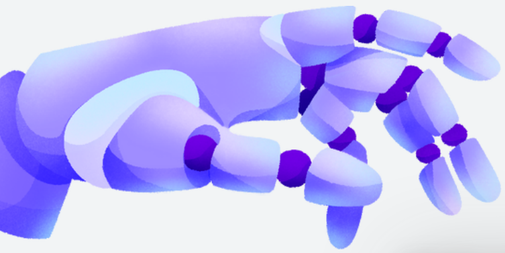
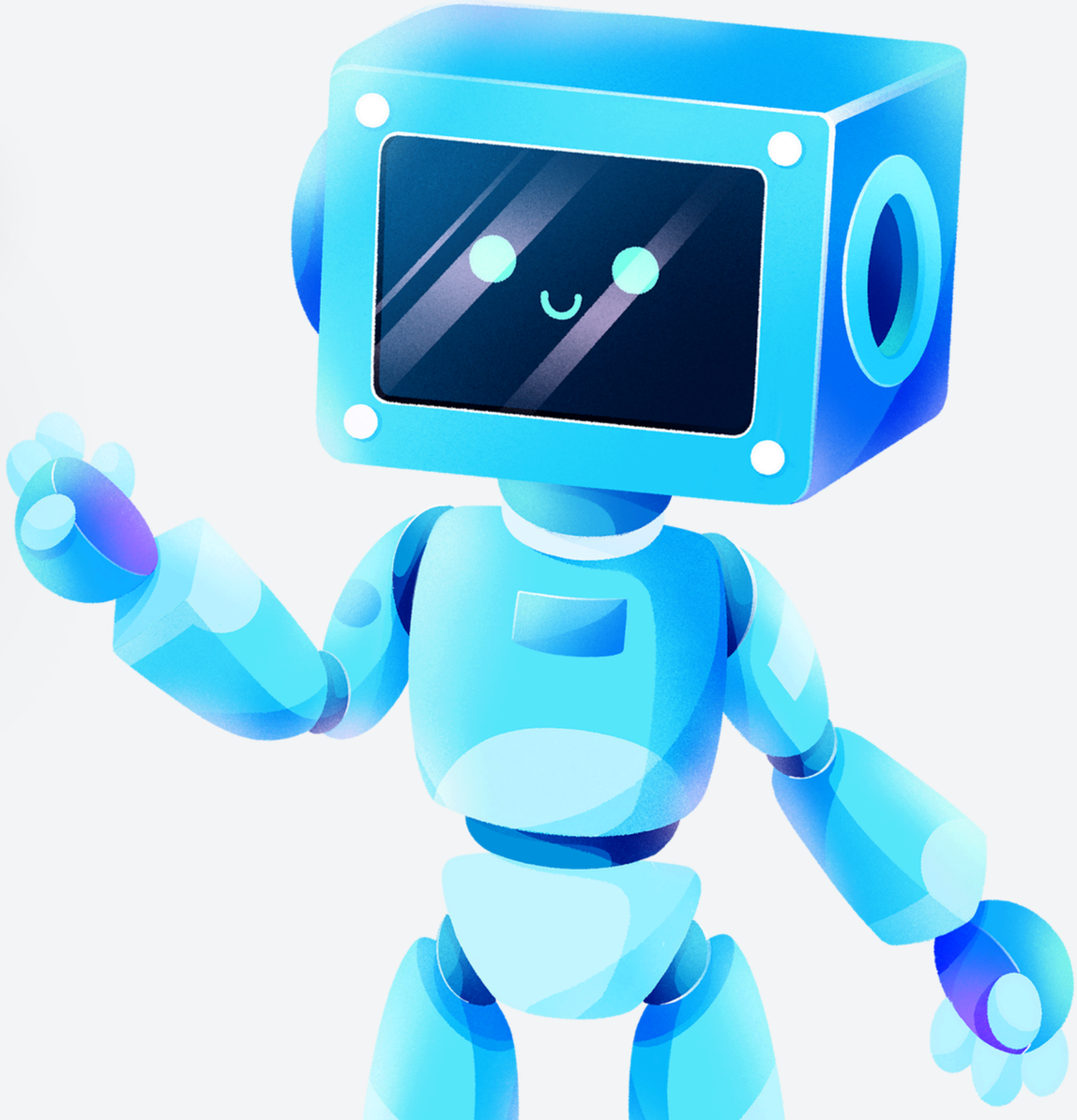
TRIPADVISOR SENTIMENT ANALYSIS PROJECT

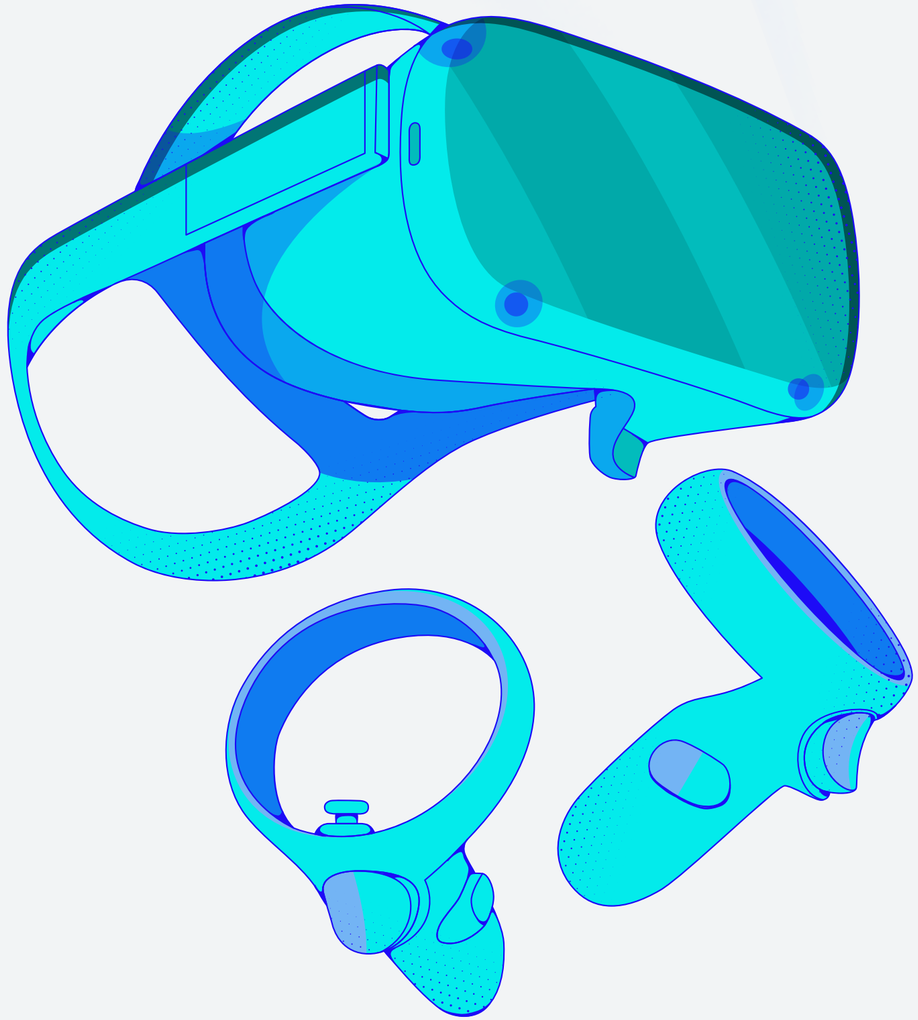
By Karen, Islamia, Marieme





AGENDA

1. Introduction
2. Data exploration
3. Structuring data
4. Data Preparation
5. Modelization training
6. Test
7. Conclusion



DATA EXPLORATION

	Review	Rating
0	nice hotel expensive parking got good deal sta...	4
1	ok nothing special charge diamond member hilto...	2
2	nice rooms not 4* experience hotel monaco seat...	3
3	unique, great stay, wonderful time hotel monac...	5
4	great stay great stay, went seahawk game aweso...	5
...	...	...
20486	best kept secret 3rd time staying charm, not 5...	5
20487	great location price view hotel great quick pl...	4
20488	ok just looks nice modern outside, desk staff ...	2
20489	hotel theft ruined vacation hotel opened sept ...	1
20490	people talking, ca n't believe excellent ratin...	2

20491 rows × 2 columns

```
df.duplicated().sum()
```

```
0
```

```
#Checking for missing values
df.isnull().sum()
```

```
Review    0
Rating    0
dtype: int64
```

Structure Data

In [14]:
```
#Test textblob on the first review
first_line=df['Review'].iloc[0]
print(first_line)
```

```
nice hotel expensive parking got good deal stay hotel anniversary, arrived late evening took advice previous review
s did valet parking, check quick easy, little disappointed non-existent view room room clean nice size, bed comfort
able woke stiff neck high pillows, not soundproof like heard music room night morning loud bangs doors opening clos
ing hear people talking hallway, maybe just noisy neighbors, aveda bath products nice, did not goldfish stay nice t
ouch taken advantage staying longer, location great walking distance shopping, overall nice experience having pay 4
0 parking night,
```

In [15]:
```
text = TB(first_line)
text.sentiment
```

Out[15]:
```
Sentiment(polarity=0.20874404761904758, subjectivity=0.687)
```

TextBlob is a Python library that analyzes text sentiment (positive or negative) and determines subjectivity (opinion vs. fact)

- Polarity = 0.2
- Subjectivy = 0.687

Structure Data

In [16]:
```
df['Sentiment'] = df['Review'].apply(lambda x: (TB(x).sentiment.polarity)
df.sample(10)
```

Out[16]:

	Review	Rating	Sentiment
5771	fine time room improvement just returned fabul...	4	0.242105
13387	good, just staying night pre-cruise booked pal...	5	0.244788
6333	iberville suites review having read number rev...	3	0.127706
9958	surprise location location location, hotel fan...	5	0.447667
4775	good london hotel wife stayed just 1 night lon...	4	0.469444
70	ace hotel, reasonably priced hotel great locat...	4	0.443333
7360	great location pleasant hotel no complaints sp...	5	0.583333
19425	modern chic far centre stayed vincci arena 3 n...	3	0.075758
13301	described wonderful stayed great hotel nights ...	4	0.417187
4797	fantabulistical weekend wife just returned hom...	5	0.369835

In [17]:
```
def score_to_sentiment_category(score):
    if -1 <= score < 0:
        return 'negative'
    if score == 0:
        return 'neutral'
    if 0 < score <= 1:
        return 'positive'
```

In [18]:
```
#We apply our previous function to'Sentiment' column'
df.Sentiment = df.Sentiment.apply(score_to_sentiment_category)
df.sample(10)
```

Out[18]:

	Review	Rating	Sentiment
15905	great base first-timers tokyo stayed standard ...	4	positive
11681	ghost hotel mirrors booked room small office c...	2	positive
14616	worth, husband stopped hotel 4 nights enjoyed ...	5	positive
12524	great value best beach partner just returned g...	4	positive
2755	ok barcelo bavaro beach stayed 3 nights 10sep-...	3	positive
18461	fantastic, couple teenager stayed christmas 20...	5	positive
5146	loved hotel stayed hotel napoleon week loved, ...	5	positive
18351	loved deluxe room nice contemporary design rou...	4	positive
5807	gorgeous beds, stayed stay new york like say h...	4	positive
13618	hola.ifa nothing like u.s.a- price paid 3 peop...	4	positive

Structure Data

In [19]:
```
#Viewing the number of times each category appears in the 'Sentiment' column
df['Sentiment'].value_counts()
```

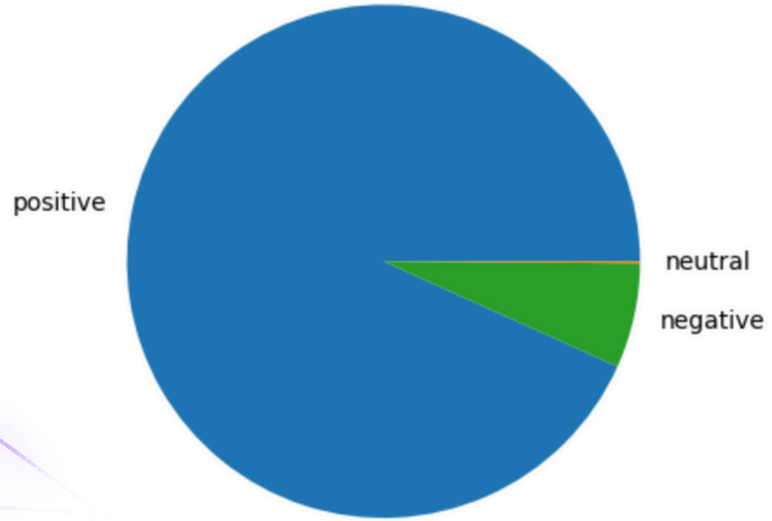
Out[19]:
```
Sentiment
positive    19112
negative     1356
neutral        23
Name: count, dtype: int64
```

The dataset seem highly imbalanced, let's visualize it:

In [20]:
```
f = np.array([19112, 1356, 23])
mylabels = ['positive', 'negative', 'neutral']

colors = ['#1f77b4', '#2ca02c', '#ff7f0e']

plt.pie(f, labels = mylabels, colors=colors)
plt.show()
```



Data preparation

Feature selection

```
x = df['Review']
```

```
y = df['Sentiment']
```

```
x.shape
```

```
(20491,)
```

```
y.shape
```

```
(20491,)
```

Splitting the data

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_siz
```

Data preparation

Tfidf Vectorizer : convert text data into numerical feature vector

Tfidf Vectorizer

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

```
tfidf_vector = TfidfVectorizer(ngram_range=(1,1), lowercase = True, stop_words= 'english')
```

```
x_train = tfidf_vector.fit_transform(x_train)
```

```
x_test  = tfidf_vector.transform(x_test)
```

MODELIZATION TRAINING

Training for decision trees

Entrée [31]:
```
pip install imbalanced-learn
```

```
Requirement already satisfied: imbalanced-learn in c:\users\islam\anaconda3\lib\site-packages (0.12.2)
Requirement already satisfied: numpy>=1.17.3 in c:\users\islam\anaconda3\lib\site-packages (from imbalanced-learn) (1.24.3)
Requirement already satisfied: scipy>=1.5.0 in c:\users\islam\anaconda3\lib\site-packages (from imbalanced-learn) (1.11.1)
Requirement already satisfied: scikit-learn>=1.0.2 in c:\users\islam\anaconda3\lib\site-packages (from imbalanced-learn) (1.
3.0)
Requirement already satisfied: joblib>=1.1.1 in c:\users\islam\anaconda3\lib\site-packages (from imbalanced-learn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\islam\anaconda3\lib\site-packages (from imbalanced-learn)
(2.2.0)
Note: you may need to restart the kernel to use updated packages.
```

Entrée [32]:
```
from imblearn.ensemble import BalancedBaggingClassifier
from sklearn.tree import DecisionTreeClassifier
```

Entrée [33]:
```
classifier = BalancedBaggingClassifier(estimator=DecisionTreeClassifier(),
                                       sampling_strategy='not majority',
                                       replacement=False,
                                       random_state=42)
classifier.fit(x_train, y_train)
```

Out[33]:
```
BalancedBaggingClassifier(estimator=DecisionTreeClassifier(), random_state=42,
                          sampling_strategy='not majority')
```

Training for decision trees

Evaluation of decision trees

Evaluation of decision trees

Entrée [45]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

Entrée [46]:
```
accuracy = accuracy_score(y_hat, y_test)
rounded_accuracy = round(accuracy, 2)
print("Accuracy :", rounded_accuracy)
```

```
Accuracy : 0.93
```

Entrée [47]:
```
precision_score(y_hat, y_test, average='weighted')
```

Out[47]:
```
1.0
```

Entrée [48]:
```
recall_score(y_hat, y_test, average='weighted')
```

Out[48]:
```
0.933636955107352
```

Entrée [49]:
```
f1_score(y_hat, y_test, average='weighted')
```

Out[49]:
```
0.9656796769851952
```

Training for logistic regression

Entrée [50]:
```
from sklearn.linear_model import LogisticRegression
```

Entrée [51]:
```
classif2 = BalancedBaggingClassifier(estimator=LogisticRegression(),
                                     sampling_strategy='not majority',
                                     replacement=False,
                                     random_state=42)
classif2.fit(x_train, y_train)
```

Out[51]:
```
BalancedBaggingClassifier(estimator=LogisticRegression(), random_state=42,
                          sampling_strategy='not majority')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

Training for logistic regression

Evaluation of logistic regression

Evaluation for logistic regression

[52]:
```
y_hat_logistic = classif2.predict(x_test)
```

[53]:
```
accuracy = accuracy_score(y_hat_logistic, y_test)
rounded_accuracy = round(accuracy, 2)
print("Accuracy :", rounded_accuracy)
```

```
Accuracy : 0.93
```

[54]:
```
precision_score(y_hat_logistic, y_test, average='weighted')
```

Out[54]:
```
1.0
```

[55]:
```
recall_score(y_hat_logistic, y_test, average='weighted')
```

Out[55]:
```
0.933636955107352
```

[56]:
```
f1_score(y_hat_logistic, y_test, average='weighted')
```

Out[56]:
```
0.9656796769851952
```

[ ]:
```
classifier = BalancedBaggingClassifier(estimator=DecisionTreeClassifier(),
                                       sampling_strategy='not majority',
                                       replacement=False,
                                       random_state=42)
classifier.fit(x_train, y_train)
```

[62]:
```
from sklearn.ensemble import RandomForestClassifier
```

[ ]:
```
# Initialize the BalancedBaggingClassifier with RandomForestClassifier as estimator
classif = BalancedBaggingClassifier(estimator=RandomForestClassifier(),
                                    sampling_strategy='not majority',
                                    replacement=False,
                                    random_state=42)

# Fit the classifier
classif.fit(x_train, y_train)
```

Training for Random Forest

Entrée [ ]:
```
classifier = BalancedBaggingClassifier(estimator=DecisionTreeClassifier(),
                                       sampling_strategy='not majority',
                                       replacement=False,
                                       random_state=42)
classifier.fit(x_train, y_train)
```

Entrée [62]:
```
from sklearn.ensemble import RandomForestClassifier
```

Entrée [ ]:
```
# Initialize the BalancedBaggingClassifier with RandomForestClassifier as estimator
classif = BalancedBaggingClassifier(estimator=RandomForestClassifier(),
                                    sampling_strategy='not majority',
                                    replacement=False,
                                    random_state=42)

# Fit the classifier
classif.fit(x_train, y_train)
```

Training for Random Forest

Evaluation for Random Forest

Entrée [65]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

Entrée [70]:
```
y_hat = classifier.predict(x_test)
```

Entrée [71]:
```
accuracy = accuracy_score(y_hat, y_test)
rounded_accuracy = round(accuracy, 2)
print("Accuracy :", rounded_accuracy)
```

```
Accuracy : 0.93
```

Entrée [72]:
```
precision_score(y_hat_logistic, y_test, average='weighted')
```

Out[72]:
```
1.0
```

Entrée [73]:
```
recall_score(y_hat, y_test, average='weighted')
```

Out[73]:
```
0.933636955107352
```

Entrée [74]:
```
f1_score(y_hat_logistic, y_test, average='weighted')
```

Out[74]:
```
0.9656796769851952
```

Evaluation for Random Forest

TEST

Entrée [82]:
```
sample_input_data = ['I had a wonderful time, good hotel, good service']
sample_input_data_to_array =  (tfidf_vector.transform(sample_input_data).toarray())
classif.predict(sample_input_data_to_array)
```

Out[82]:
```
array(['positive'], dtype=object)
```

Entrée [ ]:
```

```

CONCLUSION

RESOURCE PAGE

TECHNICAL ARCHITECTURE

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin varius, eros nec efficitur euismod, lectus turpis sollicitudin augue, non ultricies enim nunc sit amet augue. Quisque ante magna, varius et vehicula congue, viverra eu nulla. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin varius, eros nec efficitur euismod, lectus turpis sollicitudin augue, non ultricies enim nunc sit amet augue. Quisque ante magna, varius et vehicula congue, viverra eu nulla.