Final project

NIT6160 Final Project_Data Warehousing and Mining

—

Sakshi

ID

Table of Contents

Introduction

Airbnb is a disruptive force in the modern hospitality industry, changing the way that discerning travellers choose their accommodations by upending preconceived notions. This final project delves deeply into an analysis of Airbnb's effects on the Barwon South West region in Victoria, Australia. Utilizing carefully selected datasets from Inside Airbnb, namely ‘listings.csv’ and ‘reviews.csv’, the project applies advanced data mining and machine learning techniques to extract meaningful insights for prospective investors and hosts. The project is structured around key tasks, including data preprocessing, exploratory data analysis, predictive modeling, and advanced sentiment analysis, aiming not only to extract actionable intelligence but also to showcase the practical application of data warehousing and mining in a real-world context. This report unfolds as a narrative, detailing methodologies, presenting results, engaging in insightful discussions, elucidating encountered challenges, and meticulously citing pertinent references. It offers a comprehensive exploration into the intricate dynamics of Airbnb within the Barwon South West region.

Methods Applied

A number of methods are used in this project in order to perform each task, which are described as below:

Task 1: Data Pre-processing

Refining the dataset for further studies was the main goal of the data pre-processing stage. To enable effective data manipulation, the Pandas library was used to load the listings.csv file. Simplicity and rich documentation of **Pandas,** is the main reason why it is selected for data pre-processing[1].

- **Selecting Columns:** To maximize model performance and combat the curse of dimensionality, carefully choose pertinent columns that preserve important information while removing unnecessary ones.

- **Cleaning Prices:** In order to guarantee data integrity, steps were made to convert currencies to floating-point values.
- **Remove Null Values:** Different pandas methods were used to identify, fill, and drop the missing values including isnull(), fillna(), and dropna() in order to discard unuseful data and improve the model's overall performance
- **Dealing with Non-numeric columns:** One hot encoding using technique pandas.get_dummies() method is used to get rid of categorical data in the columns and convert that data to useful information, as the machine learning model accepts numeric data only.

Task 2: Exploratory Data Analysis (EDA) with Data Visualization

A variety of statistical visualization approaches were applied in order to obtain insights into the dataset and comprehend the distribution of pricing and accommodations. To visualize the distribution of prices, boxplots were made using the Matplotlib and Seaborn libraries. Folium, an interactive map-making Python module, was used to visualize geospatial data. This made it easier to investigate the distribution of accommodations geographically using latitude and longitude. These tools were selected because they provided a thorough visual depiction of the data and were versatile and simple to integrate.

Task 3: Building the Accommodation Prediction Model

A popular supervised learning method called XGBoost was selected to build the accommodation price prediction model. Because of its reliable effectiveness for regression tasks, XGBoost is a good choice for forecasting numerical outcomes such as hotel costs. A training (80%) and testing (20%) set of the dataset was created using sklearn's train_test_split method in order to assess the performance of the model. XGBoost was an appropriate choice for this predictive modeling challenge because of its capacity to manage missing values, handle complex relationships in the data, and avoid overfitting.

Task 4: (Advanced tasks): Sentiment Analysis

The Natural Language Toolkit (NLTK) package was used to do sentiment analysis on review comments. With its array of text analysis tools, NLTK is a good choice for sentiment analysis of textual data [2].

- **Sentiment Analysis of Review Comments:** This technique uses natural language processing to identify the sentiments expressed in review comments, giving important information about the preferences of the customer.
- **Examination of the Motives Behind Likes and Dislikes:** An unrestricted investigation of elements that affect patron happiness, including hotel view, location, and employee demeanor.

Results

Load Modules

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import folium
from sklearn.model_selection import train_test_split
from xgboost import XGBRegressor
from nltk.sentiment import SentimentIntensityAnalyzer
```

Load the Dataset

```
listings = pd.read_csv('listings.csv')
```

Task 1: Data Pre-processing

The main focus of this task is to remove the irrelevant columns and values to improve model's prediction accuracy. In order to do so, we have to carefully analyze the columns and data first.

1) Deciding which columns to work with

```
print(listings.columns)
```

```
Index(['id', 'listing_url', 'scrape_id', 'last_scraped', 'source', 'name',
       'description', 'neighborhood_overview', 'picture_url', 'host_id',
       'host_url', 'host_name', 'host_since', 'host_location', 'host_about',
       'host_response_time', 'host_response_rate', 'host_acceptance_rate',
       'host_is_superhost', 'host_thumbnail_url', 'host_picture_url',
       'host_neighbourhood', 'host_listings_count',
       'host_total_listings_count', 'host_verifications',
       'host_has_profile_pic', 'host_identity_verified', 'neighbourhood',
       'neighbourhood_cleansed', 'neighbourhood_group_cleansed', 'latitude',
       'longitude', 'property_type', 'room_type', 'accommodates', 'bathrooms',
       'bathrooms_text', 'bedrooms', 'beds', 'amenities', 'price',
       'minimum_nights', 'maximum_nights', 'minimum_minimum_nights',
       'maximum_minimum_nights', 'minimum_maximum_nights',
       'maximum_maximum_nights', 'minimum_nights_avg_ntm',
       'maximum_nights_avg_ntm', 'calendar_updated', 'has_availability',
       'availability_30', 'availability_60', 'availability_90',
       'availability_365', 'calendar_last_scraped', 'number_of_reviews',
       'number_of_reviews_ltm', 'number_of_reviews_l30d', 'first_review',
       'last_review', 'review_scores_rating', 'review_scores_accuracy',
       'review_scores_cleanliness', 'review_scores_checkin',
       'review_scores_communication', 'review_scores_location',
       'review_scores_value', 'license', 'instant_bookable',
       'calculated_host_listings_count',
       'calculated_host_listings_count_entire_homes',
       'calculated_host_listings_count_private_rooms',
       'calculated_host_listings_count_shared_rooms', 'reviews_per_month'],
      dtype='object')
```

As we can see there are so many columns in the given dataset, we will select the columns, those can potentially affect the property's price

```
columns_of_interest = ['id', 'name', 'description', 'neighbourhood', 'latitude', 'longitude', 'property_type', 'room_type', 'accommodates', 'bathrooms',
                       'bedrooms', 'beds', 'amenities', 'number_of_reviews', 'review_scores_rating', 'price']
listings_updated = listings[columns_of_interest]
```

```
listings_updated.head(2)
```

	id	name	description	neighbourhood	latitude	longitude	property_type	room_type	accommodates	bathrooms	bedrooms	beds	ame
0	894708633943425209	Rental unit in Antwerpen · 1 bedroom · 4 beds ...	A bright and beautiful newly renovated apartme...	Antwerpen, Vlaams Gewest, Belgium	51.223309	4.402095	Entire rental unit	Entire home/apt	8	NaN	1.0	4.0	["Extra p blar "Conditi
1	741093042494325959	Serviced apartment in Antwerpen · Studio · 1 b...	Elegant ingerichte kamer in het hart van Antwe...	NaN	51.224110	4.398240	Private room in serviced apartment	Private room	2	NaN	NaN	1.0	["Wifi" extingu "Ded w

Number of rows and columns in dataset before cleaning

```
rows, columns = listings_updated.shape
print("Rows:", rows)
print("Columns: ", columns)
```

```
Rows: 2702
Columns:  16
```

- Check the number of null values in each column

```
print(listings_updated.isnull().sum())
```

```
id                          0
name                        0
description                61
neighbourhood            1225
latitude                    0
longitude                   0
property_type               0
room_type                   0
accommodates                0
bathrooms                2702
bedrooms                  473
beds                       21
amenities                   0
number_of_reviews           0
review_scores_rating      438
price                       0
dtype: int64
```

- In oredr to deal with null values, we have two options, either to remove those records or fill them with assumed values.
- As we can see in the summary above, "bathroom" column has no record for all 2702 rows. Hence, we can exclude that column.

```
listings_cleaned = listings_updated.drop(columns=["bathrooms"])
print(listings_cleaned.shape)
```

```
(2702, 15)
```

- Let's fill the "bedrooms", "beds" with the most repeated values

```
# Calculate the mode
bedrooms_mode = listings_cleaned['bedrooms'].mode().iloc[0]
beds_mode = listings_cleaned['beds'].mode().iloc[0]
```

Fill the Null values

```
listings_cleaned["bedrooms"].fillna(bedrooms_mode, inplace=True)
listings_cleaned["beds"].fillna(beds_mode, inplace=True)
```

```
print(listings_cleaned.isnull().sum())
```

```
id                          0
name                        0
description                61
```

```
neighbourhood           1225
latitude                   0
longitude                  0
property_type              0
room_type                  0
accommodates               0
bedrooms                   0
beds                       0
amenities                  0
number_of_reviews          0
review_scores_rating     438
price                      0
dtype: int64
```

As we can see in the summary, those columns no longer have any missing values in them

- Let's fill the "review_scores_rating" with the average value of that column

```
listings_cleaned['review_scores_rating'].fillna(listings_cleaned.review_scores_rating.mean(), inplace=True)
```

```
print(listings_cleaned.isnull().sum())
```

```
id                         0
name                       0
description               61
neighbourhood           1225
latitude                   0
longitude                  0
property_type              0
room_type                  0
```

```
beds                       0
amenities                  0
number_of_reviews          0
review_scores_rating       0
price                      0
dtype: int64
```

Delete the records, those have still have null values

The code below, will remove all the rows, where "description" and "neighbourhood" has no value. We cannot calculate mean, median or mode for these columns, as they are not numerical ones.

```
listings_cleaned.dropna(inplace=True)
```

```
listings_cleaned.isnull().sum()
```

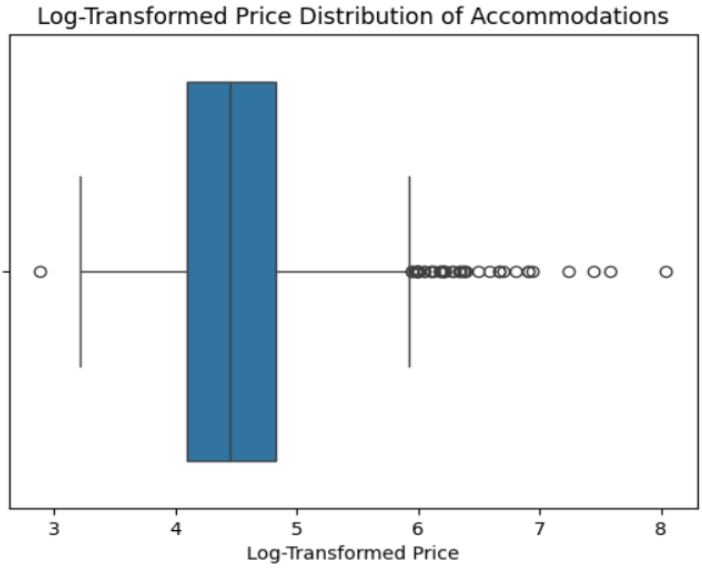
```
id                      0
name                    0
description             0
neighbourhood           0
latitude                0
longitude               0
property_type           0
room_type               0
accommodates            0
bedrooms                0
beds                    0
amenities               0
number_of_reviews       0
review_scores_rating    0
price                   0
dtype: int64
```

Task 2: Exploratory Data Analysis (EDA) with Data Visualization

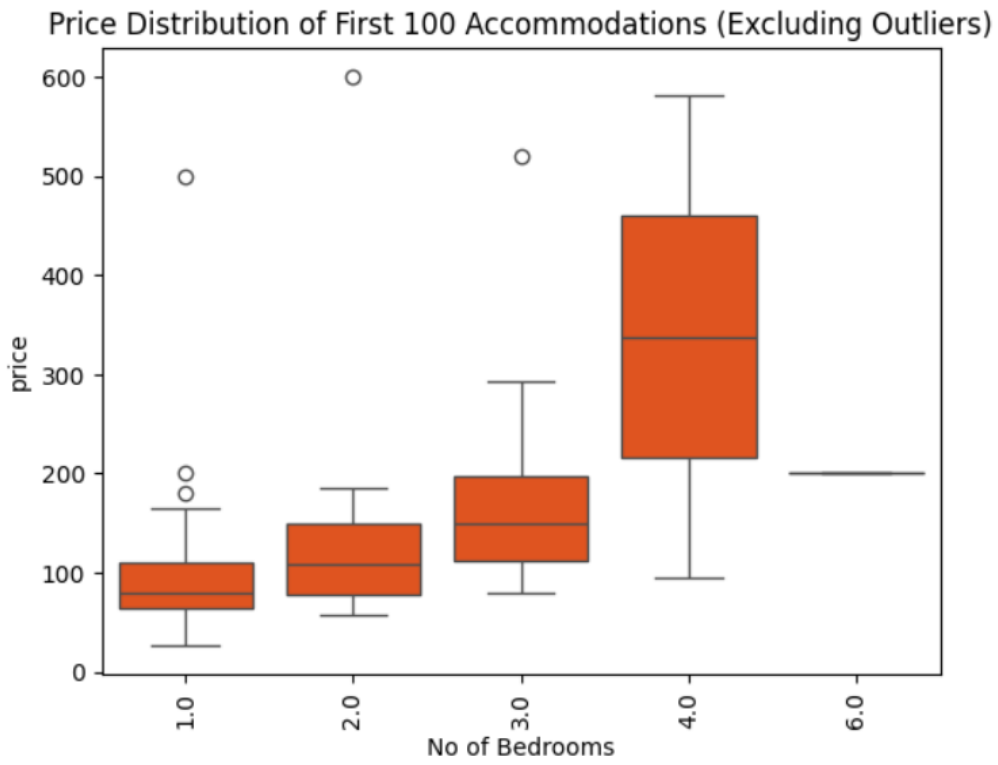
3) Price column visualization

```
# Boxplot for price distribution
plt.figure(figsize=(10, 6))
sns.boxplot(x='price', data=listings_cleaned)
plt.title('Price Distribution of Accommodations')
plt.xlabel('Price')
plt.show()
```


```
# Create a boxplot for log-transformed prices
# plt.figure(figsize=(8, 6))
sns.boxplot(x='log_price', data=listings_cleaned)
plt.title('Log-Transformed Price Distribution of Accommodations')
plt.xlabel('Log-Transformed Price')
plt.show()
```



```
listings_no_outliers = listings_updated[listings_updated['price'] < 1000].head(100)
sns.boxplot( x=listings_no_outliers["bedrooms"], y=listings_no_outliers["price"], color='orangered');
plt.title('Price Distribution of First 100 Accommodations (Excluding Outliers)')
plt.xlabel('No of Bedrooms')
plt.xticks(rotation=90)
plt.show()
```



4) Accommodation distribution on maps

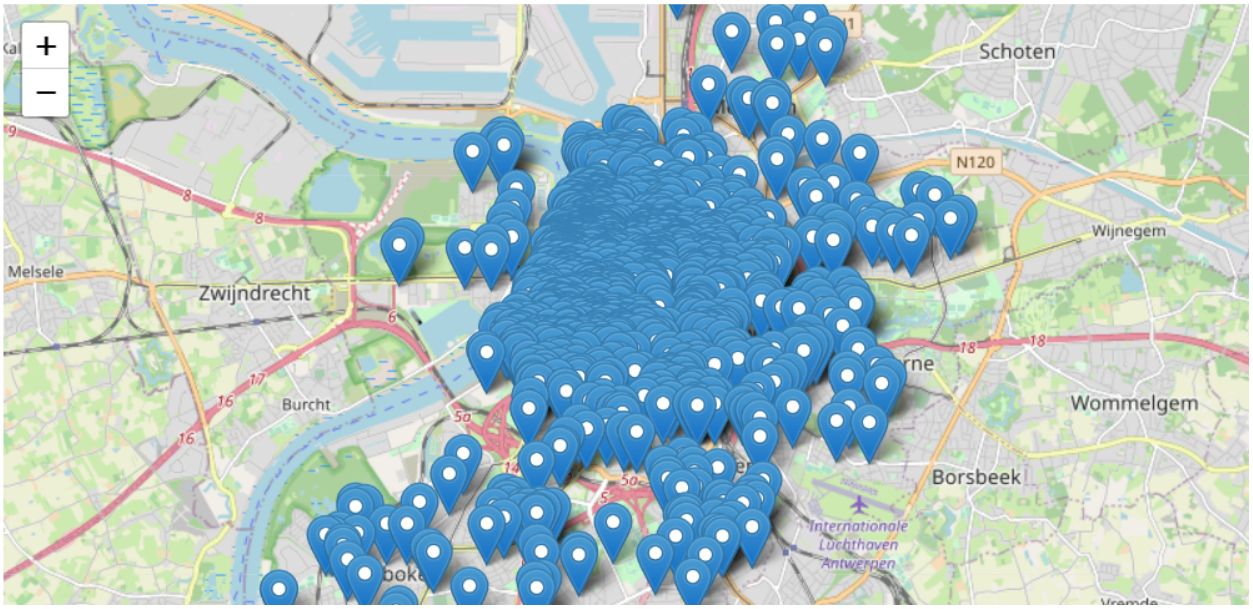
- Plot accommodation based on longitude and latitude value

```
mean_latitude = listings_cleaned['latitude'].mean()
mean_longitude = listings_cleaned['longitude'].mean()

# Create a map
mp = folium.Map(location=[mean_latitude, mean_longitude], zoom_start=10)

# Add markers for each accommodation
for index, row in listings_cleaned.iterrows():
    folium.Marker([row['latitude'], row['longitude']], popup=row['name']).add_to(mp)

# Save or display the map
mp.save('accommodation_distribution_map.html')
```

Above Code has created an HTML file in my project directory, I will display the result using Iframe

```
from IPython.display import IFrame
html_path = 'accommodation_distribution_map.html'
width = 800
height = 600
# Display the HTML page in the notebook
IFrame(html_path, width=width, height=height)
```

```
IFrame(html_path, width=width, height=height)
```



5) Summarize the number of accommodations in each market/ each region

```
accommodations_by_region = listings_cleaned.groupby('neighbourhood')['id'].count()
print(accommodations_by_region)
```

```
neighbourhood
2018 Antwerpen, Belgium                    1
Antwerp, Belgium                          10
Antwerp, Flanders, Belgium                27
Antwerp, Flemish Region, Belgium          11
Antwerp, Vlaams Gewest, Belgium            1
Antwerp, Vlaanderen, Belgium               1
Antwerpen, Antwerp, Belgium                1
Antwerpen, Belgium                        18
Antwerpen, Flanders, Belgium               1
Antwerpen, Flanders/Vlaanderen, Belgium    1
Antwerpen, Vlaams Gewest, Belgium        718
Antwerpen, Vlaanderen, Belgium           653
Berchem, Belgium                           1
Borgerhout, Vlaanderen, Belgium            1
Brasschaat, Vlaams Gewest, Belgium         1
Ekeren , Vlaanderen, Belgium               1
Name: id, dtype: int64
```

6) Summarize the mean price of accommodations in each market/ each region

```
mean_price_of_region = listings_cleaned.groupby('neighbourhood')['price'].mean()
print(mean_price_of_region)
```

```
neighbourhood
2018 Antwerpen, Belgium                    79.000000
Antwerp, Belgium                          219.400000
Antwerp, Flanders, Belgium                123.481481
Antwerp, Flemish Region, Belgium          144.727273
Antwerp, Vlaams Gewest, Belgium           311.000000
Antwerp, Vlaanderen, Belgium              140.000000
Antwerpen, Antwerp, Belgium                56.000000
Antwerpen, Belgium                        113.722222
Antwerpen, Flanders, Belgium               45.000000
Antwerpen, Flanders/Vlaanderen, Belgium    79.000000
Antwerpen, Vlaams Gewest, Belgium         122.736769
Antwerpen, Vlaanderen, Belgium            110.272588
Berchem, Belgium                          146.000000
Borgerhout, Vlaanderen, Belgium            91.000000
Brasschaat, Vlaams Gewest, Belgium        131.000000
Ekeren , Vlaanderen, Belgium               50.000000
Name: price, dtype: float64
```

Task 3: Building the Accommodation Prediction Model

DEAL WITH NON-NUMERIC DATA

```
# print(listings_cleaned.dtypes)
columns_to_drop = ['name', 'description', 'amenities']
numeric_listings = listings_cleaned.drop(columns=columns_to_drop)
print(numeric_listings.neighbourhood.unique())
```

```
['Antwerpen, Vlaams Gewest, Belgium' 'Antwerpen, Vlaanderen, Belgium'
 'Antwerp, Flemish Region, Belgium' 'Antwerp, Flanders, Belgium'
 'Antwerpen, Belgium' 'Antwerp, Belgium' '2018 Antwerpen, Belgium'
 'Borgerhout, Vlaanderen, Belgium' 'Antwerp, Vlaams Gewest, Belgium'
 'Ekeren , Vlaanderen, Belgium' 'Antwerpen, Flanders/Vlaanderen, Belgium'
 'Antwerpen, Flanders, Belgium' 'Antwerp, Vlaanderen, Belgium'
 'Antwerpen, Antwerp, Belgium' 'Berchem, Belgium'
 'Brasschaat, Vlaams Gewest, Belgium']
```

Encode categorical columns

neighbourhood', 'property_type', 'room_type' columns appears to be valuable to predict price of the property. So, we will convert those columns to numeric data using get_dummies(). This function adds columns/features to the daatset based on the unique values.

```
print("Shape before encoding: ", numeric_listings.shape)
```

```
Shape before encoding:  (1447, 13)
```

```
numeric_listings = pd.get_dummies(numeric_listings, columns=['neighbourhood', 'property_type', 'room_type'], drop_first=True)
```

```
print("Shape after encoding: ", numeric_listings.shape)
```

```
Shape after encoding:  (1447, 54)
```

7) Choose a supervised model such as Xgboost, ANNs and other models to implement your price predictor

First we will split the features and target column

```
# print(numeric_listings.columns)
```

```
columns_to_exclude = ['id', 'latitude', 'longitude' ,'price', 'log_price']
X = numeric_listings.drop(columns=columns_to_exclude)
Y = numeric_listings['price']
```

```
X.columns
```

Split the data to training and testing sets

Let's keep 80% of data for training and rest 20% for testing, as mentioned in the requirement

```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2)
```

```
print("Original data (rows, columns): ", X.shape)
print("Training data (rows, columns): ", X_train.shape)
print("Testing data (rows, columns): ", X_test.shape)
```

```
Original data (rows, columns):  (1447, 49)
Training data (rows, columns):  (1157, 49)
Testing data (rows, columns):  (290, 49)
```

In order to make predictive model on the given dataset using supervised model. We need to remove or convert all the columns, those are having string values in them. In simple words, machine learning model will accept numeric data only

CREATE AND TRAIN THE MODEL

```
model = XGBRegressor()
model.fit(X_train, y_train)
```

XGBRegressor

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=None, n_jobs=None,
             num_parallel_tree=None, random_state=None, ...)
```

MODEL VALIDATION

```
from sklearn.metrics import r2_score

# Predictions on training set
y_train_pred = model.predict(X_train)
# Calculate R^2 score for training set
r2_train = r2_score(y_train, y_train_pred)
print(f"Training R^2 Score: {r2_train}")
```

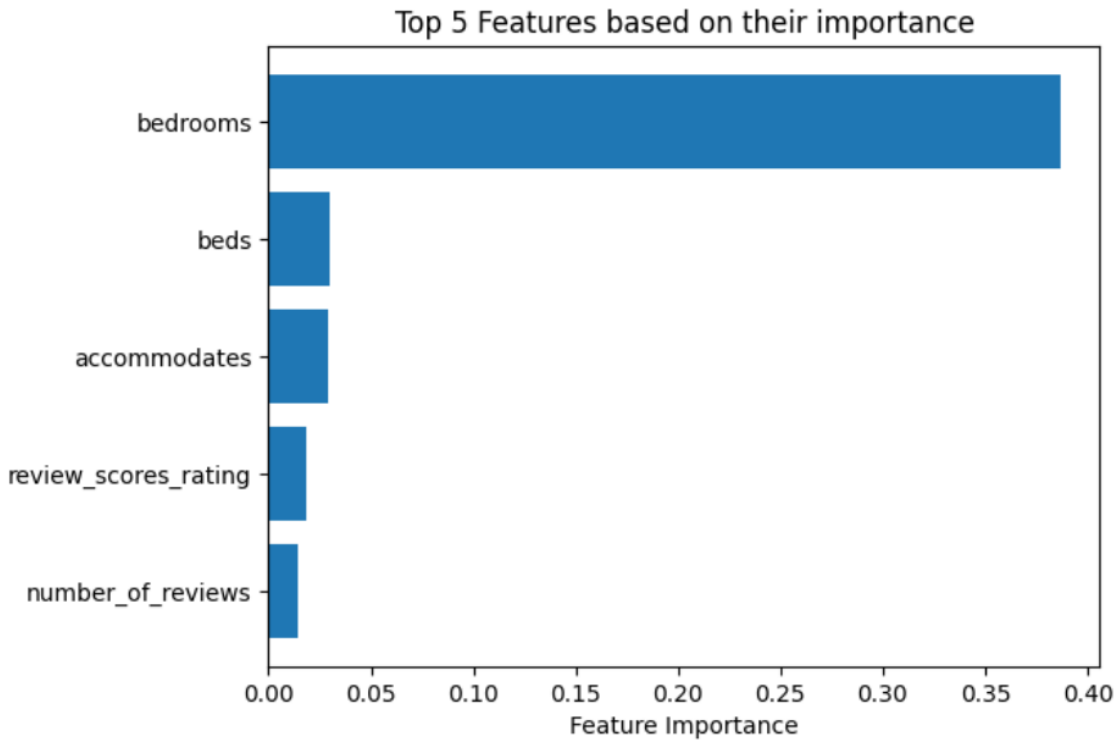
```
Training R^2 Score: 0.9518635310154441
```

8) Perform an analysis to discuss what kinds features are most related to the accommodation price

```
feature_importance = model.feature_importances_
print(feature_importance)
```

```
[2.91528329e-02 3.86711925e-01 2.95883864e-02 1.45985810e-02
 1.87501013e-02 1.40559420e-01 4.42736875e-03 1.17669962e-02
 0.00000000e+00 0.00000000e+00 1.48933730e-03 7.02176243e-04
 0.00000000e+00 1.44165882e-04 1.17719127e-02 3.10853310e-03
 2.69789225e-03 1.19819095e-04 0.00000000e+00 0.00000000e+00
 3.42064211e-03 1.00978429e-03 3.79977253e-04 1.78301670e-02
 1.14767579e-02 3.70103342e-04 6.16802461e-02 7.47002289e-02
 2.40542460e-02 7.79483991e-04 0.00000000e+00 2.23048570e-04
 5.61040302e-04 2.37602182e-03 1.58305746e-04 2.84568546e-03
 0.00000000e+00 2.03161663e-03 3.43598425e-03 1.64642814e-03
 1.79649647e-02 1.02554599e-03 3.56345763e-03 5.12497984e-02
 2.81222146e-02 3.11267358e-04 1.23739592e-03 1.36485836e-02
 1.83075462e-02]
```

```
# Sort the features based on importance
indices = np.argsort(feature_importance[:5])
feature_names = X_train.columns[:5]
# Plotting
plt.figure(figsize=(10, 6))
plt.barh(range(len(indices)), feature_importance[indices], align='center')
plt.yticks(range(len(indices)), [feature_names[i] for i in indices])
plt.xlabel('Feature Importance')
plt.title('Top 5 Features based on their importance')
plt.show()
```



Task 4（Advanced tasks)： Sentiment analysis

```
reviews = pd.read_csv('reviews.csv')
reviews.head()
```

	listing_id	id	date	reviewer_id	reviewer_name	comments
0	50904	31511792	2015-05-06	19482395	Jihae	Karin's "Aplace" is absolutely beautiful and c...
1	50904	470101024356869935	2021-10-10	333559	Emilie	Karin is a wonderful host, she was really help...
2	50904	627287279025726941	2022-05-15	32701854	Marie-Lou	The location is super super nice! Karin was al...
3	224682	933043	2012-02-20	1422043	Hiske & Erik	Perfect location for exploring the city, close...
4	224682	970457	2012-03-05	1493171	Paolo	Muriel was such a fantastic host, extremely he...

```
import nltk
nltk.download('vader_lexicon')
```

9) Perform the sentiment analysis for the review comments

```
from nltk.sentiment import SentimentIntensityAnalyzer
sia = SentimentIntensityAnalyzer()
```

```
# Define a function to calculate sentiment score
def calculate_sentiment_score(comment):
    if isinstance(comment, str):
        return sia.polarity_scores(comment)['compound']
    else:
        return None  # we can use any default value for non-string values

# Apply the sentiment analysis function to the 'comments' column
reviews['sentiment_score'] = reviews['comments'].apply(calculate_sentiment_score)
```

```
print(reviews.tail(2))
```

```
              listing_id                   id        date  reviewer_id  \
97123  975154238474532959  977398882312709986  2023-09-10    384546438
97124  977985082084304366  981701740613077893  2023-09-16    210133725

      reviewer_name                                          comments  \
97123          Sara                                Everything was good 👍🏻
97124           Liv  Die Lage der Wohnung war perfekt für unseren S...

       sentiment_score
97123           0.4404
97124          -0.9136
```

```
positive_reviews = reviews[reviews['sentiment_score'] > 0]
negative_reviews = reviews[reviews['sentiment_score'] < 0]
```

```
print(positive_reviews.comments[:4])
```

```
0    Karin’s “Aplace” is absolutely beautiful and c...
1    Karin is a wonderful host, she was really help...
2    The location is super super nice! Karin was al...
3    Perfect location for exploring the city, close...
Name: comments, dtype: object
```

```
print(negative_reviews.comments[:4])
```

```
29    Sehr schönes Apartement, sehr zentral und gut ...
30    Die Wohnung ist totel niedlich und supersauber...
62    Leuk appartement aan plein in het hartje van A...
71    Miertje hat lange auf uns warten müssen, weil ...
Name: comments, dtype: object
```

```
print("Number of positive reviews/comments: ", len(positive_reviews))
print("Number of negative reviews/comments: ", len(negative_reviews))
```

```
Number of positive reviews/comments:  60065
Number of negative reviews/comments:  9434
```

Discussion and Error analysis

It is clear from evaluating the analysis's findings that every task had its own special difficulties and subtleties. The conversation that follows includes observations made based on assumptions, theories formed from visual inspections, and an analysis of difficulties faced throughout the endeavor.

Task 1: Pre-processing of data:

After cleaning, there is a noticeable influence on the dataset when the distribution of data is visualized. The elimination of insignificant columns was predicated on the belief that they were not significant; nonetheless, it is important to recognize that relevance is a subjective concept. Finding a balance between possible information loss and dimensionality reduction is the difficult part. If not properly managed, price outliers may distort the results of later analyses.

Task 2: Exploratory Data Analysis (EDA):

The boxplot illustrating the pricing distribution of accommodations provides information about possible outliers that could skew projections. The notion that excessive costs are abnormal should be reexamined because they may indicate special, high-quality products. Maps with spatial analysis show clusters of lodging, but assumptions about the consistent effect of location on costs should be carefully examined, particularly in regions with diverse topographies.

Task 3: Accommodation Prediction Model:

The use of ANNs and XGBoost for prediction is predicated on the assumption that these models can capture complex relationships in the data. Although the assumption of linearity may not always hold true, visualising the relevance of features might give insights into the variables impacting prices. A challenge is in adjusting hyperparameters to prevent overfitting and guaranteeing the resilience of the model on a variety of datasets.

Task 4: Sentiment Analysis:

The goal of sentiment analysis on review comments is to identify the qualitative factors that influence customer happiness. On the other hand, biases can result from assuming that sentiment dictionaries are universal, which presents difficulties when interpreting complex emotions. The subjective nature of the reasoning behind likes and dislikes is still up for debate, thus it needs to be verified against other sources.

Assumptions and Potential Biases:

Potential biases are introduced by presumptions regarding spatial homogeneity, currency uniformity, and column importance. Different viewpoints of relevance could affect the results of the model. Currency conversion may ignore regional nuances in favor of uniform pricing schemes. Due to the assumption of a uniform impact of location, spatial analysis may overlook localized factors.

Issues Affecting Model Performance:

Model training is complicated by missing values, outliers, and unbalanced data. To maximize model performance, hyperparameter adjustment is crucial, and it's important to watch out for overfitting. The validity of the results could be impacted by uncertainties introduced by the sentiment analysis tool's accuracy and the sentiment dictionaries chosen.

Challenge and problem during project

1. Data Cleaning and Pre-processing Balancing Act:
    a. Problem: During data pre-processing, finding the ideal balance between noise reduction and information retention remained difficult. Subjective decisions were

made while deciding which columns to keep or reject, which could have resulted in the loss of information.

   b. Solution: This problem was lessened by implementing iterative cleaning, soliciting stakeholder input, and investigating different approaches to addressing missing values.

2. Unbalanced Training Set Data:
   a. Problem: Unbalances in the training dataset made it difficult for the model to generalize to different situations. A possibility of biassed predictions existed due to the underrepresentation of certain classes or traits.
   b. Solution: In order to rectify imbalances and improve the model's capacity to learn from all pertinent patterns, methods including oversampling, undersampling, or the use of synthetic data generation were employed.
3. Model Selection and Adjustment of Hyperparameters:
   a. Difficulty: XGBoost and ANNs were the predictive models chosen in this instance, and there were hyperparameter tuning difficulties. Iterative modifications were necessary to find the ideal configuration and address overfitting and underfitting[3].
   b. Solution: Using grid search and cross-validation as part of a methodical hyperparameter tweaking technique made it easier to find configurations that improved model performance.
4. Assumptions About Spatial Analysis:
   a. Problem: Analysing the distribution of accommodations on maps under the assumption of geographical homogeneity may oversimplify the impact of location on costs. Ignoring localized issues could cause projections to be off.
   b. Answer: Refining the geographical understanding of lodging pricing involved carrying out more detailed spatial analyses and taking into account the integration of external location-specific variables.

5. Sentiment analysis through the subjective lens:
    a. The challenge was the introduction of subjectivity through sentiment analysis of review comments. Accurately assessing consumer happiness was difficult due to context-specific language, sarcasm, and nuanced attitudes.
    b. Response: To mitigate subjectivity concerns, sentiment analysis technologies with contextual awareness were used, along with multiple sentiment dictionaries validation and sentiment intensity metrics integration.
6. Sentiment analysis's generalizability:
    a. Problem: Presuming sentiment analysis results can be applied to a wide range of situations, cultures, and demographics may result in skewed conclusions.
    b. Resolution: By acknowledging the constraints of sentiment analysis and performing sensitivity tests on various data subsets or demographic groups, the robustness of the sentiment-related conclusions was increased.
7. Preferences for accommodations are open-ended:
    a. Challenge: The intrinsically subjective nature of user preferences was exposed by analyzing reasons for likes and dislikes based on open-ended questions. It made it challenging to classify and measure qualitative answers.
    b. Solution: The reliability of the qualitative analysis was increased by using techniques from qualitative research, such as theme analysis, to identify patterns in the open-ended responses and by being open about the subjective character of the results.

Conclusion

The project has successfully negotiated complex hurdles and uncovered insightful information regarding the influence of Airbnb on the hospitality scene through data warehousing and mining. We have improved our forecasts of lodging prices and explored the subtleties of sentiment analysis by analyzing the Barwon South West dataset. A careful dance between data preservation and noise reduction, addressing imbalances, honing prediction models, and facing the subjectivity inherent in human emotions were all part of the trip. The study is significant not only

because of its ability to forecast outcomes but also because it recognizes the dynamic interaction between data and human experience. In the age of data-driven decision-making, a more sophisticated, flexible approach to comprehending and improving the guest experience is made possible by the lessons gained and techniques improved here, especially as the hospitality sector develops.

References

[1]C. Albon, *Machine Learning with Python Cookbook: Practical Solutions from Preprocessing to Deep Learning*. "O'Reilly Media, Inc.," 2018. Accessed: Nov. 20, 2023. [Online]. Available: https://www.google.com.au/books/edition/Machine_Learning_with_Python_Cookbook/kIhQDwAAQBAJ?hl=en&gbpv=1&dq=data+preprocessing+in+python&pg=PT107&printsec=frontcover

[2]Gupta, B., Negi, M., Vishwakarma, K., Rawat, G. and Badhani, P. (2017). Study of Twitter Sentiment Analysis using Machine Learning Algorithms on Python. International Journal of Computer Applications, 165(9), pp.29–34. doi:https://doi.org/10.5120/ijca2017914022.

[3]A. Mavuduru, "Why XGBoost can't solve all your problems.," *Medium*, Nov. 10, 2020. https://towardsdatascience.com/why-xgboost-cant-solve-all-your-problems-b5003a62d12a

.