exploratory_analysis

October 3, 2022

1 EXPLORATORY ANALYSIS

1.1 TRIPADVISOR: HOTELS

- Esteban Ariza
- Johan Giraldo
- Mateo Valdes

1.2 Prerequisites

1.2.1 Install python libraries

[ ]:
```
# !pip install pandas
# !pip install seaborn
# !pip install numpy
# !pip install regex
# !pip install matplotlib
# !pip install scipy
# !pip install PyEnchant
```

1.2.2 Install exporter

[ ]:
```
# !pip install nbconvert[webpdf]
# !sudo apt-get install texlive-xetex texlive-fonts-recommended␣
 ↪texlive-plain-generic
!export PATH=/Library/TeX/texbin:$PATH
```

1.2.3 Import python libraries

[ ]:
```
import pandas as pd
import seaborn as sns
import numpy as np
import re
```

```
import matplotlib.pyplot as plt
from scipy import stats
from google.colab import drive
import datetime
```

1.2.4 Import data (CSV)

[ ]:
```
drive.mount('/content/drive')
HOTEL_DATA_PATH = '/content/drive/MyDrive/Estudio/9 Semestre/Inteligencia␣
↪Artificial II/TripAdvisor/data.csv'
HOTEL_DATA = pd.read_csv(HOTEL_DATA_PATH)
```

1.3 Data Review

[ ]:
```
HOTEL_DATA.head(10)
```

[ ]:
```
HOTEL_DATA.tail(10)
```

[ ]:
```
HOTEL_DATA.info()
```

The csv have 8 columns and non of them contain null values. There are some adjustments we should do to each column: * **HOTEL_NAME:** The first number should be eliminated from the name. * **HOTEL_RATING:** The rating must be a integer number from 1 to 5. * **HOTEL_PRICE:** The currency must be removed and the price should be numeric. * **REVIEW_RATING:** The same as the "HOTEL_RATING" * **REVIEW_DATE:** The phrase "Date of stay:" should be removed.

[ ]:
```
GENERAL_HOTEL_DATA = HOTEL_DATA.groupby(["HOTEL_NAME",␣
↪"HOTEL_RATING","HOTEL_PRICE","HOTEL_LOCATION","HOTEL_REVIEW_LINK"]).size().
↪reset_index(name='HOTEL_REVIEW_COUNT') # COUNT
GENERAL_HOTEL_DATA.head(30)
```

1.4 Data Cleaning

1.4.1 Common use cleaning functions

[ ]:
```
def clean_int(column, pattern):
    return column.replace(pattern, '', regex = True).astype(np.int64)

def clean_string(column, pattern):
    return column.replace(pattern, '', regex = True)

HOTEL_DATA = HOTEL_DATA.dropna()
```

1.4.2 HOTEL_NAME

[ ]:
```
def clean_hotel_name(data):
  return data['HOTEL_NAME'].str.split('.').str[1].str[1:]

HOTEL_DATA['HOTEL_NAME'] = clean_hotel_name(HOTEL_DATA)
```

1.4.3 HOTEL_RATING

[ ]:
```
def clean_hotel_rating(data):
  return data['HOTEL_RATING'].str[0].astype(np.int64)

HOTEL_DATA['HOTEL_RATING'] = clean_hotel_rating(HOTEL_DATA)
```

1.4.4 HOTEL_PRICE

[ ]:
```
def clean_hotel_price(data): # 4 --> COP
  column = data['HOTEL_PRICE'].str.split('COP').str[1]
  column = column.str.replace(',','').astype(np.int64)
  return column

HOTEL_DATA['HOTEL_PRICE'] = clean_hotel_price(HOTEL_DATA)
```

1.4.5 REVIEW_RATING

[ ]:
```
def clean_review_rating(data):
  return (clean_int(data['REVIEW_RATING'],'[bubble_.]') / 10).astype(np.int64)

HOTEL_DATA['REVIEW_RATING'] =  clean_review_rating(HOTEL_DATA)
```

1.4.6 REVIEW_DATE

[ ]:
```
def parseDate(strDate):
  strMonth, strYear = strDate.split(' ')
  strYear = int(strYear)
  strMonth = datetime.datetime.strptime(strMonth[0:3],'%b').month
  date = pd.Timestamp(year=strYear, month=strMonth, day=1)
  return date

def clean_review_date(data):
  column = data['REVIEW_DATE'].str[14:]
  column = column.map(parseDate)
```

```
  return column

HOTEL_DATA['REVIEW_DATE'] = clean_review_date(HOTEL_DATA)
```

1.5 Data Analysis

1.5.1 GENERAL

[ ]:
```
HOTEL_DATA.info()
```

[ ]:
```
HOTEL_DATA.describe()
```

1.5.2 REVIEWS

[ ]:
```
# DF - REVIEW LENGTH (Add new col REVIEW TEXT LENGTH)
REVIEW_LENGTH_HOTEL_DATA = HOTEL_DATA.copy()
REVIEW_LENGTH_HOTEL_DATA["REVIEW_TEXT_LENGTH"] =␣
↪REVIEW_LENGTH_HOTEL_DATA["REVIEW_TEXT"].map(len)
```

REVIEW_RATING

[ ]:
```
REVIEW_LENGTH_HOTEL_DATA.sort_values(by = ['REVIEW_RATING'], ascending = False)
```

REVIEW_TEXT

[ ]:
```
REVIEW_LENGTH_HOTEL_DATA.sort_values(by = ['REVIEW_TEXT_LENGTH'], ascending =␣
↪False)
```

REVIEW DATE

[ ]:
```
REVIEW_LENGTH_HOTEL_DATA.sort_values(by = ['REVIEW_DATE'], ascending = False)
```

1.5.3 HOTELS

[ ]:
```
# DF - GENERAL HOTEL DATA (The hotel cols)
GENERAL_HOTEL_DATA = HOTEL_DATA.groupby(["HOTEL_NAME",␣
↪"HOTEL_RATING","HOTEL_PRICE","HOTEL_LOCATION","HOTEL_REVIEW_LINK"]).size().
↪reset_index(name='HOTEL_REVIEW_COUNT') # COUNT
GENERAL_HOTEL_DATA.info()
```

HOTEL_RATING

[ ]:
```
GENERAL_HOTEL_DATA.sort_values(by = ['HOTEL_RATING'], ascending = True)
```

HOTEL_PRICE

[ ]:
```
GENERAL_HOTEL_DATA.sort_values(by = ['HOTEL_PRICE'], ascending = False)
```

HOTEL_REVIEW_COUNTS

[ ]:
```
GENERAL_HOTEL_DATA.sort_values(by = ['HOTEL_REVIEW_COUNT'], ascending = False).
↪head()
```

1.6 Visualization

1.6.1 HOTELS PRICE (COP)

[ ]:
```
sns.set(rc={'figure.figsize':(12,8)})
sns.barplot(x="HOTEL_PRICE", y="HOTEL_NAME", data=GENERAL_HOTEL_DATA.
↪sort_values(by=['HOTEL_NAME']), capsize=.2, palette='flare')

plt.tight_layout(h_pad=2)
```

1.6.2 HOTELS NUMBER OF REVIEWS

[ ]:
```
sns.set(rc={'figure.figsize':(12,8)})
sns.barplot(x="HOTEL_REVIEW_COUNT", y="HOTEL_NAME", data=GENERAL_HOTEL_DATA.
↪sort_values(by=['HOTEL_NAME']), capsize=.2, palette='flare')

plt.tight_layout(h_pad=2)
```

1.6.3 REVIEW RATING OVER TIME

[ ]:
```
def date_to_float(date):
  return date.year+((date.month-1)/12)

# DF - OVERTIME (REVIEW_DATE as FLOAT)
OVERTIME_HOTEL_DATA = HOTEL_DATA.copy()
OVERTIME_HOTEL_DATA['REVIEW_DATE'] = OVERTIME_HOTEL_DATA['REVIEW_DATE'].
↪map(date_to_float)
```

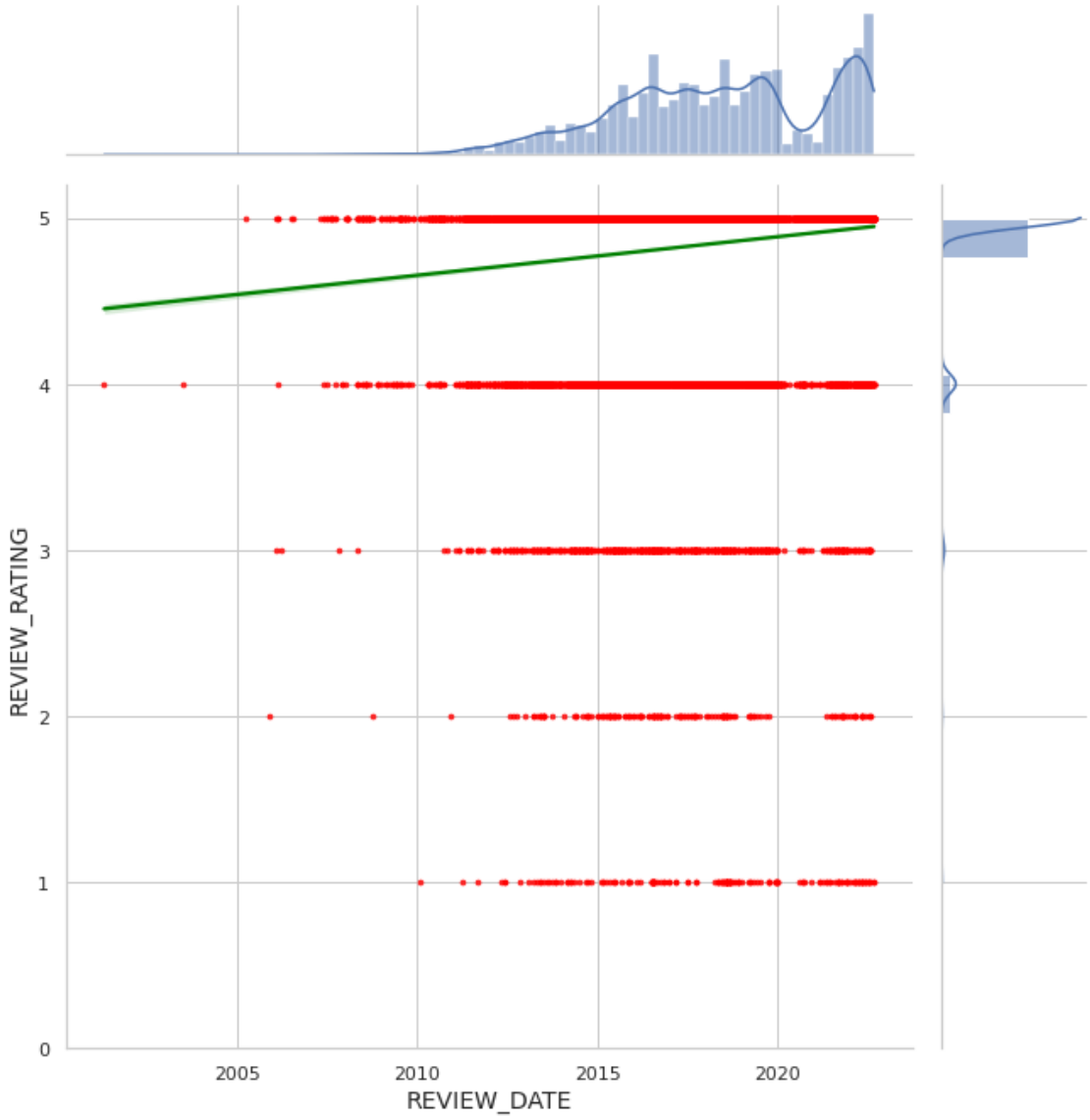
[1171]:
```
sns.set_theme(style="whitegrid")
```

```
plot = sns.jointplot(data = OVERTIME_HOTEL_DATA, x = 'REVIEW_DATE', y =␣
↪'REVIEW_RATING', color='b', kind = 'reg', line_kws={'color': 'green'},␣
↪scatter_kws={'s': 7, 'color':'red'})

plot.fig.set_size_inches(10,10)

plot.set_axis_labels('REVIEW_DATE', 'REVIEW_RATING', fontsize=14)
plot.ax_marg_y.set_ylim(bottom=0)

plt.show()
```



1.6.4 AVERAGE REVIEW RATING BY MONTH

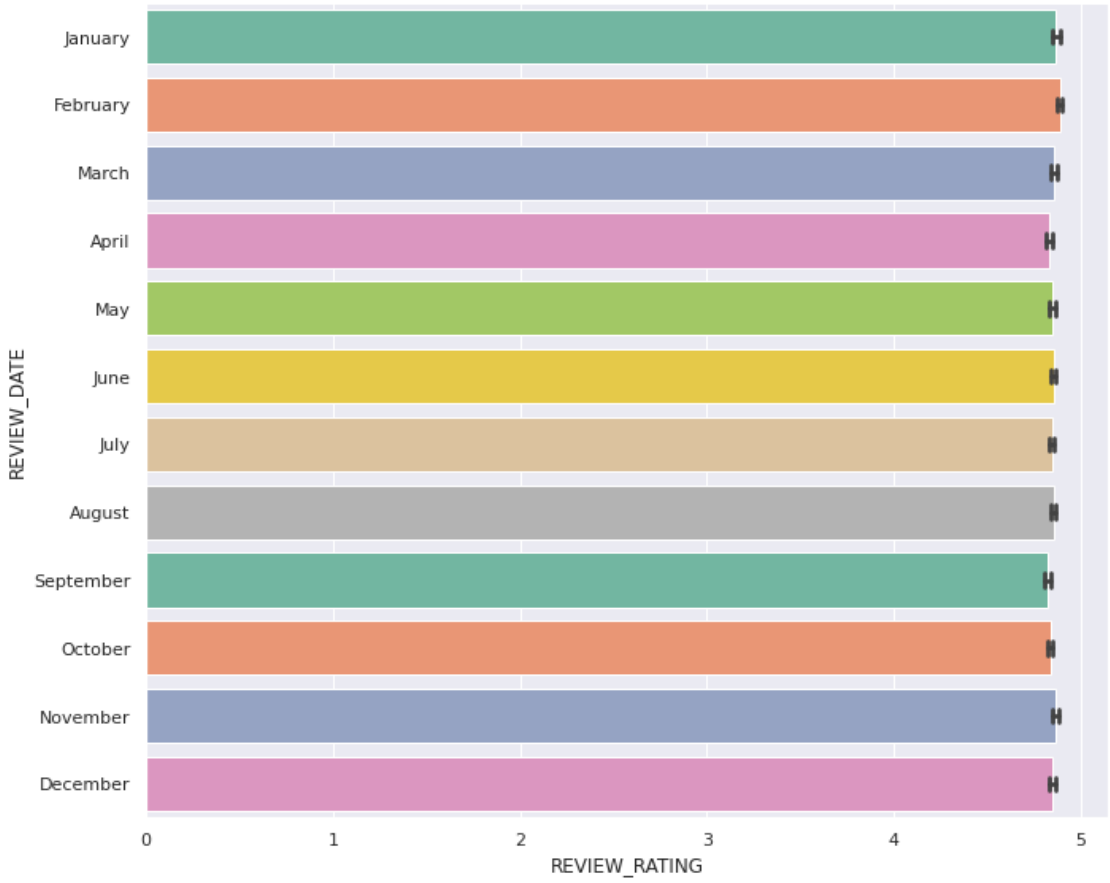
[1172]:
```
def date_to_month(date):
  return date.month

def month_to_name(month):
  return datetime.date(2022, month, 2).strftime('%B')

# DF - MONTHLY (REVIEW_DATE as FLOAT)
MONTHLY_HOTEL_DATA = HOTEL_DATA.copy()
MONTHLY_HOTEL_DATA['REVIEW_DATE'] = MONTHLY_HOTEL_DATA['REVIEW_DATE'].
 ↪map(date_to_month)
MONTHLY_HOTEL_DATA = MONTHLY_HOTEL_DATA.sort_values(by = ['REVIEW_DATE'])
MONTHLY_HOTEL_DATA['REVIEW_DATE'] = MONTHLY_HOTEL_DATA['REVIEW_DATE'].
 ↪map(month_to_name)
```

[1173]:
```
sns.set(rc={'figure.figsize':(10,8)})
sns.barplot(x="REVIEW_RATING", y="REVIEW_DATE", data=MONTHLY_HOTEL_DATA,␣
 ↪capsize=.2, palette='Set2')

plt.tight_layout(h_pad=2)
```



1.6.5 NUMBER OF REVIEWS OVER TIME

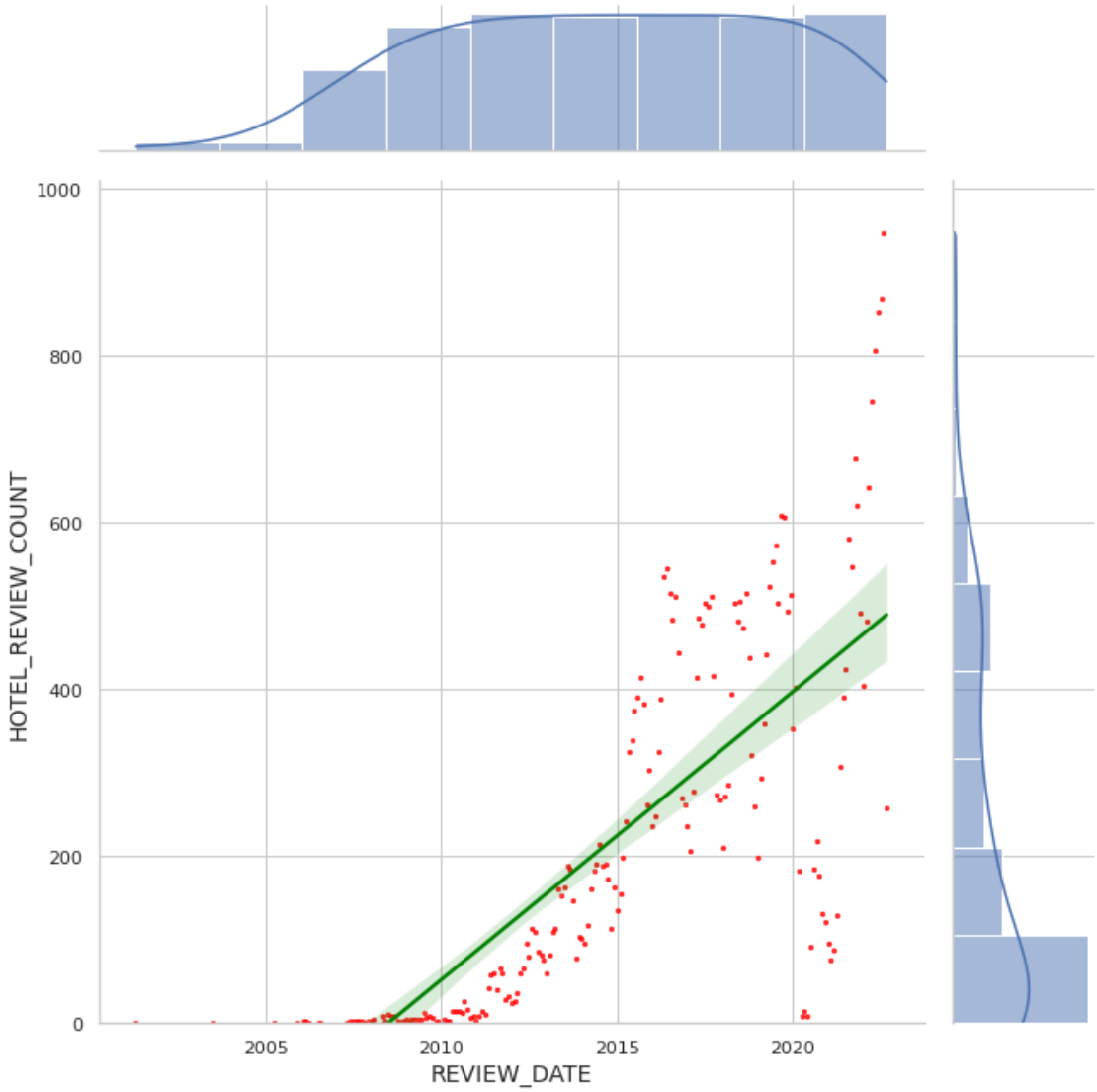
[1174]:
```
# DF - OVERTIME REVIEW COUNT (With columns: REVIEW_DATE as FLOAT,␣
↪HOTEL_REVIEW_COUNT)
REVIEW_HOTEL_DATA = OVERTIME_HOTEL_DATA.groupby(["REVIEW_DATE"]).size().
↪reset_index(name='HOTEL_REVIEW_COUNT')
```

[1175]:
```
sns.set_theme(style="whitegrid")
plot = sns.jointplot(data = REVIEW_HOTEL_DATA, x = 'REVIEW_DATE', y =␣
↪'HOTEL_REVIEW_COUNT', color='b', kind = 'reg', line_kws={'color': 'green'},␣
↪scatter_kws={'s': 5, 'color':'red'})

plot.fig.set_size_inches(10,10)

plot.set_axis_labels('REVIEW_DATE', 'HOTEL_REVIEW_COUNT', fontsize=14)
plot.ax_marg_y.set_ylim(bottom=0)

plt.show()
```



1.6.6 NUMBER OF REVIEWS BY MONTH

[ ]:
```
sns.set(rc={'figure.figsize':(10,8)})
sns.barplot(x="REVIEW_RATING", y="REVIEW_DATE", data=MONTHLY_HOTEL_DATA,␣
 ↪capsize=.2, estimator=sum, palette='Set2')

plt.tight_layout(h_pad=2)
```

1.6.7 HOTELS VARIABLES CORRELATION

[ ]:
```
sns.set_theme(style="white")
fig, axs = plt.subplots(figsize = (10,5))
plt.title('HOTELS VARIABLES CORRELATION')
axs.tick_params(axis = 'y', labelsize = 14, pad = 5)
axs.tick_params(axis = 'x', labelsize = 14, pad = 5)
sns.set(font_scale = 1.1)
cmap = sns.diverging_palette(250, 15, center="dark")

corr = GENERAL_HOTEL_DATA.corr()
mask = np.triu(np.ones_like(corr, dtype=bool))
sns.heatmap(corr, mask=mask, cmap=cmap, annot = True)

plt.show()
```

1.6.8 HOTEL PRICE DISTRIBUTION BY REVIEW RATING

[ ]:
```
# DF - RATING AS CATEGORY
RATING_AS_CATEGORY_HOTEL_DATA = HOTEL_DATA.copy()
RATING_AS_CATEGORY_HOTEL_DATA['REVIEW_RATING'] =␣
 ↪RATING_AS_CATEGORY_HOTEL_DATA['REVIEW_RATING'].astype("category")
```

[ ]:
```
plot = sns.catplot(data=RATING_AS_CATEGORY_HOTEL_DATA, x="HOTEL_PRICE",␣
 ↪y="REVIEW_RATING", kind="box", height=8, palette="Set1")
plt.text(x=0, y=-0.75, s='HOTEL PRICE DISTRIBUTION PER REVIEW RATING',␣
 ↪fontsize=16)
```

1.6.9 REVIEW LENGTH DISTRIBUTION BY REVIEW RATING

[ ]:
```
REVIEW_LENGTH_HOTEL_DATA['REVIEW_RATING'] =␣
 ↪REVIEW_LENGTH_HOTEL_DATA['REVIEW_RATING'].astype("category")
REVIEW_MAX_LENGTH = 2000

plot = sns.
 ↪catplot(data=REVIEW_LENGTH_HOTEL_DATA[REVIEW_LENGTH_HOTEL_DATA["REVIEW_TEXT_LENGTH"]<␣
 ↪REVIEW_MAX_LENGTH], x="REVIEW_TEXT_LENGTH", y="REVIEW_RATING", kind="box",␣
 ↪height=8, palette="Set1")
plt.text(x=0, y=-0.75, s='HOTEL PRICE DISTRIBUTION PER REVIEW RATING',␣
 ↪fontsize=16)
```

1.7 Export

1.7.1 PDF

[ ]:
```
IPYNB_PATH = "drive/MyDrive/Estudio/'9 Semestre'/'Inteligencia Artificial II'/
 ↪'TripAdvisor'/exploratory_analysis.ipynb"

# HTML
# !jupyter nbconvert $IPYNB_PATH --to html

# PDF
!jupyter nbconvert $IPYNB_PATH --to pdf
```

1.7.2 DATA

[ ]:
```

```