REPORT ON HOTEL REVIEWS

We have to download and import some libraries for the feature extraction process. The libraries are shown in the figure below,

```
import numpy as np
import pandas as pd
import re
import nltk
from nltk.tokenize import word_tokenize,sent_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer,LancasterStemmer,SnowballStemmer,WordNetLemmatizer
import sklearn
from sklearn.feature_extraction.text import CountVectorizer
from autocorrect import Speller
import emoji
```

After completing the importing of libraries, we have to import the datasetin our IDE.

We have to use command “pd.read_json(r”file path”)”.

Converting json file to csv and saved it

```
data2 = data1.to_csv(r'C:\\Users\srava\Downloads.csv')
```

```
data_c = pd.read_csv(r"C:\Users\srava\Downloads\hotel_reviews (2).csv")
```

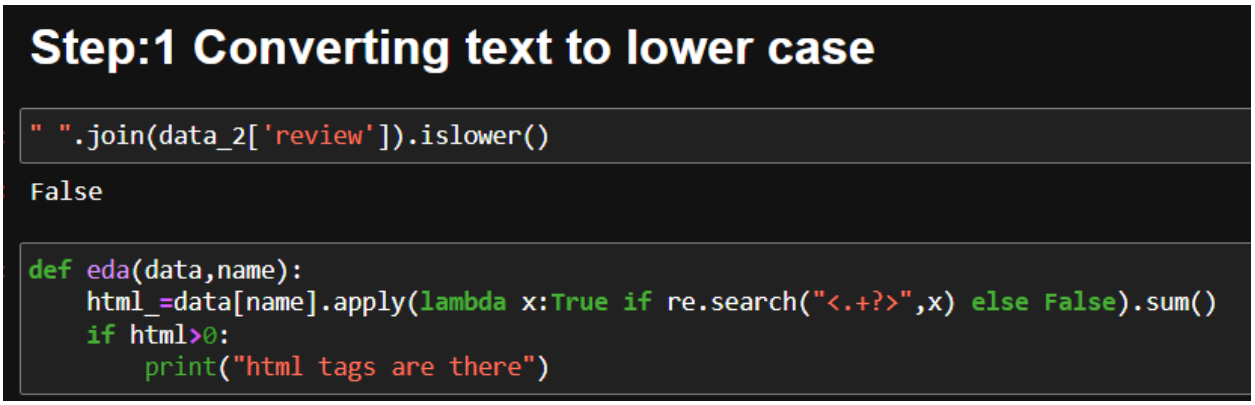
```
data_c.head()
```

	Unnamed: 0	review	sentiment
0	0	The walls were thin and could hear noise from ...	positive
1	1	The walls were thin and could hear noise from ...	negative
2	2	The food was overpriced and not very tasty. 😬	neutral
3	3	An average stay, nothing special but no major ...	positive
4	4	Loved the spa and pool area! A truly relaxing ...	negative

To convert the json file to csv, we use the keyword called “to_csv”.

Coying the duplicate of original dataset.

Step:1 Converting text to lowercase



Step:1 Converting text to lower case

```
" ".join(data_2['review']).islower()
```

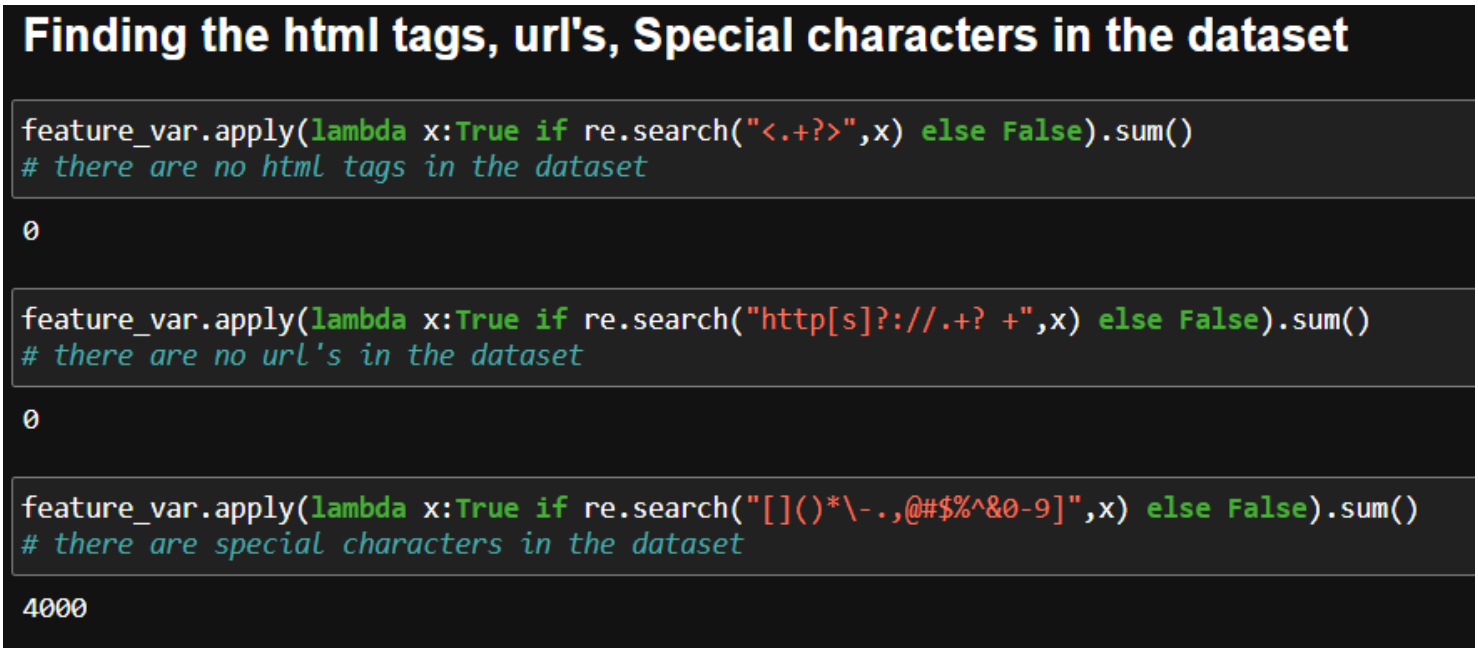
```
False
```

```
def eda(data,name):
    html_=data[name].apply(lambda x:True if re.search("<.+?>",x) else False).sum()
    if html>0:
        print("html tags are there")
```

Converting the text to lower case because python is case sensitive.

Step 2: Finding the html tags, url's, Special characters

By performing EDA we can clean the data.



Finding the html tags, url's, Special characters in the dataset

```
feature_var.apply(lambda x:True if re.search("<.+?>",x) else False).sum()
# there are no html tags in the dataset
```

```
0
```

```
feature_var.apply(lambda x:True if re.search("http[s]?://.+? +",x) else False).sum()
# there are no url's in the dataset
```

```
0
```

```
feature_var.apply(lambda x:True if re.search("[]()*\-.,@#$%^&0-9]",x) else False).sum()
# there are special characters in the dataset
```

```
4000
```

- There are no html tags in the dataset.
- There are no url’s in the dataset.
- There are special characters in each data point in the dataset.

Stem 3: Text pre-processing.

- We have to write a code to remove unwanted items in the dataset after the execution of the code we will get the cleaned data with no unwanted items.

```
def txtpp(y,correct,emojii):
    from textblob import TextBlob
    spell=Speller(lang="en")
    y=y.lower()
    y=re.sub("[]:()*\-.,@#$%^&0-9]"," ",y)
    y=re.sub("\s+"," ",y)
    if correct=="t":
        y=TextBlob(y).correct().string
    else:
        y=spell(y)
    return y
```

Step 4: Removing Stop Words and Perform Stemming.

For finding the stop words in the entire English language we can write code as,

```
stop = stopwords.words('english')
print(stop)
```

After this, we have to write a for loop to iterate through the dataset and remove the stop words,

```
ps = PorterStemmer()
ls = LancasterStemmer()
ss = SnowballStemmer(language='english')
```

```
def advpp(x,stemm):
    l=[]
    for word in word_tokenize(x):
        if word in stop:
            pass
        else:
            if stemm == 'p':
                l.append(ps.stem(word))
            elif stemm == 'l':
                l.append(ps.stem(word))
            elif stemm == 's':
                l.append(ps.stem(word))
            else:
                l.append(word)
    return " ".join(l)
```

- After looping the stop words are removed in the dataset.
- Perform the stemming with different methods like porterstemmer, lancasterstemmer, snowballstemmer.
- After this the loop is performed and stemming is also done.

```
feature_var['review']=  feature_var['review'].apply(advpp)
```

```
feature_var['review']
```

```
0       wall thin could hear nois room good stay smili...
1       wall thin could hear nois room good stay disap...
2                           food overpr tasti neutral fac
3       averag stay noth special major problem smiling...
4       love spa pool area truli relax experi disappoi...
                              ...
3995    averag stay noth special major problem neutral...
3996    clean room excel locat highli recommend smilin...
3997     hotel okay great terribl either disappointed fac
3998    unfriendli staff breakfast disappoint neutral fac
3999    clean room excel locat highli recommend smilin...
Name: review, Length: 4000, dtype: object
```

Step 5: Emoji prediction and Converting Emojis into words.

```
def convert_emojis_to_text(text):
    return emoji.demojize(text)
feature_var['review']=feature_var['review'].apply(convert_emojis_to_text)
```

- The above code is to convert the emoji into the text.
- After the execution of above code, the emojis are converted to text format.
- So that we won’t lose the information.

Step 6: Converting the processed data to vector format.

- To convert the processed data into the vector format we need to use the key word called countvectorizer and fit_transform.
- The Dataset is converted to vector format that is stored in a variable called “final_cv”.

```
cv=CountVectorizer(ngram_range=(1,1))
```

```
final_cv=cv.fit_transform(feature_var['review']).toarray()
```

```
final_cv
```

```
array([[0, 0, 0, ..., 1, 1, 0],
       [0, 0, 0, ..., 1, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 1, 0]], dtype=int64)
```