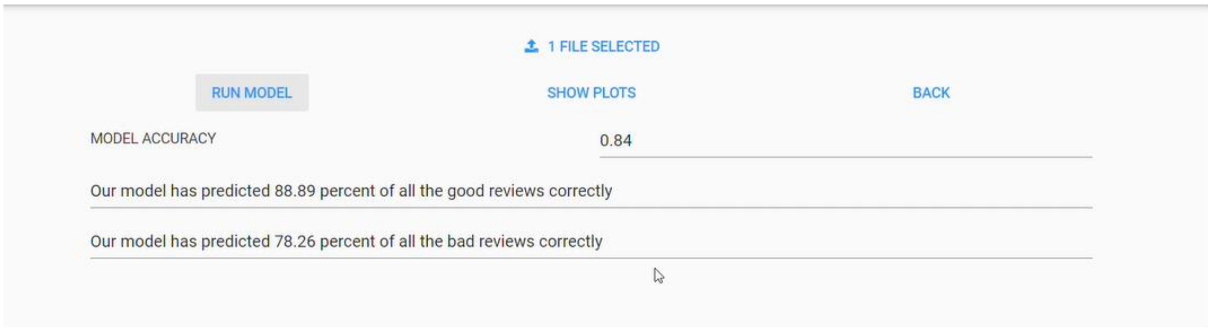
### STATE and HOTEL WISE REVIEWS



## ANALYSIS on UPLOADED REVIEWS



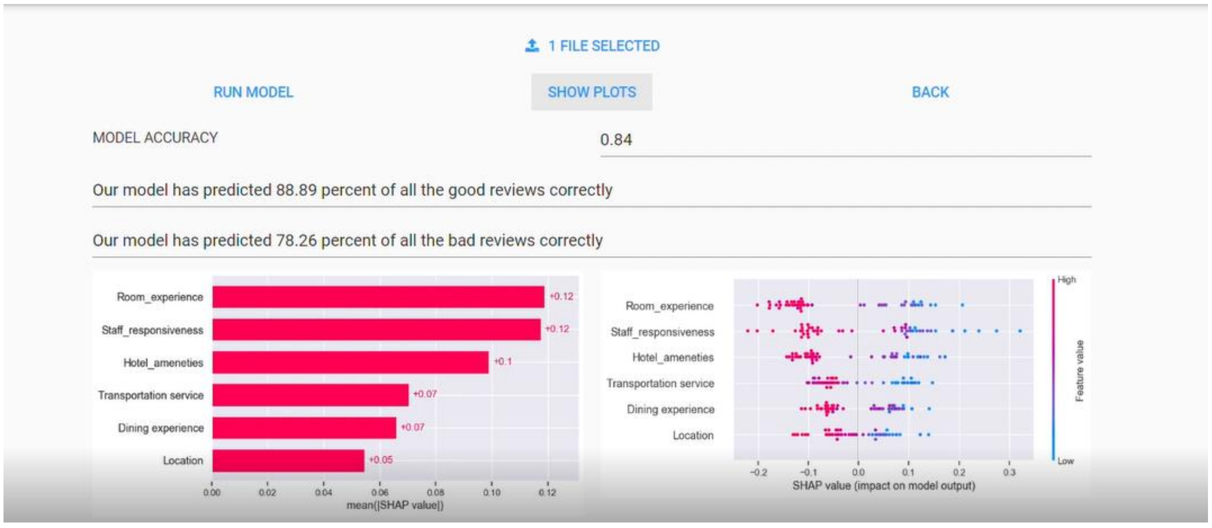
MODEL ACCURACY



MODEL ACCURACY 0.84

Our model has predicted 88.89 percent of all the good reviews correctly

Our model has predicted 78.26 percent of all the bad reviews correctly

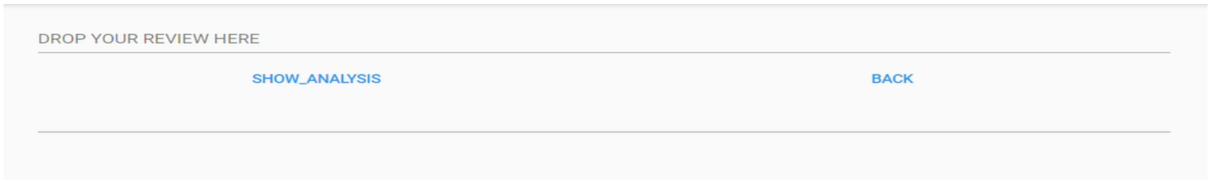


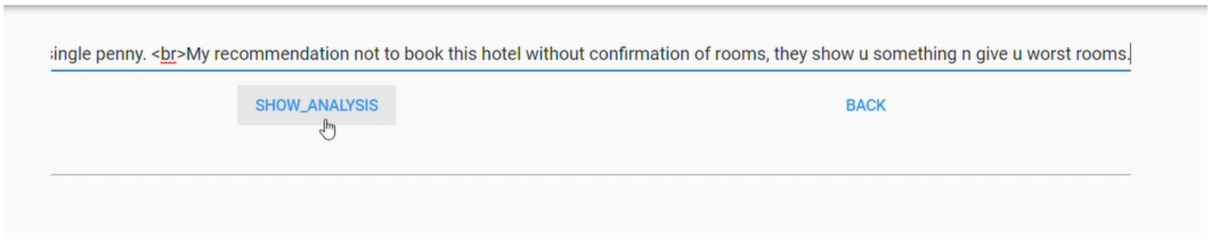
MODEL ACCURACY 0.84

Our model has predicted 88.89 percent of all the good reviews correctly

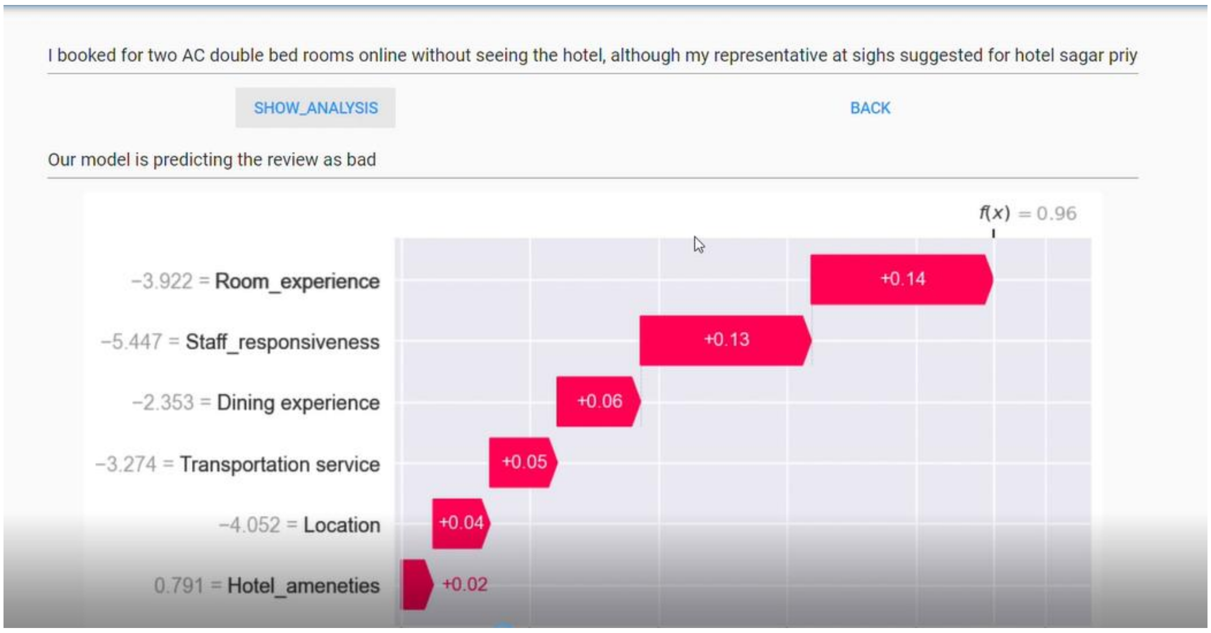
Our model has predicted 78.26 percent of all the bad reviews correctly

## ANALYSIS on a SINGLE REVIEW





ingle penny. <br>My recommendation not to book this hotel without confirmation of rooms, they show u something n give u worst rooms.



I booked for two AC double bed rooms online without seeing the hotel, although my representative at sighs suggested for hotel sagar priy

Our model is predicting the review as bad

## LIMITATIONS of the PROJECT

1. Users who will be using our web app should be good about analyzing the output of the SHAP graphs.

2. Our app needs heavy calculations to be done in the backend. So, our backend compute unit should have at least 8 GB RAM and as well as GPU support for better performance.

3. Initially, in order to focus only on the service parameters of hotels, we have not considered the price of the hotels as a feature. If we include price, we would be able to explain the sentiments of the customers connecting with the hotel industry service parameters using the SHAP graphs more accurately.

4. We have scrapped reviews only from TripAdvisor. Reviews from other service providers were not included.

5. We have considered around 10 hotels on average from 20 states in India. So, if we can include more hotels in our database then we can much better analysis.

## FUTURE SCOPE

1. We can include more reviews so that our outputs can generate better state wise representations of hotels.

2. We are planning to give our users an option using which they can get the outputs w.r.t the reviews being bad or good. Now we are providing all the analysis with respect to our reviews being bad.

3. We would also update our model and web app by adding the “Price” feature to analyze the SHAP graphs and give more accurate suggestions from both “Hotels” and the “Customers” point of view.

4. We are also planning to add a feature which can analyze our output graphs automatically and present the same in natural language.

## REFERRENCES

https://spacy.io/models/en#en_core_web_sm

https://towardsdatascience.com/bayesian-optimization-concept-explained-in-layman-terms-1d2bcdeaf12f

https://towardsdatascience.com/a-conceptual-explanation-of-bayesian-model-based-hyperparameter-optimization-for-machine-learning-b8172278050f

https://towardsdatascience.com/evaluate-topic-model-in-python-latent-dirichlet-allocation-lda-7d57484bb5d0#:~:text=Topic%20Coherence%20measures%20score%20a,are%20artifacts%20of%20statistical%20inference

https://medium.com/dataseries/understanding-pointwise-mutual-information-in-nlp-e4ef75ecb57a

https://stats.stackexchange.com/questions/140935/how-does-the-logpx-y-normalize-the-point-wise-mutual-information

https://www.engati.com/glossary/cosine-similarity#:~:text=In%20NLP%2C%20Cosine%20similarity%20is,in%20a%20multi%2Ddimensional%20space

https://www.analyticsvidhya.com/blog/2021/06/part-2-topic-modeling-and-latent-dirichlet-allocation-lda-using-gensim-and-sklearn/#:~:text=Latent%20Dirichlet%20Allocation%20(LDA)%20is,are%20also%20%E2%80%9Chidden%20topics%E2%80%9D

https://www.analyticsvidhya.com/blog/2021/12/fine-tune-bert-model-for-sentiment-analysis-in-google-colab/#:~:text=Introduction%20to%20BERT%20Model%20for,negative%2C%20or%20neutral%20about%20it

https://towardsdatascience.com/sentiment-analysis-in-10-minutes-with-bert-and-hugging-face-294e8a04b671

https://towardsdatascience.com/shap-explained-the-way-i-wish-someone-explained-it-to-me-ab81cc69ef30