NLP - Hotel review sentiment analysis in python

```
#warnings :)
import warnings
warnings.filterwarnings('ignore')

import os
dir_Path = 'F:\\'
os.chdir(dir_Path)
```

Data Facts and Import

```
import pandas as pd
# Local directory
Reviewdata = pd.read_csv('train.csv')
#Data Credit - https://www.kaggle.com/anu0012/hotel-review/data
```

```
Reviewdata.shape
```

```
(38932, 5)
```

```
Reviewdata.head()
```

User_ID	Description	Browser_Used	Device_Used	Is_Response

```
Reviewdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38932 entries, 0 to 38931
Data columns (total 5 columns):
User_ID         38932 non-null object
Description     38932 non-null object
Browser_Used    38932 non-null object
Device_Used     38932 non-null object
Is_Response     38932 non-null object
dtypes: object(5)
memory usage: 1.5+ MB
```

```
Reviewdata.describe().transpose()
```

	count	unique	top	freq
User_ID	38932	38932	id41139	1
Description	38932	38932	I am a Hilton Diamond member and have stayed a...	1
Browser_Used	38932	11	Firefox	7367
Device_Used	38932	3	Desktop	15026
Is_Response	38932	2	happy	26521

Data Cleaning / EDA

```
### Checking Missing values in the Data Set and printing the Percentage for Missing Values for Each Columns ###

count = Reviewdata.isnull().sum().sort_values(ascending=False)
percentage = ((Reviewdata.isnull().sum()/len(Reviewdata)*100)).sort_values(ascending=False)
missing_data = pd.concat([count, percentage], axis=1,
keys=['Count','Percentage'])
```

```
print('Count and percentage of missing values for the columns:')

missing_data
```

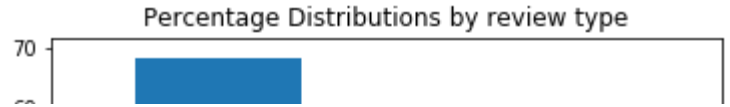
Count and percentage of missing values for the columns:

	Count	Percentage
Is_Response	0	0.0
Device_Used	0	0.0
Browser_Used	0	0.0
Description	0	0.0
User_ID	0	0.0

```
### Checking for the Distribution of Default ###
import matplotlib.pyplot as plt
%matplotlib inline
print('Percentage for default\n')
print(round(Reviewdata.Is_Response.value_counts(normalize=True)*100,2))
round(Reviewdata.Is_Response.value_counts(normalize=True)*100,2).plot(kind='bar')
plt.title('Percentage Distributions by review type')
plt.show()
```

```
Percentage for default

happy        68.12
not happy    31.88
Name: Is_Response, dtype: float64
```




```
#Removing columns
Reviewdata.drop(columns = ['User_ID', 'Browser_Used', 'Device_Used'], inplace = True)
```


```
# Apply first level cleaning
import re
import string

#This function converts to lower-case, removes square bracket, removes numbers and punctuation
def text_clean_1(text):
    text = text.lower()
    text = re.sub('\[.*?\]', '', text) # square brackets
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text) # punctuation
    text = re.sub('\w*\d\w*', '', text) # numbers
    return text

cleaned1 = lambda x: text_clean_1(x)


# Let's take a look at the updated text
Reviewdata['cleaned_description'] = pd.DataFrame(Reviewdata.Description.apply(cleaned1))
Reviewdata.head(10)
```

	Description	Is_Response	cleaned_description
0	The room was kind of clean but had a VERY stro...	not happy	the room was kind of clean but had a very stro...
1	I stayed at the Crown Plaza April -- - April -...	not happy	i stayed at the crown plaza april april th...
2	I booked this hotel through Hotwire at the low...	not happy	i booked this hotel through hotwire at the low...
3	Stayed here with husband and sons on the way t...	happy	stayed here with husband and sons on the way t...
4	My girlfriends and I stayed here to celebrate ...	not happy	my girlfriends and i stayed here to celebrate ...

```
# Apply a second round of cleaning
def text_clean_2(text):
    text = re.sub('[‘’“”…]', '', text) # quotes
    text = re.sub('\n', '', text) # new line
    return text

cleaned2 = lambda x: text_clean_2(x)


# Let's take a look at the updated text
Reviewdata['cleaned_description_new'] = pd.DataFrame(Reviewdata['cleaned_description'].apply(cleaned2))
Reviewdata.head(10)
```

	Description	Is_Response	cleaned_description	cleaned_description_new
	-...	not happy	th...	th...

Model training

```
from sklearn.model_selection import train_test_split

Independent_var = Reviewdata.cleaned_description_new
Dependent_var = Reviewdata.Is_Response

IV_train, IV_test, DV_train, DV_test = train_test_split(Independent_var, Dependent_var, test_size = 0.1, random_state = 225)

print('IV_train :', len(IV_train))
print('IV_test  :', len(IV_test))
print('DV_train :', len(DV_train))
print('DV_test  :', len(DV_test))
```

```
IV_train : 35038
IV_test  : 3894
DV_train : 35038
DV_test  : 3894
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression

tvec = TfidfVectorizer()
clf2 = LogisticRegression(solver = "lbfgs")
```

```
from sklearn.pipeline import Pipeline
```

```
model = Pipeline([('vectorizer',tvec),('classifier',clf2)])

model.fit(IV_train, DV_train)
```

```
from sklearn.metrics import confusion_matrix

predictions = model.predict(IV_test)

confusion_matrix(predictions, DV_test)
```

```
array([[2418,  306],
       [ 153, 1017]], dtype=int64)
```

Model prediciton

```
from sklearn.metrics import accuracy_score, precision_score, recall_score

print("Accuracy : ", accuracy_score(predictions, DV_test))
print("Precision : ", precision_score(predictions, DV_test, average = 'weighted'))
print("Recall : ", recall_score(predictions, DV_test, average = 'weighted'))
```

```
Accuracy :  0.8821263482280431
Precision :  0.8888758956340842
Recall :  0.8821263482280431
```

Trying on new reviews

```
example = ["I'm not happy"]
result = model.predict(example)

print(result)
```

```
['happy']
```