Enhancing Profits from Electronic Word-of-Mouth for Hotel Booking Electronic Intermediaries

유혜영 (2016195045)

Contents

1 Introduction

2 Data Collection

3 Data Preprocessing

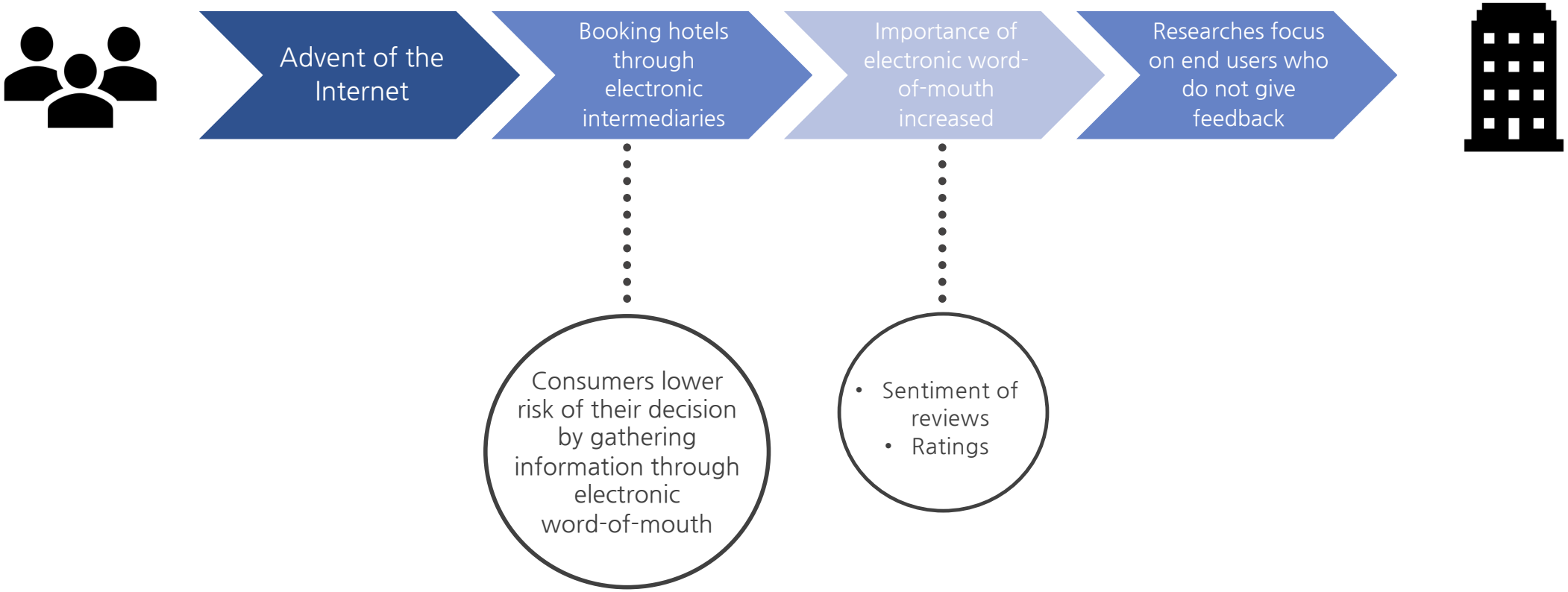
4 Data Exploration

5 Data Analysis

6 Conclusion

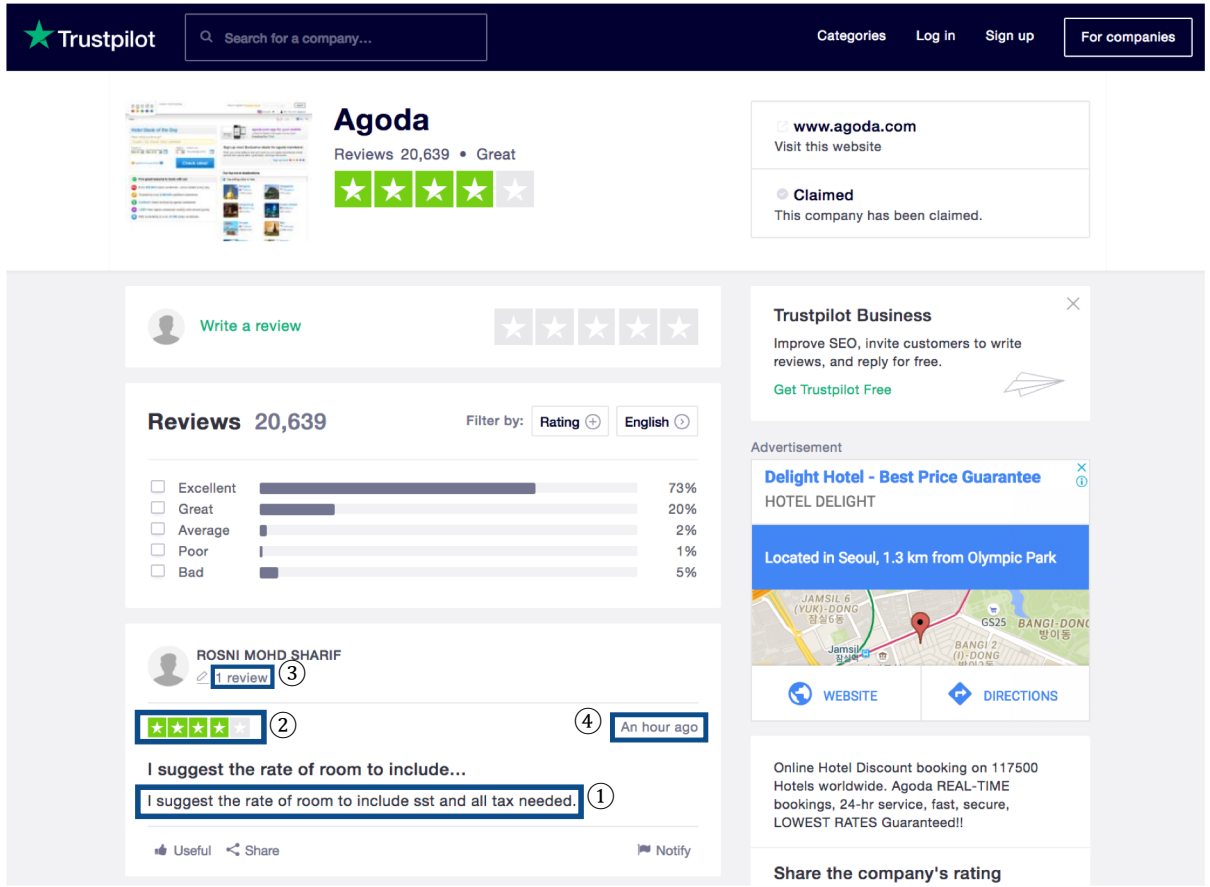
01

Current Focus of Research for Hotel Booking E-Intermediaries



02 Data Collection

- Hotel booking electronic intermediaries
  - Agoda, Booking.com, Expedia, Hostel World, Priceline
- Collected data
  (1) Content of review (used for sentiment analysis)
  (2) Rating of review
  (3) Number of review written by reviewer
  (4) Weekday the review was written



02 Data Collection

Step 1. Import necessary libraries for data collection

In [1]:
```
# Import necessary libraries
from bs4 import BeautifulSoup
import lxml
from urllib.request import urlopen
import pandas as pd
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
```

Step 2. Create a list for storing the URLs to scrape data from

In [126]:
```
# Create an empty list to store expedia review urls
expedia_review_urls = []
# Create a url for each expedia review page (there are 107)
for x in range (0,107):
    url = "https://www.trustpilot.com/review/www.expedia.com?languages=en&page="+str(x+1)
    # Append the urls to the expedia_review_urls list
    expedia_review_urls.append(url)
    #print(url)
# Print the number of urls in the expedia_review_urls list
print(len(expedia_review_urls))
```

Step 3. Scrape data (content, score, number of reviews, weekday)

In [127]:
```
# Process indicator
process = "Crawling webpage: "

# Making empty dictionary for expedia reviews
expedia_reviews = {}

# Making empty lists for expedia review components
expedia_review_contents = []
expedia_review_scores = []
expedia_review_numbers = []
expedia_review_weekdays = []

# For loop for scraping data for expedia_review_urls
for url in expedia_review_urls:

    html = urlopen(url)
    soup = BeautifulSoup(html,"html.parser")

    # Review content
    review_content_infos = soup.find_all("p",{"class":"review-info__body__text"})
    for review_content_info in review_content_infos:
        review_content_info_str = str(review_content_info)
        review_content = review_content_info_str.split('body__text">')[1].split('</p>')[0].replace("\n","").replac
        expedia_review_contents.append(review_content)

    # Review score
    review_score_infos = soup.find_all("script",{"data-initial-state":"review-info"})
    for review_score_info in review_score_infos:
        review_score = review_score_info.text.split('stars":')[1].split("}")[0]
        expedia_review_scores.append(review_score)

    # Number of reviews the reviewer wrote
    review_number_infos = soup.find_all("span",{"class":"consumer-info__details__review-count"})
    for review_number_info in review_number_infos:
        review_number_info_str = str(review_number_info)
        review_number = review_number_info_str.split('review-count">')[1].split(' review')[0].replace("\n","")
        expedia_review_numbers.append(review_number)

    # Weekday of review
    review_date_infos = soup.find_all("div",{"class":"header__verified__date"})
    for review_date_info in review_date_infos:
        review_date_info_str = str(review_date_info)
        review_weekday = review_date_info_str.split('title="')[1].split(",")[0]
        expedia_review_weekdays.append(review_weekday)


    # Print process to know how your crawler is doing
    print(process + str(url))

# Add all the information collected in the four lists to the dictionary
expedia_reviews["review_contents"] = expedia_review_contents
expedia_reviews["review_numbers"] = expedia_review_numbers
expedia_reviews["review_scores"] = expedia_review_scores
expedia_reviews["review_weekdays"] = expedia_review_weekdays

# Add the company name "expedia" to company_name
expedia_reviews["company_name"] = "expedia"
```

02 Data Collection

Step 4. Check number of data in each column

In [133]:
```
# Check the number of data in each column to see if there is any data that does not match up
print(len(expedia_review_weekdays))
print(len(expedia_review_scores))
print(len(expedia_review_contents))
print(len(expedia_review_numbers))
```

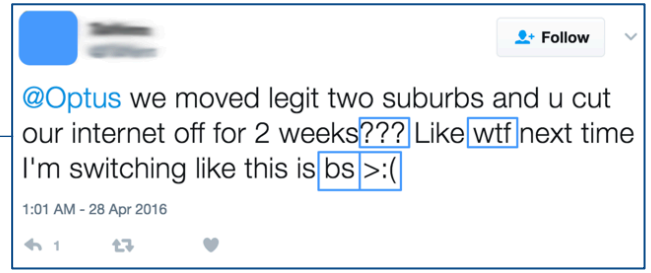
Step 5. Calculate and determine the sentiment valence of review content

In [134]:
```
# Assign SentimentIntensityAnalyzer() to variable analyzer
analyzer = SentimentIntensityAnalyzer()

# Create an empty list that is labeled expedia_compounds to collect valence values
expedia_compounds = []

# For loop to get the polarity scores of the review content and the valence
for expedia_review_content in expedia_review_contents:
    # Polarity score of the review content
    vs = analyzer.polarity_scores(expedia_review_content)
    # Determining valence of review content
    if vs['compound'] >= 0.5:
        expedia_compounds.append("positive")
    elif vs['compound'] <= -0.5:
        expedia_compounds.append("negative")
    else:
        expedia_compounds.append("neutral")

# Assign the valence values to the key sentiment_compound
expedia_reviews["sentiment_compound"] = expedia_compounds
# Check the number of data in expedia_compounds to check if it matches with review_contents
print(len(expedia_compounds))
```

Step 6. Save data frame into a tsv file

In [135]:
```
# Make a dataframe with the expedia_reviews dictionary
expedia_df = pd.DataFrame(expedia_reviews)
expedia_df
```

In [136]:
```
# Save expedia_reviews dataframe to tsv file on laptop
expedia_df.to_csv("/Users/angieryu2202/Desktop/expedia_reviews_tp.tsv'
```



03 Data Preprocessing

Step 1. Import necessary libraries for data processing

In [219]:
```
# Import necessary module for data processing
import pandas as pd
```

Step 2. Read tsv file as data frame and take out unnecessary columns

In [220]:
```
# Read tsv file as a dataframe and save it to variable expedia_org_df
expedia_org_df = pd.read_csv("/Users/angieryu2202/Desktop/expedia_reviews_tp.tsv", sep="\t")
# Check if the file has been read correctly into the dataframe
expedia_org_df.head(5)
```

In [221]:
```
# Save only columns from review_numbers to sentiment_compound onto variable expedia_org_df1
expedia_org_df1 = expedia_org_df.iloc[:,2:7]
# Check if the change has been implemented properly
expedia_org_df1.head(5)
```

Step 3. Drop null value containing rows

In [222]:
```
# Check shape of dataset
# 2123 instances with 5 attributes
expedia_org_df1.shape
```

In [223]:
```
# Check null value
expedia_org_df1.dropna()
```

In [224]:
```
# Check the shape again
expedia_org_df1.shape
# The dataset had no null value from the beginning
```

Step 4. Concatenate the 5 data frames

In [245]:
```
# Group the dataframes into one list
grouped_frames = [expedia_org_df1, booking_org_df1, agoda_org_df1, priceline_org_df1, hostel_org_df1]
# Concatenate the dataframes on grouped_frames and save it to a variable called grouped_df
grouped_df = pd.concat(grouped_frames)
```

In [249]:
```
# Put the company_name column at the end
preprocessed_df = grouped_df[['review_numbers', 'review_scores', 'review_weekdays', 'sentiment_compound', 'company_
```

In [250]:
```
# Check if the company_name column has been placed at the last column
preprocessed_df.head(5)
```

03 Data Preprocessing

Step 5. Change categorical values to numerical values

In [251]:
```
# Check which data types are in the dataframe
preprocessed_df.info()
```

In [252]:
```
# Select only the columns that contain object values
preprocessed_df_cat_columns = preprocessed_df.select_dtypes(['object']).columns[:-1]
```

In [253]:
```
# Check the columns
preprocessed_df_cat_columns
```

(1) Select object type value containing columns

In [254]:
```
# Use loop to change the object type to category type
for i in preprocessed_df_cat_columns:
    preprocessed_df[i] = preprocessed_df[i].astype('category')
```

In [255]:
```
# Check the change from object type to category type
preprocessed_df.info()
```

(2) Change object value to categorical value

In [256]:
```
# Change the categorical value to numerical value using lambda and apply
preprocessed_df[preprocessed_df_cat_columns] = preprocessed_df[preprocessed_df_cat_columns].apply(lambda x: x.cat.co
```

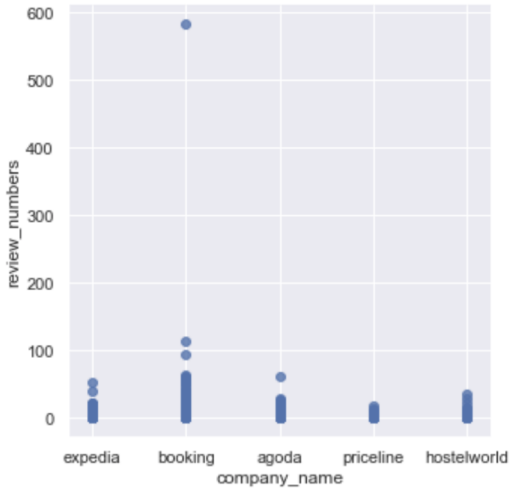
In [257]:
```
# Check the change
preprocessed_df.head(5)
```

(3) Change categorical value to numerical value using lambda

03 Data Preprocessing

Step 6. Detecting outliers

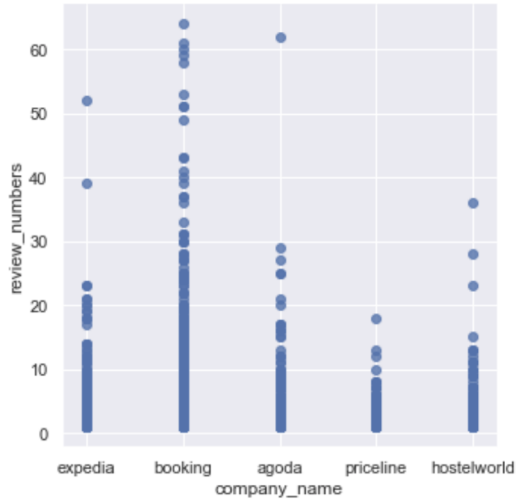
In [326]:
```
# Import necessary modules such as matplotlib.pylab and seaborn to visualize the data
import matplotlib.pylab as plt
import seaborn as sns

# Plot and show facetgrid for review number dispersion for all the companies
fig = sns.lmplot(x="company_name", y="review_numbers", data=preprocessed_df, fit_reg=False, legend=False)
plt.show()
```



In [330]:
```
# Drop all the rows that have the outliers of over a 80 number of reviews for one reviewer
booking_org_df2 = booking_org_df1[booking_org_df.review_numbers < 80]
grouped_df1 = grouped_df[grouped_df.review_numbers < 80]
preprocessed_df1 = preprocessed_df[preprocessed_df.review_numbers < 80]
```

In [331]:
```
# Save preprocessed_df1 to tsv file on laptop
preprocessed_df1.to_csv("/Users/angieryu2202/Desktop/preprocessed_df.tsv", sep="\t")
```

In [324]:
```
# Plot and show facetgrid for review number dispersion for all the companies after deleting the outlier
fig = sns.lmplot( x="company_name", y="review_numbers", data=preprocessed_df1, fit_reg=False, legend=False)
plt.show()
```



04 Data Exploration

	Expedia	Booking	Agoda	Priceline	Hostelworld
count	2123	2137	2140	2140	2140
mean	1.769	4.167	1.578	1.382	1.518
std	2.522	6.092	2.376	0.998	1.634
min	1	1	1	1	1
25%	1	1	1	1	1
50%	1	2	1	1	1
75%	2	5	1	1	1
max	52	64	62	18	36

[Number of reviews written by one reviewer]

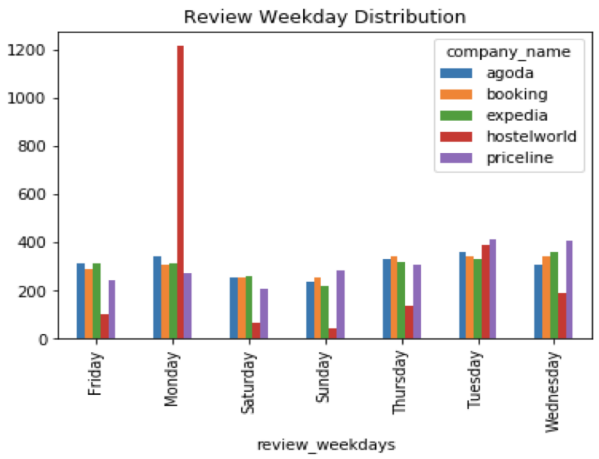
1. Expedia
   - At least 25% wrote more than 2 reviews
2. Booking.com
   - At least 50% wrote more than 2 reviews
   - At least 25% wrote more than 5 reviews
3. Except for Booking.com, 50% or more reviewers of each company write only one review.

	Expedia	Booking	Agoda	Priceline	Hostelworld
count	2123	2137	2140	2140	2140
mean	1.324	1.779	4.017	4.476	4.439
std	1.015	1.428	1.541	1.055	1.143
min	1	1	1	1	1
25%	1	1	4	4	4
50%	1	1	5	5	5
75%	1	2	5	5	5
max	5	5	5	5	5

[Review score]

1. Expedia, Booking.com
   - Less than 25% give a score over 1
2. Agoda, Priceline, Hostel World
   - More than 75% give a score of 4 or 5

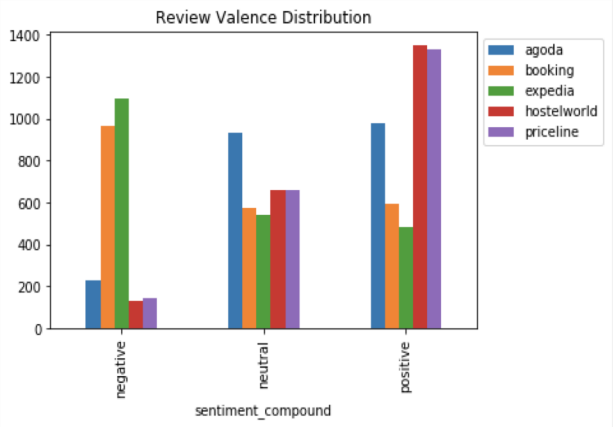
04 Data Exploration



	Concatenated Dataframe Weekday Value Counts
Monday	2454
Tuesday	1836
Wednesday	1612
Thursday	1439
Friday	1258
Saturday	1044
Sunday	1037

[Weekday review was written on]

1. Tuesdays, Wednesdays, and Thursdays are when most reviewers write their reviews
   - Monday has the most reviews, but mostly due to Hostel World
2. Hostel World
   - High peak of review on Monday
   - Decreases dramatically from Tuesday to Sunday

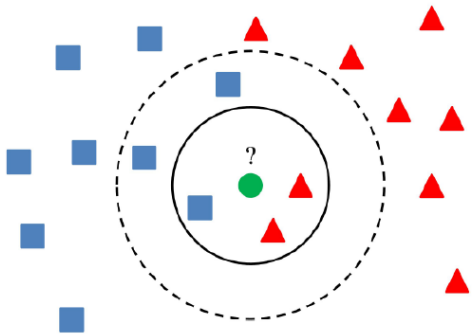


	Concatenated Dataframe Valence Value Counts
positive	4735
neutral	3372
negative	2573

[Sentiment valence of review]

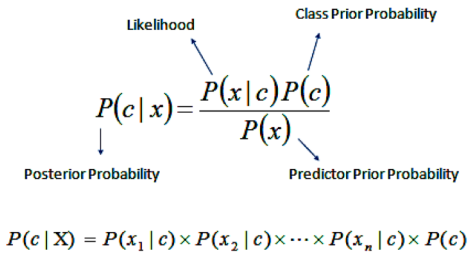
1. Positive > Neutral > Negative
2. Negative reviews due mainly on
   - Expedia and Booking.com
3. Positive reviews due mainly on
   - Agoda, Hostel World, and Priceline
4. Agoda
   - Has same quartiles as Hostel World and Priceline, but has a significantly higher portion of neutral reviews

05 Data Analysis



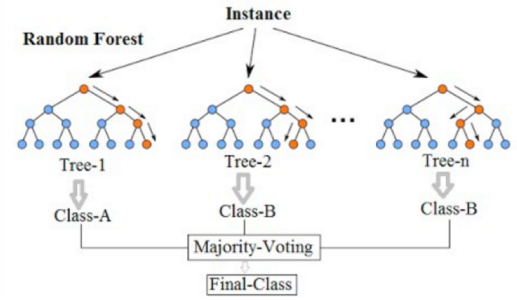
- Works well on low dimensional data
- Works well on smaller data sets
- Easy to implement
  - 1 parameter: k value

1. K-Nearest Neighbors



- Works well on mixed data sets of continuous numerical values and categorical values
- Predicts well in case of multi-class prediction

2. Gaussian Naïve Bayes



- Works well on mixed data sets of continuous numerical values and categorical values
- Less problems with overfitting since there are more random and diverse decision trees

3. Random Forest

05 Data Analysis

Step 1. Setting up training and testing sets

(1) Creating data frame that contains the input and output variables

In [1]:
```
# Import necessary module to handle the dataframe
import pandas as pd

# Read the preprocessed_df.tsv file
df = pd.read_csv('/Users/angieryu2202/Desktop/preprocessed_df.tsv', sep ="\t")

# Check if the data has been read properly
df.head(5)
```

In [2]:
```
# Take out the unnecessary "Unnamed: 0" column
df1 = df.iloc[:, 1:6]
# Check if the column has been taken out properly
df1.head(5)
```

(2) Setting the input variables (X) and output variables (y) from the data frame

In [4]:
```
# Set X and y from the dataframe
import numpy as np
feature_names = [i for i in df1.columns if df1[i].dtype in [np.int64]]
X = df1[feature_names]
y = df1.iloc[:, -1].values
```

(3) Dividing the data set into training and testing set (7:3 ratio)

In [5]:
```
# Divide the dataset into training set and testing set
# Test set size is 0.3
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state = 0)
```

(4) Creating a training set and testing set on which feature scaling is conducted

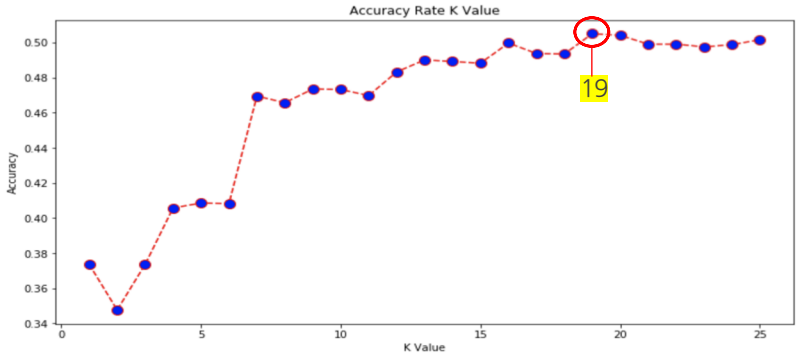
In [6]:
```
# Feature Scaling is needed since the review_numbers data has a significantly wider range than other columns
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)

X_train1 = scaler.transform(X_train)
X_test1 = scaler.transform(X_test)
```

05 Data Analysis

Step 2. Implementing machine learning classification algorithms

(1) Find the most optimal k value (for KNN only)

In [26]:
```python
# Import necessary modules for finding k value with the highest accuracy
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt

# Make empty dictionary for accuracies and k values
k_detector = {}

# Make empty lists for the values of accuracies and kvalues
accuracies = []
kvalues = []

# For k values from 1 to 25, find the accuracy of each value
for i in range(1,26):
    classifier = KNeighborsClassifier(n_neighbors=i)
    classifier.fit(X_train1, y_train)
    y_test_pred = classifier.predict(X_test1)
    accuracy = accuracy_score(y_test,y_test_pred)
    accuracies.append(accuracy)
    kvalues.append(i)

# Append the lists as values to the keys of kvalues and accuracies
k_detector['kvalues'] = kvalues
k_detector['accuracies'] = accuracies

# Plot the accuracy plot
plt.figure(figsize=(12, 6))
plt.plot(range(1, 26), accuracies, color='red', linestyle='dashed', marker='o',
         markerfacecolor='blue', markersize=10)
plt.title('Accuracy Rate K Value')
plt.xlabel('K Value')
plt.ylabel('Accuracy')

# Find the k value that has the highest accuracy
k_detector_df = pd.DataFrame(k_detector)
k_detector_df.loc[k_detector_df.iloc[:, -1].idxmax()]
```



(2) Train and test the classifiers

In [27]:
```python
# Train the classifier with optimal k value
from sklearn.neighbors import KNeighborsClassifier
knn_clf = KNeighborsClassifier(n_neighbors=19)
knn_clf.fit(X_train1, y_train)

# Predict output
knn_y_train_pred = knn_clf.predict(X_train1)
knn_y_test_pred = knn_clf.predict(X_test1)
```

(3) Get the classification reports, confusion matrix, and accuracy score

In [64]:
```python
# Import classifcation_report, confusion_matrix, and accuracy_score
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

# Check the confusion matrix (row-true class, column-predicted class)
knn_cm = pd.DataFrame(confusion_matrix(y_test, knn_y_test_pred), index = ["agoda","booking","expedia","hostelworld"
print(knn_cm)

# Check the classification report
print(classification_report(y_test, knn_y_test_pred))

# Check the accuracy rate
print ("Accuracy rate for the training set: " + str(accuracy_score(y_train, knn_y_train_pred)*100) + "%")
print ("Accuracy rate for the testing set: " + str(accuracy_score(y_test, knn_y_test_pred)*100) + "%")
```

* GNB: Used the non-scaled training and testing sets

05 Data Analysis

Step 3. Observing permutation importance scores

In [111]:

```
# Import necessary modules to determine the permutation importance of each feature
import eli5
from eli5.sklearn import PermutationImportance

# Determine permutation importance of each feature
perm = PermutationImportance(knn_clf).fit(X_test, y_test)
eli5.show_weights(perm, feature_names = X_test.columns.tolist())
```

Weight	Feature
0.0664 ± 0.0040	review_weekdays
0.0650 ± 0.0070	review_numbers
0.0138 ± 0.0019	review_scores
0.0101 ± 0.0045	sentiment_compound

1. K-Nearest Neighbors

Weight	Feature
0.1569 ± 0.0116	review_scores
0.0365 ± 0.0106	review_numbers
0.0052 ± 0.0070	review_weekdays
0.0022 ± 0.0034	sentiment_compound

2. Gaussian Naïve Bayes

Weight	Feature
0.0443 ± 0.0063	review_numbers
0.0214 ± 0.0065	sentiment_compound
0.0190 ± 0.0070	review_weekdays
0 ± 0.0000	review_scores

3. Random Forest

05 Data Analysis

Step 4. Comparing results of the models

Hotel Booking Electronic Intermediary	Support
Agoda	644
Booking	640
Expedia	618
Hostel World	646
Priceline	656
Total	3204

	Accuracy Score (%)	F1 Score
K-Nearest Neighbors	50.593	0.50
Gaussian Naive Bayes	39.794	0.32
Random Forest	48.907	0.48

Accuracy can be used to compare models for balanced data

For cases where a classifier may predict one class well, but not others, we need to consider F1 score

KNN has the highest accuracy score and the F1 score

06 Conclusion

Weight	Feature
0.0664 ± 0.0040	review_weekdays
0.0650 ± 0.0070	review_numbers
0.0138 ± 0.0019	review_scores
0.0101 ± 0.0045	sentiment_compound

Permutation Importance of the KNN classifier

1. Weekday review was written
   - Very high number of reviews written on Monday
2. Number of reviews
   - Variance seemed to be wider than other hotel booking electronic intermediaries
   - 50% wrote at least 2 or more reviews
   - 25% wrote at least 5 reviews

Arduous to give feedback

3. Low review scores and sentiment valence

Hypercritical

Arduous to give feedback + Hypercritical =

Booking.com Feedback Returning Customer Base

07 References

Casado-Díaz, A. B., Pérez-Naranjo, L. M., & Sellers-Rubio, R. (2017). Aggregate consumer ratings and booking intention: the role of brand image. Service Business, 11(3), 543-562.

East, R., Hammond, K., & Lomax, W. (2008). Measuring the impact of positive and negative word of mouth on brand purchase probability. International journal of research in marketing,25(3),215-224.

Hong, S., & Park, H. S. (2012). Computer-mediated persuasion in online reviews: Statistical versus narrative evidence. Computers in Human Behavior, 28(3), 906-919.

Hutto, Clayton J. and Eric Gilbert. 2014. "VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text". Eighth International Conference on Weblogs and Social Media, 14.

Koehrsen, W. (2017, December 28). Random Forest Simple Explanation. Retrieved from https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d

Kolowich, L. (2018, September 19). 22 Customer Review Sites for Collecting Business & Product Reviews. Retrieved from https://blog.hubspot.com/service/customer-review-sites

Le, J. (n.d.). A Tour of The Top 10 Algorithms for Machine Learning Newbies. Retrieved from https://towardsdatascience.com/a-tour-of-the-top-10-algorithms-for-machine-learning-newbies-dde4edffae11

Line, N. D., & Runyan, R. C. (2012). Hospitality marketing research: Recent trends and future directions. International Journal of Hospitality Management, 31(2), 477-488.

McCormick, E. (2018, August 1). Best Hotel Booking Services of 2018. Retrieved from https://www.toptenreviews.com/services/home/best-hotel-booking-services/

Murray, K. B. (1991). A test of services marketing theory: consumer information acquisition activities. The journal of marketing, 10-25.

07

References

Noone, B. M., & McGuire, K. A. (2014). Effects of price and user-generated content on consumers' prepurchase evaluations of variably priced services. Journal of Hospitality & Tourism Research, 38(4), 562-581.

Pulipaka, G. (2017, January 15). Applying Gaussian Naïve Bayes Classifier in Python: Part One. Retrieved from https://medium.com/@gp_pulipaka/applying-gaussian-naïve-bayes-classifier-in-python-part-one-9f82aa8d9ec4

Random Forest Analysis in ML and when to use it. (2018, August 17). Retrieved from https://www.newgenapps.com/blog/random-forest-analysis-in-ml-and-when-to-use-it

Ray, S. (2017, September 11). 6 Easy Steps to Learn Naive Bayes Algorithm. Retrieved from https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/

Robinson, S. (2018, February 15). K-Nearest Neighbors Algorithm in Python and Scikit-Learn. Retrieved from https://stackabuse.com/k-nearest-neighbors-algorithm-in-python-and-scikit-learn/

Viglia, G., Furlan, R., & Ladron-de-Guevara, A. (2014). Please, talk about it! When hotel popularity boosts preferences. International Journal of Hospitality Management, 42, 155-164.

Winship, T. (2016, May 22). The 10 Best Hotel Booking Sites. Retrieved from https://www.smartertravel.com/the-10-best-hotel-booking-sites/

Xie, H. J., Miao, L., Kuo, P. J., & Lee, B. Y. (2011). Consumers' responses to ambivalent online hotel reviews: The role of perceived source credibility and pre-decisional disposition. International Journal of Hospitality Management, 30(1), 178-183.

Thank You :)