BIG DATA PROJECT: REPORT

F743: Big Data In Finance

Presented to: Adeel Mahmood

Name	Student ID
Abhinav Bhatnagar	400074769
Antony Britto	400360102
Francis David	400406567
Maunica Nemalipuri	400413352
Vladislav Perevalov	400422374
Yidi Geng	400423487

Table of Contents

Problem Description

Airbnb is a global public company that provides booking services and travel information. Customers can book lodging, homestays, and tourism services via their app. The company went public in late 2020, and its stock has had a volatile journey over the last 2 years. Over the last year, they have lost 50% of their share value due to the various problems that they are facing.

When Airbnb first launched, they were seen as a more cost-effective alternative to traditional hotel booking, allowing people to travel worldwide on a budget. But over the past few years, one can see that the gap between Airbnb and Hotels has been closing. This gap has significantly narrowed during the pandemic. While the 2 services may provide different experiences at times, the small gap in pricing between a professionally run hotel in downtown Toronto and a condo owned by someone who doesn't reside in the area shouldn't be this narrow.

The other issue short-term rentals face is the growing recession and the lack of bookings ahead of the holidays. Rising interest rates and inflation worry owners about the properties they acquired during the pandemic and the mortgages they need to pay. When a host prices an Airbnb property, all Airbnb provides is the ability to search similar listings in the area to gauge what the host should charge. Our team believes that such an approach does not consider the unique types of properties, levels or services, or experiences Airbnb has to offer. One part of Airbnb's unique differentiation is its ability to offer lower prices than hotels while still providing high-quality and professional stays. We believe that Airbnb's lack of resources for its hosts for pricing and the slow process of designating new "Superhosts" is damaging their business. Mispricing can lead to people booking with Airbnb's competitors. Optimal pricing is imperative, especially during a time of economic uncertainty and the opening of travel routes once again.

Meanwhile, the limited number of reviews of Superhosts will make new customers less willing to use the service. If this problem is not resolved, Airbnb's share price will likely continue to decline quarter over quarter as they struggle to keep up with the fast pace of the travel and tourism industry. Our team believes that Airbnb would benefit from an in-depth analysis of its recent listing activity in downtown Toronto. Unsupervised machine learning techniques will be used to analyze trends relating to the number of reviews, quality of listings, location, amenities, host status, price, and other variables. Supervised machine learning will be used to create a model to suggest pricing to hosts for new listings based on historical data. A model will also be designed to automate the process of suggesting new Superhosts for Airbnb.

Solution Summary

Airbnb started as a solution for travelers looking for a place to stay the night without paying the expensive price tag of a hotel. As the prices of Airbnb listings have risen since then, travelers are seeing less difference between the two, thus causing Airbnb to lose its competitive edge. Currently, there are only suggestions on how listers should price their property, causing a vast variation in the pricing strategy followed by Airbnb. This pricing strategy doesn't follow clear guidance on which factors drive up Airbnb's prices: is it dependent on the location? The amenities? The number of beds? The response time and rating of the host? Our solution is to develop a predictive pricing model that looks at historical listings of Airbnb in Toronto, finding trends between their price and different features such as ratings, amenities, bedrooms, and location. With historical data analyzed and displaying the elements that drive up prices, hosts can determine where their listings are best positioned based on price, allowing them to be more competitive in the market. Hosts and Airbnb revenue are expected to increase through better decision-making based on big data.

The team will first begin by conducting data cleaning on over 16,000 Toronto Airbnb listings scraped from the web in September 2022. We will then use matplotlib, seaborn, and other statistical and visualization modules to look at trends and provide recommendations based on the data. Finally, we will create two learning models, one to predict price and one to predict Superhost status for new listings. We expect that the model developed can be used by Airbnb for more contemporary hosts. Airbnb will better understand the property features that result in higher value through the model. Meanwhile, hosts will be given more information on how to list their properties.

Solution Details

Assumptions Used or Made

The dataset we used was sourced from Kaggle, an online data hub. The dataset contained 16,035 rows and 75 columns. Going through the dataset, we identified that of the 75, only 37 contained useful data for our analysis.

Data Gathering and Import, Including the Specific Files and Format, Used

The file name present online was Listings_Details.csv, which was later changed to Raw_AirbnbData.csv for the Data Clean-up process. The file was then imported into Jupyter Notebook using the **pd.read_csv** command. To test the successful import of the file, we used **df.head(10)** which should show the first 10 rows of the file.

Data Cleanup, Transformation, and Normalization

The data cleanup process started with dropping the 38 columns identified in step 1 using **df.drop()**. We then checked for any NULL values in the existing columns using **df.isnull().sum(),** which lists all the columns with NULL values. We identified that 'columns host since', 'review scores rating', 'review scores accuracy', 'review scores cleanliness', 'review scores checkin', 'review scores communication', 'review scores location', 'review scores value' had a significant number of NULL values. So we dropped these rows using the command **df.dropna()**, which dropped NULL values in only those columns. To validate the success of this command, **df.isnull ().sum()** was used as before.

The column 'host response time' had a few NULL values. By using **df['host_response_time'].replace(np.NaN, "no response"),** the NULL values were replaced with No response. To validate the above command, **df.host_response_time.unique()** was used, which displays the unique values contained in those columns. 'Host response rate' was the next column which contained NULL values. This column was an 'object' data type, so we needed to convert it to a 'float' data type and strip the % sign to replace the NULL values with 0 successfully. We then used **df['host_response_rate'].str.rstrip("%").astype(float)/100** to strip the % and converted the column to float, and divided all the values by 100. We then used **df['host_response_rate'].replace(np.NaN,0)** to replace the NULL values with 0. To validate the successful replacement of 0 **df.host_response_rate.unique()** was used. The same steps were then used on the 'host acceptance rate' column to replace NULL values with 0.

The next column with NULL values was 'host total listings count'. Unlike the other 2 columns, this column was a float data type. So we could directly use **replace (np.NaN,1)** to replace the NULL values with the minimum value of 1. To validate the step, **df.host_total_listings_count.unique()** was used to display the unique values.

The next column with NULL values was 'bathrooms'. In this column, we first filled the NULL values with 0 using **df['bathrooms'].fillna('0')**. Since this column contained string values and could not be used for calculation, the string values were converted to lowercase using **df['bathrooms'].str.lower()** followed by **df['bathrooms'].str.rstrip("shared half-bath private")** to strip the string values. To convert it into float **df['bathrooms'].apply(pd.to_numeric).** It was observed that ' ' was present in the data, which was attributed to removing the "half-shared" string values. To fix this issue, we decided to replace it with the minimum half values, i.e., 1.5, by using **df['bathrooms'].fillna(1.5)**. We then used **df['bathrooms'].unique()** to check the various values in the column.

The next 2 columns that contained NULL values were ‘beds’ and ‘bedrooms’. While going through these columns, it was observed that there were rows where both these columns contained NULL values. We considered that these 24 rows didn’t contain useful data for our analysis and dropped them using **df.drop(df[(df.beds==0) & (df.bedrooms==0)].index)**.

The remaining rows were not dropped as they contained useful information. Using linear regression, we used columns ‘price’ and ‘accommodates’ to predict the missing values ‘bedrooms’. Test and Training data were created by creating a new function newdf that contained the values of the 3 columns by using **new_df =df[[‘price’,’ accommodates’, “bedrooms”]].copy().**

Next test data was created by using **test_data = new_df[new_df["bedrooms"].isnull()]** train data using **train_data = new_df.dropna().** X-Train data was then created by dropping ‘beds’ using **X_train = train_data.drop("bedrooms", axis = 1)**. Y-Train only contains ‘bedrooms’ by using **Y_train = train_data[“bedrooms”]**. The X & Y-Test was created using the same logic. The linear regression package was then imported using **from sklearn import linear_model**. The X &Y-Train datasets were then trained using the LINREG package: **LINREG = linear_model.LinearRegression()** followed by **LINREG.fit(X_train,Y_train)**. After training the data, we then predicted the X-test value (Bedrooms) using the training data using **y_predict = LINREG.predict(X_test).** The predicted values were rounded and saved to another variable, **predicted_bedrooms,** by using **y_predictrounded = y_predict.round(0)** followed by **X_test[“predicted_bedrooms”] = y_predictrounded.** The predicted values were then substituted to the NULL values in the ‘bedrooms’ column using **df[“bedrooms”].fillna(X_test[“predicted_bedrooms”], inplace=True).** The unique values were validated using **df.bedrooms.unique().** The same steps were followed for predicting the NULL values of the ‘beds’ column.

The next column to modify was ‘accommodates’, where we replaced 0 with the minimum value 1 by using **df[‘accommodates’]=df[‘accommodates’].replace(0,1)** and validated it by using **df.accommodates.unique().**

Following this, the column ‘price’ had to be cleaned by converting it to type float and dropping the $ sign. This was done by using **df['price']=df['price'].str.replace(',', '', regex=True)** followed by **df['price']=df['price'].str.replace('$','',regex=True), df["price"] = df['price'].apply(pd.to_numeric)** which changes the data type to float. This can be checked by **df.price.dtypes.**

Multiple columns contained values True and False but had an object data type. These columns need to be converted to a ‘Boolean’ data type. The steps to achieve this can be demonstrated using the ‘host is superhost’ column. The first step was to replace the ‘t’ and ‘f’ values with True and False. This was done

by using **dictionary_replace = {'t': True, 'f': False}**. The next step was assigning this variable to the column and completing the replacement process. This was done by using **boolean_host_has_profile_pic = df['host_has_profile_pic'].map(dictionary_replace)** followed by **df["host_has_profile_pic"] = boolean_host_has_profile_pic.** To check the replacement process, we used **df.host_is_superhost.unique()**. The same steps were followed for column 'instant bookable'. The same steps were followed for column 'host has profile pic', which contained NULL values. These were replaced with 'False' using **df['host_has_profile_pic'] = df['host_has_profile_pic'].replace(np.nan,False)**. The same steps were followed for the 'host identity verified' column, but we replaced the NULL values with 'None' using **df[host_identity_verified] = df[host_identity_verified].replace(np.nan, "None").**

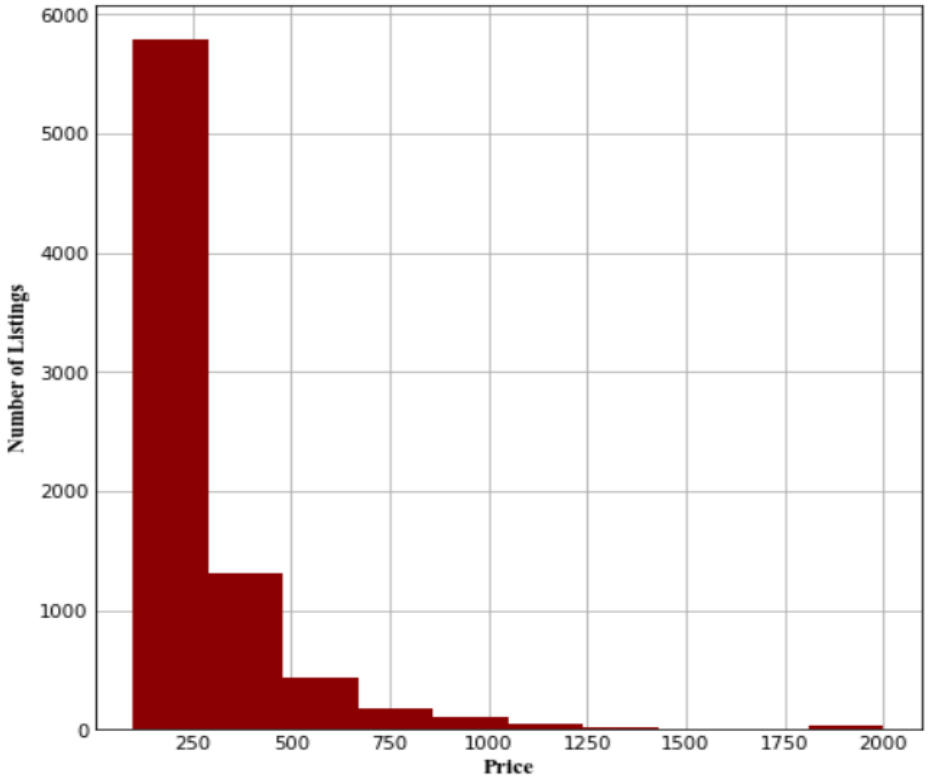
The next column to transform was 'amenities', which was a list and could not be used for calculation purposes. To make it useful, the first step was to convert it into a string by using **df['amenities'] = df['amenities'].astype('string').** After this, we converted the list inputs into a countable format using **df["amenities"] = df['amenities'].apply(eval)** followed by **for i, l in enumerate(df["amenities"]): print("list", i, "is", type(l)).** Next, the individual list values were counted and saved as an integer in the 'amenities count' column using **df['amenities_count']=df["amenities"].apply(lambda x: len(x))**.

The last column to transform was 'host since'; as this was a 'date' data type, we converted it into an 'int' format specifying the no. of days the person has been a host. This was done by **df['host_since'] = pd.to_datetime(df['host_since'])** followed by calculating the no. of days and saving it to 'host experience days' by **df['host_experience_days'] = (df['host_since'] - df['host_since'].min())/ np.timedelta64(1, 'D').** The 'host since' and 'amenities' columns were then dropped to avoid confusion.

As a final check, **result = df_final.dtypes print(result)** and **df.shape** were run to determine the no. of columns and data types of the modified columns.

Results

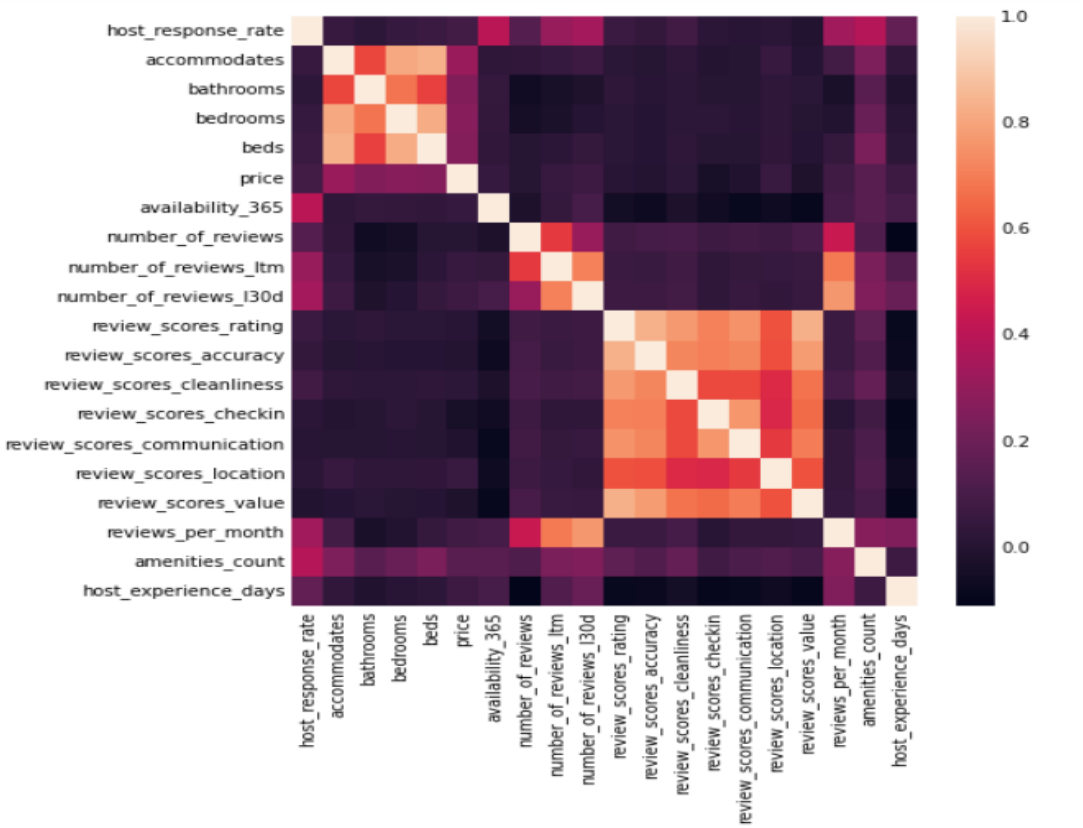
Toronto Airbnb Price Histogram




```
plt.figure(figsize=(6,6))
plt.hist(df["price"], range=[100, 2000], facecolor='darkred', align='mid')
plt.title("Toronto Airbnb Price Distribution", fontsize = 15, fontweight="bold", fontname="Times New Roman")
plt.xlabel("Price", fontsize = 12, fontweight="bold", fontname="Times New Roman")
plt.ylabel("Number of Listings", fontsize = 12, fontweight="bold", fontname="Times New Roman")
plt.show()
```

Our team used matplotlib to create a histogram for the prices of Airbnbs. Here we can see that prices range between $100 to $2,000. From the analysis, our group has concluded that most of the listings are between the $100 to $250 range. The probability of seeing a list over $750 is relatively low, and the histogram demonstrates a right skewed pattern. Additionally, we have observed a few listings classified in the price range between $1,980 to $2,000, which we may want to count as outliers for further analysis. Overall, this analysis shows that most the listings in our predictive model should fall within a certain range and that there may be some data of overly priced locations that might skew our machine learning model. We are interested in comparing the features of a very expensive Airbnb in the $1000 plus range to the median.

Qualitative Variable - Heatmap



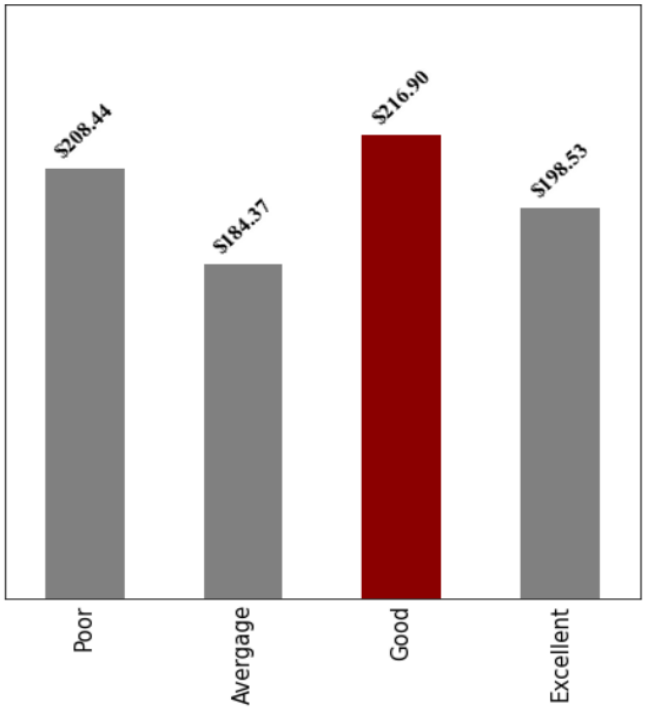

```
stats_df = df.drop(["host_response_time","availability_30","availability_60","availability_90","host_acceptance_rate","
corr = stats_df.corr()

# Heatmap
plt.figure(figsize=(9,8))
sns.heatmap(corr)
```

Using seaborn, our team created a heatmap of all the quantitative variables in our data. As per the correlation heatmap above, we wanted to determine if there are strong correlations between various variables that affect the price variable. As can be seen, the strongest correlation with price occurs with accommodations, bedrooms, and baths. The number of people the location can service is a key factor in how Airbnb hosts price their properties. Hosts might see the potential for splitting the cost between all the people staying at their property simultaneously to justify the higher property price. The overall largest correlations between independent variables can be seen between the various review factors available such as accuracy, cleanliness, location, etc. This correlation tells us that when reviewing, people are more likely to give high/low ratings across multiple categories rather than differentiate them. The customer mindset is that if the overall experience was good, then individual factors all measured up for the most part. The overall rating score correlates with accuracy and value, meaning that if the location meets the customer's

expectation based on the description and their perceived value paid, they are more likely to give a high score.

We can discern from this that accurate photos of the listings, accurate descriptions of what is available, and pricing that truly reflects the value of the property will lead hosts to have higher ratings on their properties overall. Perceived value is, of course, tied to what is currently available in the market. Pricing plays a huge role in demand and keeping up with trends is imperative for consistently high ratings.

Price vs. Rating



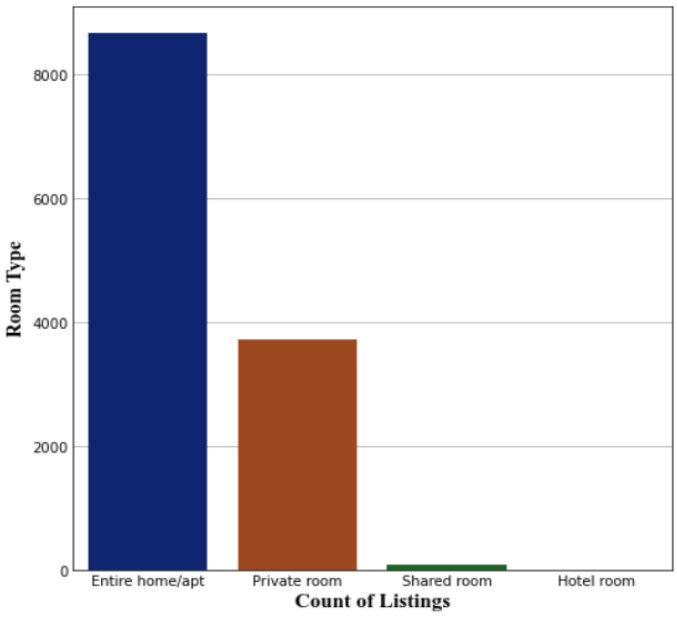

```
categories = pd.cut(df['review_scores_value'], 4,
                    labels = ['Poor', 'Avergage', 'Good', 'Excellent'])

# The 'categories' variable data will become a new column in the DF.
df['rating_category'] = categories
```

```
groups = df.groupby('rating_category')['price'].mean()

chart = groups.plot(kind = 'bar', ylim = (100,250) , xlabel = '', yticks = [],
                    fontsize = 10, grid = False, color = ['grey','grey', 'darkred', 'grey'])

for i, price in enumerate(groups):
    plt.text(-0.2 + i, price + 3, f'${price:.2f}', fontdict = {'size':12})

plt.title('Price by Rating', fontdict = {'size':15})
```

The rating, property size, and location of a property will understandably raise the appeal of listings. However, it is important to determine whether there is a relationship between ratings and prices, as higher ratings do not necessarily equate to higher prices. This is a product of perceived value due to the location

or size of the listing and not because of overall guest satisfaction. Using matplotlib, our team grouped the historical listings based on their reviews as poor, average, good, or excellent, and plotted them based on the average price per group. The results show some discrepancy between the overall ratings of listings and the prices charged. Poorly rated locations were charging higher prices on average compared to locations that were rated as excellent. Interestingly, the most expensive places on average have a good rating but were not excellent. The results that excellent ratings are not the highest priced is a positive sign for Airbnb, as this demonstrates the value this business model offers customers. The firm should further market these excellent listings as a lower prices value option for price sensitive consumers. This point further highlights the needs to identify our "stars" or Superhosts for Airbnb to market.

Types of Airbnb Listings

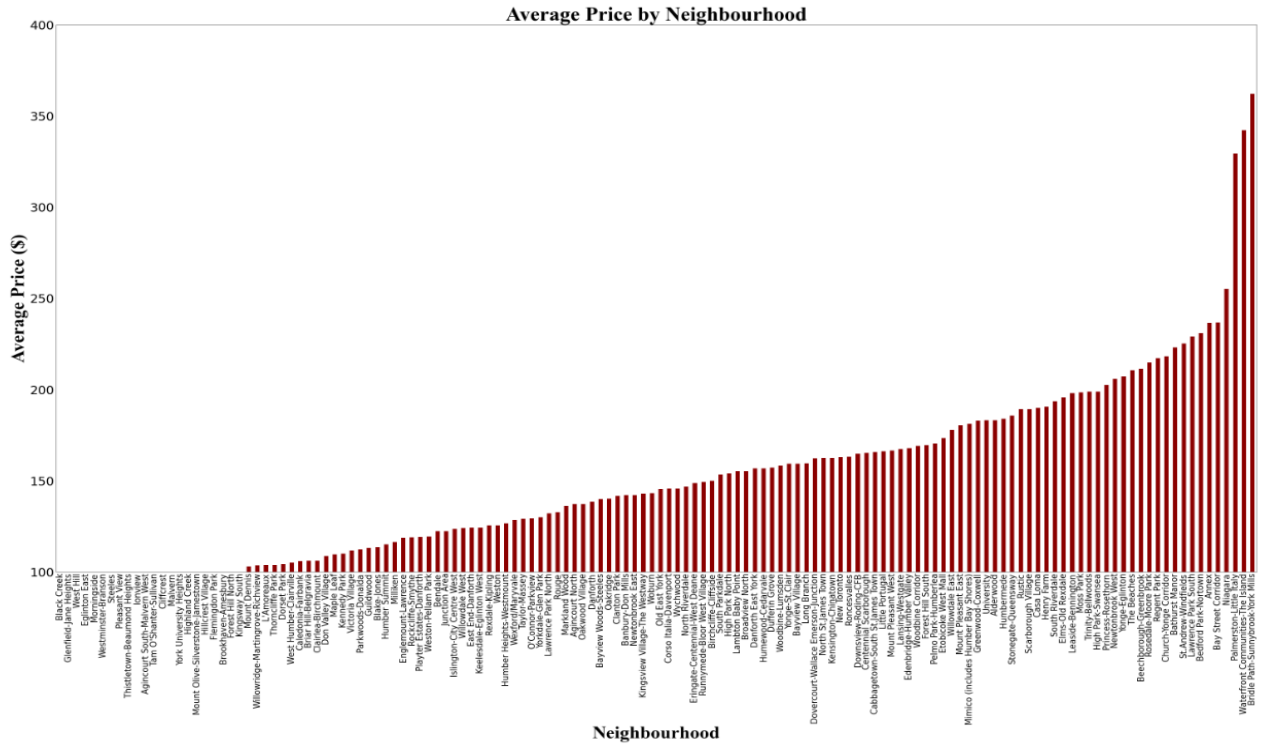



```
plt.figure(figsize=(6,6))
plt.title("Types of Airbnb Listings in Toronto", fontsize = 18, fontweight="bold", fontname="Times New Roman")
sns.countplot(df.room_type, palette="dark")
fig = plt.gcf()
plt.xlabel("Count of Listings", fontsize = 15, fontweight="bold", fontname="Times New Roman")
plt.ylabel("Room Type", fontsize = 15, fontweight="bold", fontname="Times New Roman")
plt.show()
```

Our team used a bar chart to identify the most common types of listings available in Toronto. The common listings are the ones that have the entire home/apartment listed, followed by the places that have at least a private room. This analysis can help Airbnb hosts understand the types of rooms that are in high supply across the city and adjust their listings accordingly. Moreover, hosts tend to give their entire property for Airbnb much more frequently than private rooms. The ability to give guests access to an entire home, including a kitchen and living room is one of the benefits of Airbnb and a competitive advantage when

compared to hotels. For further analysis, it is recommended that Airbnb investigate private rooms and see how they compete against traditional hotels.

Price vs. Neighborhood



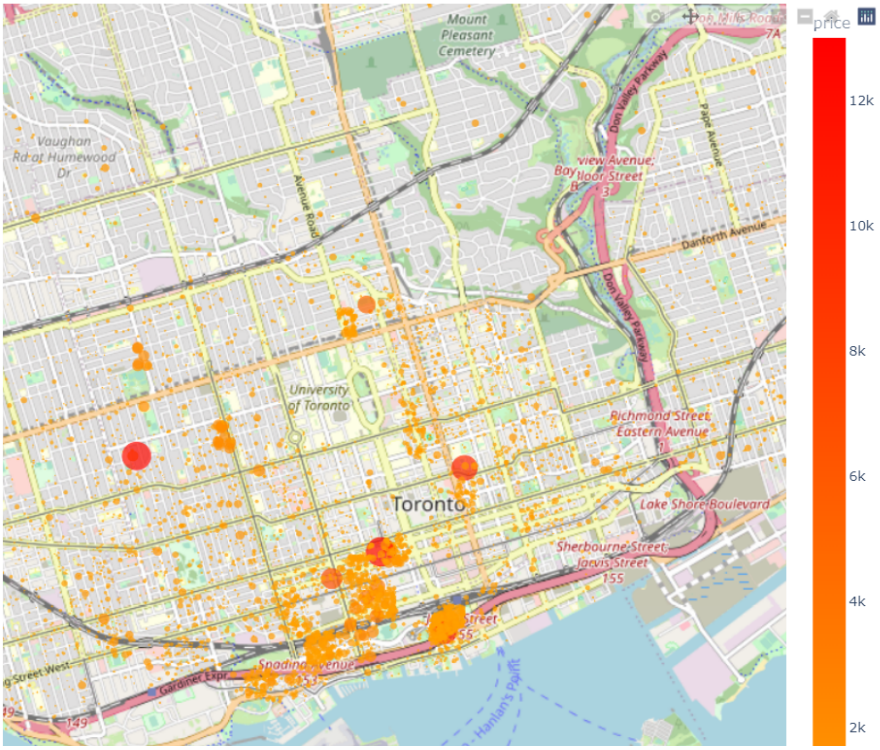

```
groups = df.groupby('neighbourhood_cleansed')['price'].mean().sort_values()
```

```
plt.figure(figsize=(40,20))
a = df.tail(30).reset_index()
chart = groups.plot(kind = 'bar', ylim = (100,400) , xlabel = 'Neighbourhood', grid = False, color = ["darkred"])

plt.title('Average Price by Neighbourhood', fontdict = {'size':50}, fontweight="bold", fontname="Times New Roman")
plt.xlabel("Neighbourhood", fontsize = 45, fontweight="bold", fontname="Times New Roman")
plt.ylabel("Average Price ($)", fontsize = 45, fontweight="bold", fontname="Times New Roman")
plt.tick_params(axis='x', labelsize=20.5)
plt.tick_params(axis='y', labelsize=30)
```

Determining the average price by neighborhood was an important step because location can be a critical factor for driving price. Deciding factors such as proximity to public transport and proximity to most frequent destinations are fading as the ease of booking private transportation has made it easier for people to move around. Additionally, we are seeing a trend of “staycations” where the quality of the property is more important than the location. This is especially true for longer stays such as cabins and cottages. From the historical data, we grouped each listing into specific neighborhoods and produced a bar chart displaying the average price. Based on the data, it was observed that the neighborhoods with the highest average price included Bridle Path, Waterfront Communities, Little Italy, and Niagara. This data

demonstrates that there some relationship between location and prices charged on Airbnb. It is recommended that further analysis using detailed latitude and longitude data is used to pinpoint trends.

Location and Price – Interactive Map



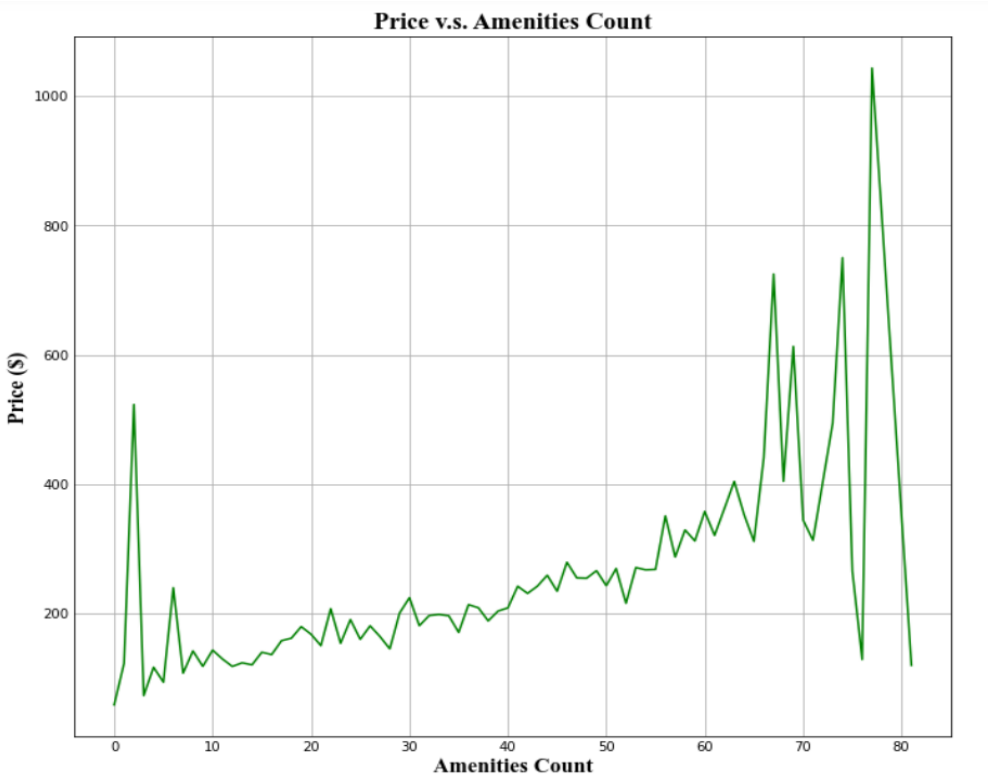

```
import plotly.express as px
color_scale = [(0, 'orange'), (1,'red')]
fig = px.scatter_mapbox(df2,
                        lat="latitude",
                        lon="longitude",
                        hover_name="neighbourhood_cleansed",
                        hover_data=["neighbourhood_cleansed", "price"],
                        color="price",
                        color_continuous_scale=color_scale,
                        size="price",
                        zoom=8,
                        height=800,
                        width=800)

fig.update_layout(mapbox_style="open-street-map")
fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```

Based on longitude, latitude, neighborhood, and price data, our team used plotly and open-street map layouts to create an interactive map for the Toronto listings. Here, each data point's size and color indicate the listing's price. We wanted to know which areas were the most popular and cross-referenced this data with hotels in the area. First, it was clear that there is an Airbnb within walking distance of every part of the city, which is a positive sign. Second, our team observed more condensed clusters around The Financial District, The Entertainment District, Cityplace, Harbourfront, and Niagara. These hotspots make sense as these are often areas where guests like to stay while attending events at large venues in the area. When comparing Airbnb trends to hotels in the area, it was noted that Airbnb has better coverage of

Cityplace and Niagara towards the west of the city. These findings suggest that Airbnb uses a targeted marketing strategy for guests attending events closer to these parts of the city, such as the Budweiser Stage or Exhibition.

Price vs. Number of Amenities

```
plt.figure(figsize = (9,8))
ax = sns.lineplot(x = 'amenities_count', y = 'price', data = df, ci = None, color = 'green')
plt.title("Price v.s. Amenities Count", fontsize = 18, fontweight="bold", fontname="Times New Roman")
plt.xlabel("Amenities Count", fontsize = 16, fontweight="bold", fontname="Times New Roman")
plt.ylabel("Price ($)", fontsize = 16, fontweight="bold", fontname="Times New Roman")
plt.show()
```

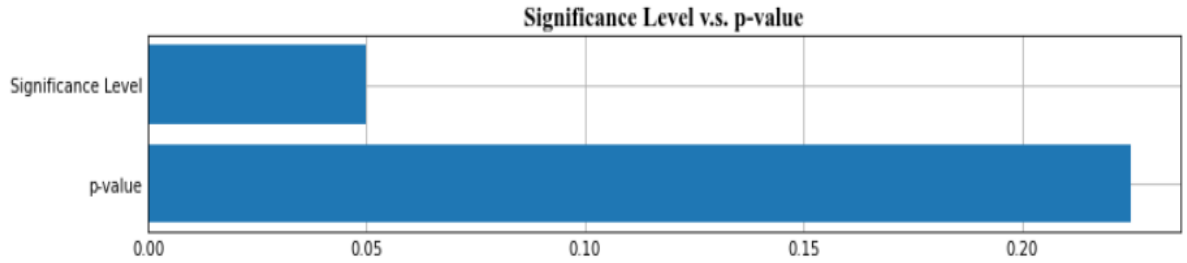


Our team used seaborn lineplot to see the relationship between the number of amenities offered and price. The graph shows a relatively linear trend between price and amenities between the 10-60 amenities range. However, there is more volatility when looking at the tail ends of the graph. The key takeaway from this visualization is that there is some relationship between amenities and price that could be leverage by a machine learning model. However, the team will also need to be careful with places that have too few or too many amenities, as this may skew the data.

Superhosts and Price – Sample t-tests

```
# create two samples with 1000 samples each
superhost = df[df['host_is_superhost'] == True]['price'].sample(1000)
non_superhost = df[df['host_is_superhost'] == False]['price'].sample(1000)

# With the samples, conduct a t test and calculate the p value
from scipy.stats import ttest_rel
pvalue = ttest_rel(superhost, non_superhost)[1]

print(pvalue)
fig = plt.figure(figsize = (10,1.5))
plt.barh(y = ['p-value', 'Significance Level'], width = [pvalue, 0.05])
plt.title('Significance Level v.s. p-value', fontsize = 15, fontweight="bold", fontname="Times New Roman")
```

```
0.22496001780728214
```



Before creating a model to suggest whether a host should be awarded Superhost status, our team was interested in examining the current pricing trends of Superhosts. If Superhosts charge much higher prices than Non-Superhosts, Airbnb may further be at risk of charging prices that are non-competitive compared to hotels. Using the Toronto data, our team used the scripy.stats module to conduct a t-test comparing Superhosts and Non-Superhosts. Using a t-test for the related samples with a significance level of 0.05, we determined that Superhosts do not charge significantly higher prices than Non-Superhosts. This is a positive sign for Airbnb, as awarding the Superhost status to more hosts may increase customers' willingness to pay without increasing prices. In other words, hosts are unlikely to charge more just because they have been awarded this status.

Random Forest Classifier – Superhosts

Our team used a Random Forest Classifier (RFC) model to predict whether a host should be given Superhost status. Random forests are an ensemble learning method for classification that works by constructing multiple decision trees during training. For classification tasks, the output of the random forest is the class selected by most trees. For the x variables, all boolean, integer, and float variables were used. The y variable was the boolean, True or False for Superhost status. Using 100 estimators and the 'Gini' function to measure the quality of a split, we created an RFC model with 85.28% accuracy.

```
In [1]:
#most problems are classification type
# 5 nodes so 5**2 leafs possible
#Two parts to consider, how do we split and what is the order of variables we split in?
#A better decision tree, for each leaf will maximize the average of all the absolute differences of outcomes, 1 and 2

import numpy as np
import pandas as pd
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

In [2]:
df = pd.read_csv("Clean_AirbnbData.csv")

In [3]:
df.head(3)
df.shape

Out[3]: (12496, 37)

In [4]:
X = df.iloc[:,[1,2,4,5,6,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35]].values
X
Y = df.iloc[:,[3]].values
Y

Out[4]: array([[ True],
               [False],
               [False],
               ...,
               [False],
               [False],
               [ True]])

In [5]:
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size = 0.3,random_state = 100)
```

The first step of the coding process was to import the relevant libraries like NumPy, pandas, and sklearn modules. This was followed by importing the cleaned csv and validating the successful import of the file. The required X and Y variables were declared using df.iloc[].

```
In [6]:
from sklearn.tree import DecisionTreeClassifier

In [7]:
DT = DecisionTreeClassifier(random_state =100, criterion = "gini")

In [8]:
DT.fit(X_train,Y_train)

Out[8]: DecisionTreeClassifier(random_state=100)

In [9]:
DT.score(X_test,Y_test)

Out[9]: 0.7882101893838357

In [10]:
#Random forrest starts by picking a random set of X variables to combine to make a tree
#will make many trees, then take an average or vote from each tree

In [11]:
from sklearn.ensemble import RandomForestClassifier

In [12]:
RF = RandomForestClassifier(n_estimators = 100, criterion = "gini", random_state = 100)

In [13]:
RF.fit(X_train, Y_train)
```

```
In [14]: RF.score(X_test,Y_test)

Out[14]: 0.8535609495865564
```

```
In [15]: from sklearn.ensemble import GradientBoostingClassifier
         GBC = GradientBoostingClassifier(random_state = 100)

         GBC.fit(X_train,Y_train)
         GBC.score(X_test,Y_test)

         C:\Users\anton\anaconda3\lib\site-packages\sklearn\ensemble\_gb.py:494: DataConversionWarning: A column-vector y was passed
         when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
           y = column_or_1d(y, warn=True)

Out[15]: 0.8556948519605228
```

We used fit on the X and Y train data to create the model, followed by scoring the X and Y test data. This resulted in an accuracy score of 85.35%. The process was repeated using Gradient Boosting, where a model is built stage-wise by combining weak learners into a single strong learner in an iterative fashion. Gradient Boosting only increased the accuracy score to 85.5%

K-Nearest Neighbors – Price

Our team first used K-Nearest Neighbors (KNN) to create a machine learning model for Airbnb prices. KNN is a non-parametric, supervised learning classifier which uses proximity to make classifications or predictions about the grouping of an individual data point. The first step of the coding process consisted of importing the relevant libraries like Numpy, Pandas, and sklearn.model_selection into Python. This was followed by importing the cleaned CSV and validating the successful import of the file.

```
In [1]: import pandas as pd # Importing required libraries
        import numpy as np
        from sklearn.model_selection import train_test_split
```

```
In [2]: data = pd.read_csv('Clean_AirbnbData.csv') # Importing Cleaned data file
```

```
In [3]: data.head() # Validating Import process
```

Out[3]:

	host_response_time	host_response_rate	host_acceptance_rate	host_is_superhost	host_total_listings_count	host_has_profile_pic	host_identity_verified	neigl
0	within a few hours	1.0	0.69	True	2.0	True	True	
1	no response	0.0	0.00	False	1.0	True	True	Wa
2	no response	0.0	0.00	False	1.0	True	False	Pl
3	no response	0.0	0.00	False	4.0	True	True	
4	within an hour	1.0	0.91	False	70.0	True	True	Wa

5 rows × 37 columns

Based on our previous analysis, we were aware of some listings that were outliers based on extraordinarily high prices. For this reason, we used data.drop to remove all the entries where the price was greater than

$1700. We then split our pricing data into different groups with $50 price increments. This was done so that the KNN algorithm would find a match based on the correct range of pricing, rather than having to pinpoint an exact price to a dollar value. Our team felt like $50 increments were reasonable in giving hosts more information to make decisions. From here, we used df.iloc[] to isolate our X and Y variables.

```
data.drop(data[(data['price'] >1700)].index,inplace=True) # Restricting the values to price > 1700
```

```
ta['price_ranges'] = np.digitize(data["price"],[50,100,150,200,250,300,350,400]) # Splitting price into bins to imp
ta['price_ranges'] =pd.to_numeric(data['price_ranges'])
place=True
```

```
X = data.iloc[:,[1,2,3,4,5,12,13,14,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35]].values
X
Y = data.iloc[:,[36]].values
Y
```

We created test and train splits, with the test size being 20%. Next, we used fit on the X and Y train data to create the model, followed by scoring the X and Y test data. We created our KNN model using the 5 nearest neighbors and set the weights as "distance". The number of neighbors and the method used for weighting was selecting by creating a For loop that would run the model in different scenarios and selecting the one with the highest accuracy. This analysis resulted in a model with an accuracy score of 31.3%, which is unsatisfactory. Our team will develop a secondary model to create a better model for price.

```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size = 0.2, random_state = 42) # Splitting Train and Test
```

```
from sklearn.neighbors import KNeighborsClassifier # Importing KNeighborsClassifier
```

```
KNN_3 = KNeighborsClassifier (n_neighbors = 5,weights = "distance") # Setting neighbors
```

```
KNN_3.fit(X_train,Y_train) # Fitting X & Y Train data
```

```
/Users/abhinavbhatnagar/opt/anaconda3/lib/python3.9/site-packages/sklearn/neighbors/_classification.py:198: DataConve
rsionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_sample
s,), for example using ravel().
  return self._fit(X, y)
```

```
KNeighborsClassifier(weights='distance')
```

```
KNN_3.score(X_test, Y_test) # Fitting X & Y Test data
```

```
0.3132530120481928
```

Multiple Linear Regression – Price

To improve the accuracy, our team attempted to use multiple linear regression on sklearn to predict price. After importing the data and relevant libraries such as Pandas, NumPy, and sklearn, we filtered prices for all the listings with a price under $1700. All the quantitative variables that were boolean, float, or integer were captured as the X values. The Y value was the price of the listing.

```
1]: import numpy as np
    import pandas as pd
    from sklearn.model_selection import train_test_split
    from sklearn.linear_model import LinearRegression
    from sklearn.metrics import r2_score
    import matplotlib.pyplot as plt
    import seaborn as sns
    import warnings
    warnings.filterwarnings("ignore")
    data = pd.read_csv('Clean_AirbnbData.csv')

2]: data.drop(data[(data['price'] >1700)].index,inplace=True)

9]: X = data.iloc[:,[1,2,4,5,6,12,13,14,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35]].values
    Y = data["price"]
```

After splitting the data into train and test sets with a test size of 20%, we used linear_model.LinearRegression() to train our model. Using our model to then predict values in our testing set resulted in an R^2 value of 43.68%. The R^2 coefficient of determination is a statistical measure of how well the regression predictions approximate the real data points. Therefore, our model helps to explain about 43% of the variation in the data. This is lower than our team had expected and suggested that our model needed to be further refined to increase accuracy. However, when the X variable used was further reduced to only include 5 variables — bedrooms, beds, accommodates, amenities, and host_experience — the R^2 coefficient further reduced to 33.1%.

```
:
  Y_train, Y_test, X_train, X_test = train_test_split(Y,X,test_size = 0.2, random_state = 100)

  from sklearn import linear_model

  LINREG = linear_model.LinearRegression()
  LINREG.fit(X_train,Y_train)
  LINREG.score(X_test,Y_test)

: 0.4368166994167416

:
  Y_pred= LINREG.predict(X_test)
  Y_pred

: array([144.10607373, 630.93099727, 426.6956946 , ..., 191.80334968,
         427.08902098, 143.96002379])

:
  Accuracy= r2_score(Y_test,Y_pred)*100
  print(" Accuracy of the model is %.2f" %Accuracy)

   Accuracy of the model is 43.68
```

Random Forrest Classifier - Price

Finally, our team used a Random Forest Classifier (RFC) model to predict price. We wanted to make sure we also consider our qualitative variables into our model, so we used OneHotEncoder to create dummy variables for room_type, host_response_time, neighbourhood_cleansed, and property_type. This resulted in a data frame with 236 columns.

```
# creating instance of one-hot-encoder
df2 = OneHotEncoder(handle_unknown='ignore')

df2 = pd.DataFrame(enc.fit_transform(data[['host_response_time',"room_type","property_type","neighbourhood_cleanse

# merge with main data frame on key values
df3 = data.join(df2)
```

From here, the team used df.iloc[] to isolate the X and Y variables. The Y variable was price, which was converted into buckets of $50 price increments using np.digitize. For the X variables, all boolean, integer, and float variables were used including the new dummy variable created.

```
df4 = df3.drop(['host_response_time',"property_type","price","room_type","neighbourhood_cleansed","latitude","longitud

#Dropping variables
```

```
df4.head(5)
df4["price_copy"] = df3["price"]
```

```
X = df4.iloc[:,0:236].values
```

```
Y = np.digitize(df4.iloc[:,236],[50,100,150,200,250,300,350,400])
```

Using 100 estimators and the 'Gini' function to measure the quality of a split, we created an RFC model with 49.4% accuracy. To conclude, from our three methods of predicting price, our team is most confident in our Decision Tree model. It was interesting to see that including more data into the model such as neighborhoods and room type had a substantial impact on increasing accuracy of predictions. We expect that collecting even more information on hosts and existing listings, will further improve the accuracy of machine learning approaches to suggest prices in real-time

```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size = 0.2,random_state = 100)


RF = RandomForestClassifier(n_estimators = 100, criterion = "gini", random_state = 100)

RF.fit(X_train, Y_train)
RF.score(X_test,Y_test)
```

```
0.4944
```

Recommendation

How to Improve the Current Business Processes for Big Data Use

Airbnb gathers a plethora of data from their hosts listings. However, not every type of data they collected can turn into helpful information that can be translated into usable insights, making cleaning the data a very important first step. Moreover, there is more information on listings that can further increase the accuracy of machine learning approaches. For example, analyzing unstructured data such as the title of the listing, the description of the host, and the comments section. Therefore, to improve the current business process using big data analytics, our team suggests that Airbnb establish a set of guiding principles and standardized protocols to collect and store data. Using standard processes, Airbnb will have more data available in a format that is ready for analysis, rather than having to the tedious work of cleaning the data

for every data analytics project. For example, Airbnb may consider writing a script that automatically includes whether the title of listings contain strings such as “value” or “luxury”. We believe that having real-time data storage in standardized formats will help Airbnb improve the accuracy of price predictions. More importantly, it will give Airbnb the capability of including dynamic pricing in their application, where hosts can see changes in prices in their region in real-time. This can be taken to the next level by even including the price of hotels in the region using latitude and longitude as one of the considerations for price.

Next Steps Following the Results of Modeling

Based on the Superhosts analysis, we suggest that Airbnb hire data analysts who can automate the Superhost approval process. We have concluded from our t-tests that Superhosts do not charge significantly higher prices for their listings. However, research indicated that the Superhost status increases the customers willingness to want to stay with Airbnb. Therefore, we suggest that data analysts conduct monthly web scrapes from specific cities and use our model to predict potential Superhosts. After the prediction, the system should automatically notify those potential Superhosts for better engagement and potential for marketing.

From the insights gathered from the three models created for price, we saw that providing as much information as possible, both qualitative and quantitative, increases the accuracy of our model. By collecting more data and further refining the RFC model, Airbnb can improve their pricing capabilities. We suggest that Airbnb integrates a live dynamic pricing option into their app. A function in the app should be introduced that suggest prices to hosts based on the features of their property and the surrounding area, more like Uber’s dynamic pricing. They can also create a live dashboard that shows the prices of hotels in the same area. Airbnb hosts can then change their prices daily using this function, a handy tool to attract guests, while also driving revenue for Airbnb.

Potential or Projected Savings or Improvements for the Organization

After establishing a data collection standard, Airbnb will have smaller but more powerful datasets. This improvement will significantly improve the efficiency and accuracy of data analysis and increase the level of automation used in Airbnb’s decision support process. After implementing the Superhost approval automation process, we project to see the number of Superhosts on the platform increase by 15% and the number of bookings per Superhosts’ lists increase by 10%. We also project that by using the live dashboard, Airbnb owners will list their properties at a more reasonable price, further improving the rate of conversions.

Projected timeline for the implementation of the solution

The timeframe for the Superhost efficiency features we wish to implement will be as follows:

- Hiring 2 new software engineers: 2-3 months.
- Building the model that identifies and notifies potential Superhosts in Toronto and New York: 6-9 months, depending on the available extra workforce for this project.
- 1 year will be used as a pilot in Toronto and New York, followed by a larger scale expansion globally based on the results. The expected costs for the pilot phase are $175,000, depending on the firms existing resources.

The timeline for implementing the dynamic pricing tool for hosts is as follows:

- Developing standards for data extraction and storage, including unstructured data using comments, titles, and the description of listings
- Developing model that suggests pricing of an Airbnb property based on the features of the property, including the location, amenities, and analysis of unstructured data: 3-6 months.
- Piloting the live pricing suggestions model in two test cities, Toronto and New York: 3-6 months.
- Addition of live hotel pricing into dynamic capabilities of the application: 3-4 months. The expected costs for the pilot phase are $250,000, which primarily include the development of data standardization protocols and automated processing for collecting more data on hosts.
- 1 year will be used as a pilot in Toronto and New York, followed by a larger scale expansion globally based on the results.

Summary of Learnings

Overall, more variables impact the pricing of Airbnbs than expected, as the accuracy of our RFC model was only improved after the qualitative variables were included using OneHotEncoder. This highlighted the importance of including as much information as possible when developing machine learning models. Our analysis shows that there may be a discrepancy between host pricing and the experiences of customers. At times, properties with very poor ratings, still have a very high price. We also learned about different districts in the city that have more Airbnbs, and about areas where there are more Airbnbs than hotels. We learned about the correlations of different variables with price and noted how all types or ratings are often grouped together. The GTA serves as a great lesson for Airbnb on how the firm can optimize the user experience of their application in major cities. Careful inclusion of live pricing and using real-time data from hotels in the area will empower hosts to make better decisions and function to keep prices lower. Meanwhile, automating the process of identifying Superhosts will help increase the reputation and credibility of more listings on the platform, further increasing customers willingness to pay without significantly altering costs. Overall, Airbnb has many opportunities to further integrate data analytics when making decisions at the management level, including trends in different cities, suggesting prices, and capturing a higher portion of high quality Superhosts.

References

Foran, P. (2022, August 16). *Airbnb rentals almost as expensive as hotels bookings, new data finds*. Toronto. https://toronto.ctvnews.ca/airbnb-rentals-almost-as-expensive-as-hotels-bookings-new-data-finds-1.6028263

Get the Data. Inside Airbnb. (n.d.). Retrieved November 10, 2022, from http://insideairbnb.com/get-the-data/

Lord, C. (2022, November 16). *'Airbnbust'? Why Canada's short-term rental hosts are in for a harsh winter*. Global News. https://globalnews.ca/news/9271817/airbnb-vrbo-demand-mortgages-canada-short-term-rentals/