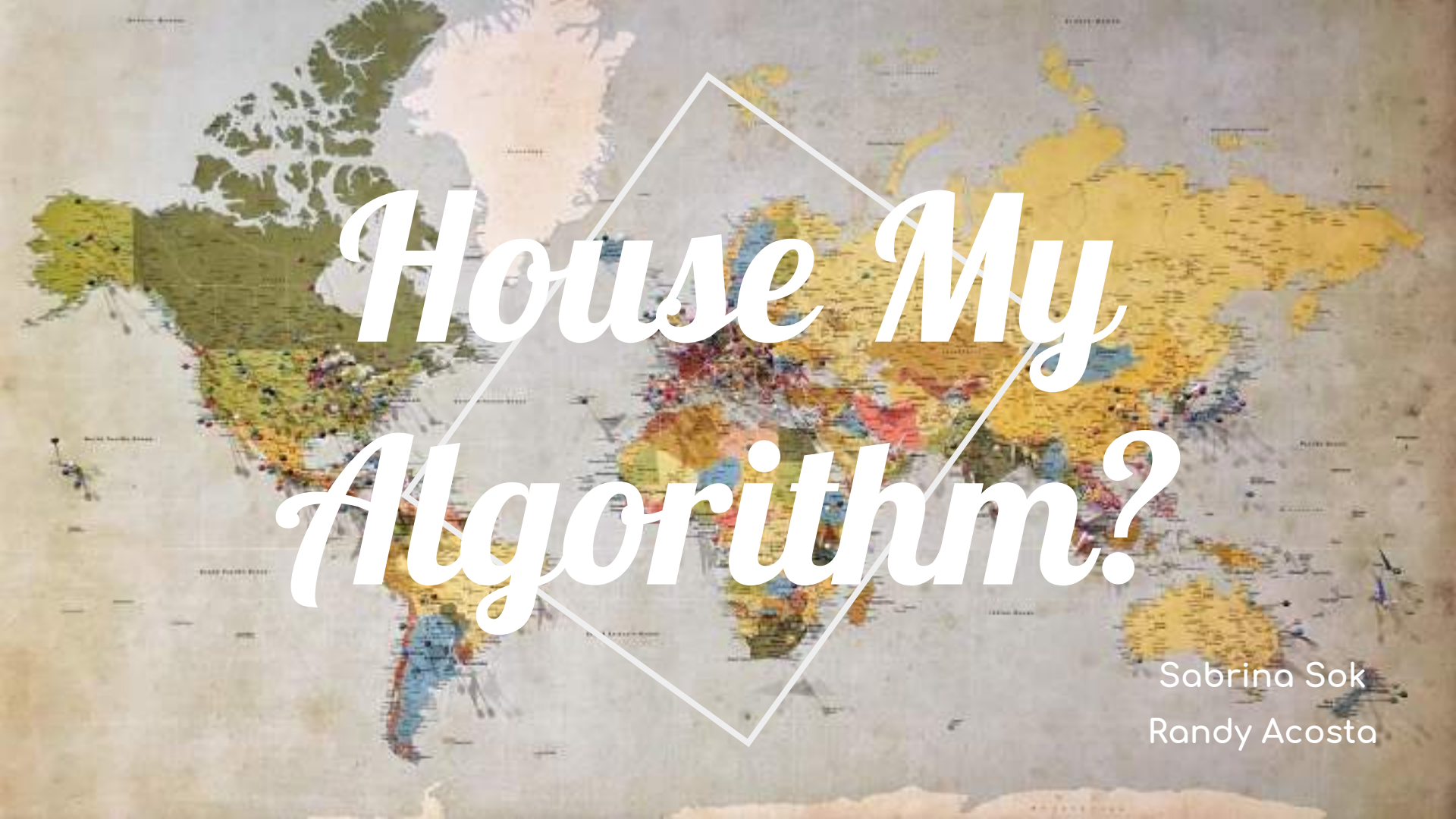

House My Algorithm?
Sabrina Sok
Randy Acosta

AGENDA

1 Introduction to Dataset and Algorithm

2 Q1: Sets of Amenities w/ High Review Scores

2.1 Across Entire Dataset vs. Within Binned Dataset

2.1.1 Apriori Algorithm, FP-Growth Algorithm, SparkML FP-Growth Algorithm

3 Q2: Sets of Amenities w/ High Review Scores Based on Location

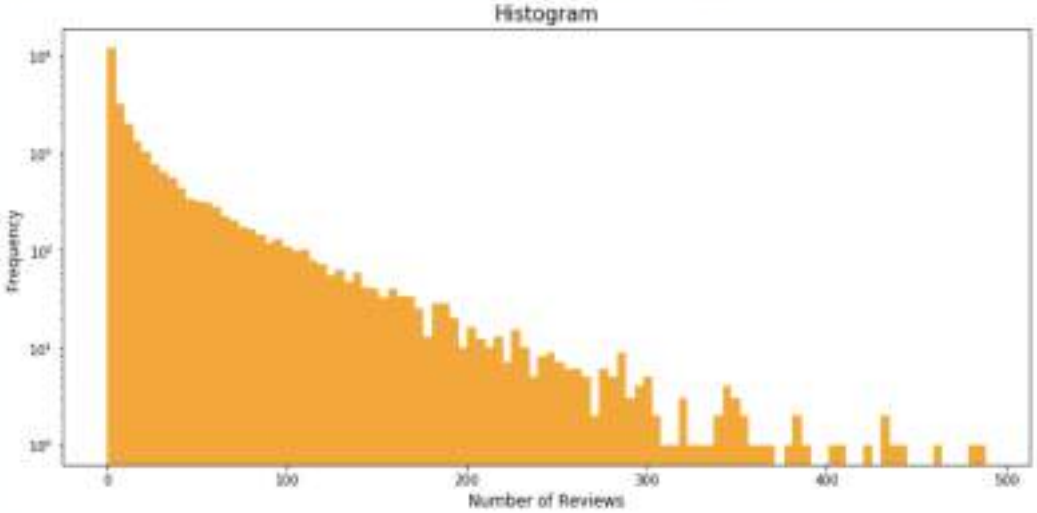
3.1 Across Entire Dataset vs. Within Binned Dataset

3.1.1 Apriori Algorithm, FP-Growth Algorithm

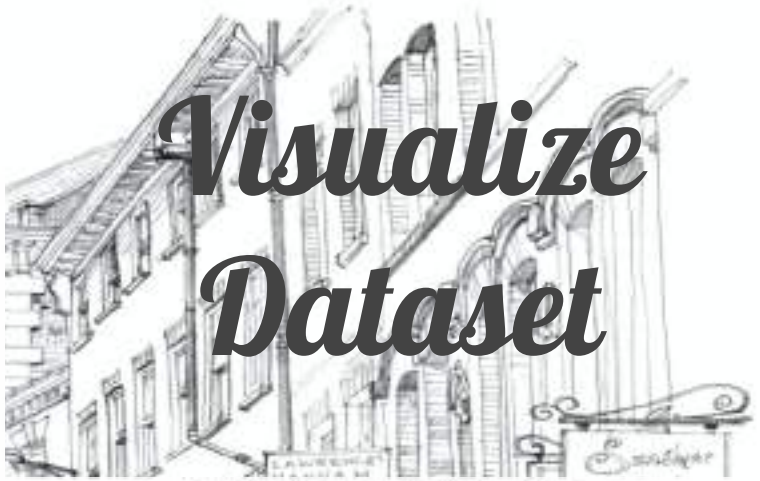
4 Recommendation

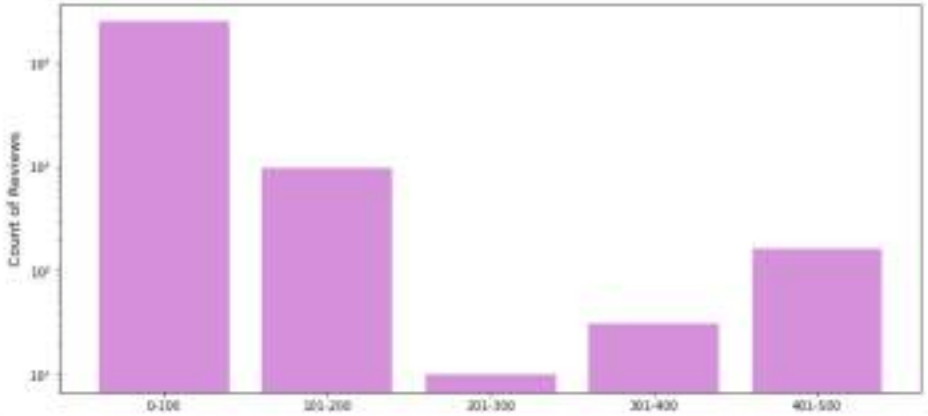
Introduction to Dataset

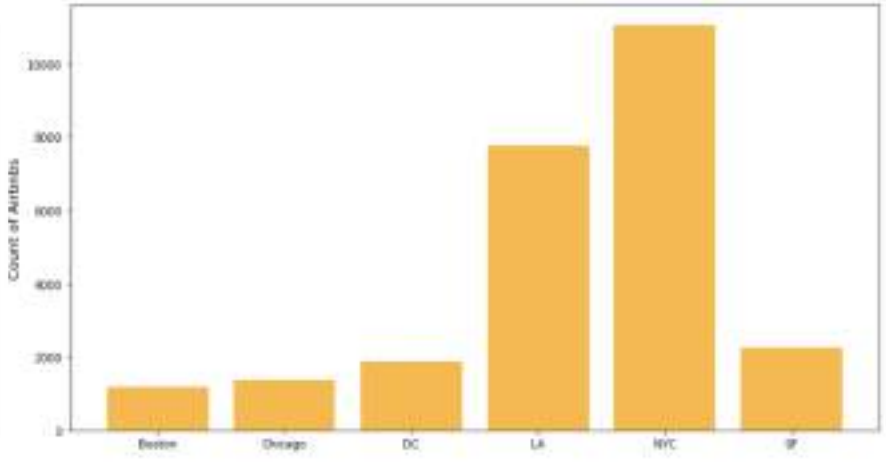
- 25,458 rows of data, 213 columns
- All columns are of type double and int
- There are no null values in the dataset
- Columns are mostly in binary form
- Example of columns are:
  - Property_Type_Condominium (0 for no, 1 for yes)
  - City_SF (0 for no, 1 for yes)
  - Firm mattress (0 for no, 1 for yes)
  - Washer (0 for no, 1 for yes)



Visualize Dataset








Google Map
United States
Seattle
Portland
Minneapolis
Chicago
Detroit
Toronto
Ottawa
Montreal
Quebec City
New York
Philadelphia
Washington
Indianapolis
Kansas City
Sacramento
San Francisco
San Jose
Las Vegas
Los Angeles
San Diego
Phoenix
Tucson
El Paso
Dallas
Houston
Austin
San Antonio
Nashville
Atlanta
Charlotte
Jacksonville
Orlando
Tampa
Miami
New Orleans
WASHINGTON
OREGON
CALIFORNIA
NEVADA
IDAHO
MONTANA
WYOMING
UTAH
ARIZONA
COLORADO
NEW MEXICO
NORTH DAKOTA
SOUTH DAKOTA
NEBRASKA
KANSAS
OKLAHOMA
TEXAS
MINNESOTA
IOWA
MISSOURI
ARKANSAS
LOUISIANA
WISCONSIN
ILLINOIS
MICHIGAN
INDIANA
OHIO
KENTUCKY
TENNESSEE
MISSISSIPPI
ALABAMA
GEORGIA
FLORIDA
SOUTH CAROLINA
NORTH CAROLINA
VIRGINIA
WEST VIRGINIA
PENNSYLVANIA
NEW YORK
VERMONT
NEW HAMPSHIRE
MAINE
MASSACHUSETTS
MARYLAND
BAJA CALIFORNIA
SONORA
CHIHUAHUA
COAHUILA
NUEVO LEON

accommodates	bathrooms	bedrooms	beds	cleaning_fee	host_response_rate	id	latitude	log_price	longitude	number_of_reviews
2	1.0	1.0	1.0	1	100	3895911	34.028372	0.0	-118.494449	6
3	1.0	1.0	1.0	1	100	9710289	40.720380	0.0	-73.942329	2
1	1.0	1.0	1.0	1	100	9051635	37.785434	0.0	-122.470284	2
1	1.0	0.0	1.0	1	100	708374	33.976026	0.0	-118.463471	7
2	1.0	1.0	1.0	1	0	626296	40.735573	0.0	-74.005996	0

Family/kid friendly	Fire extinguisher	Fireplace guards	Firm matress	Firm mattress	First aid kit	Fixed grab bars for shower & toilet	Flat	Flat smooth pathway to front door	Free parking on premises	Free parking on street	Game console	Garden or backyard	Grab-rails for shower and toilet	Ground floor access	Hair dryer	Hand or paper towel
0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0

When is Frequent Pattern Mining Algorithm Used?

- For companies to increase profits by understanding purchasing patterns
- In medical diagnosis to understand symptoms that coexist
- By studying behavior of users while navigating through websites

These information helps companies make decisions like, the kind of group discounts can be offered to customers, how to orient products at the stores or online, how to effectively advertise to customers, and what products/services/medications/websites to recommend to their customers.

Introduction to Association Rules

Association rules are most often used in businesses to help identify what items are commonly bought together. It determines how items are associated to each other.

Disadvantage: To identify the associations between lists of itemsets, the support values for each possible configuration of items need to be calculated, and then shortlist itemsets that meet minimum support threshold.

$$Support(\{X\} \rightarrow \{Y\}) = \frac{Transactions\ containing\ both\ X\ and\ Y}{Total\ number\ of\ transactions}$$

$$Confidence(\{X\} \rightarrow \{Y\}) = \frac{Transactions\ containing\ both\ X\ and\ Y}{Transactions\ containing\ X}$$

$$Lift(\{X\} \rightarrow \{Y\}) = \frac{(Transactions\ containing\ both\ X\ and\ Y)/(Transactions\ containing\ X)}{Fraction\ of\ transactions\ containing\ Y}$$

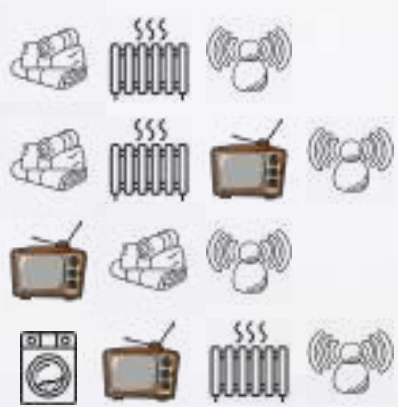
Association Rules

Antecedent

Consequent

Support Measure

$= \frac{2}{4}$



Confidence Measure

$= \frac{2}{3}$

Lift Measure

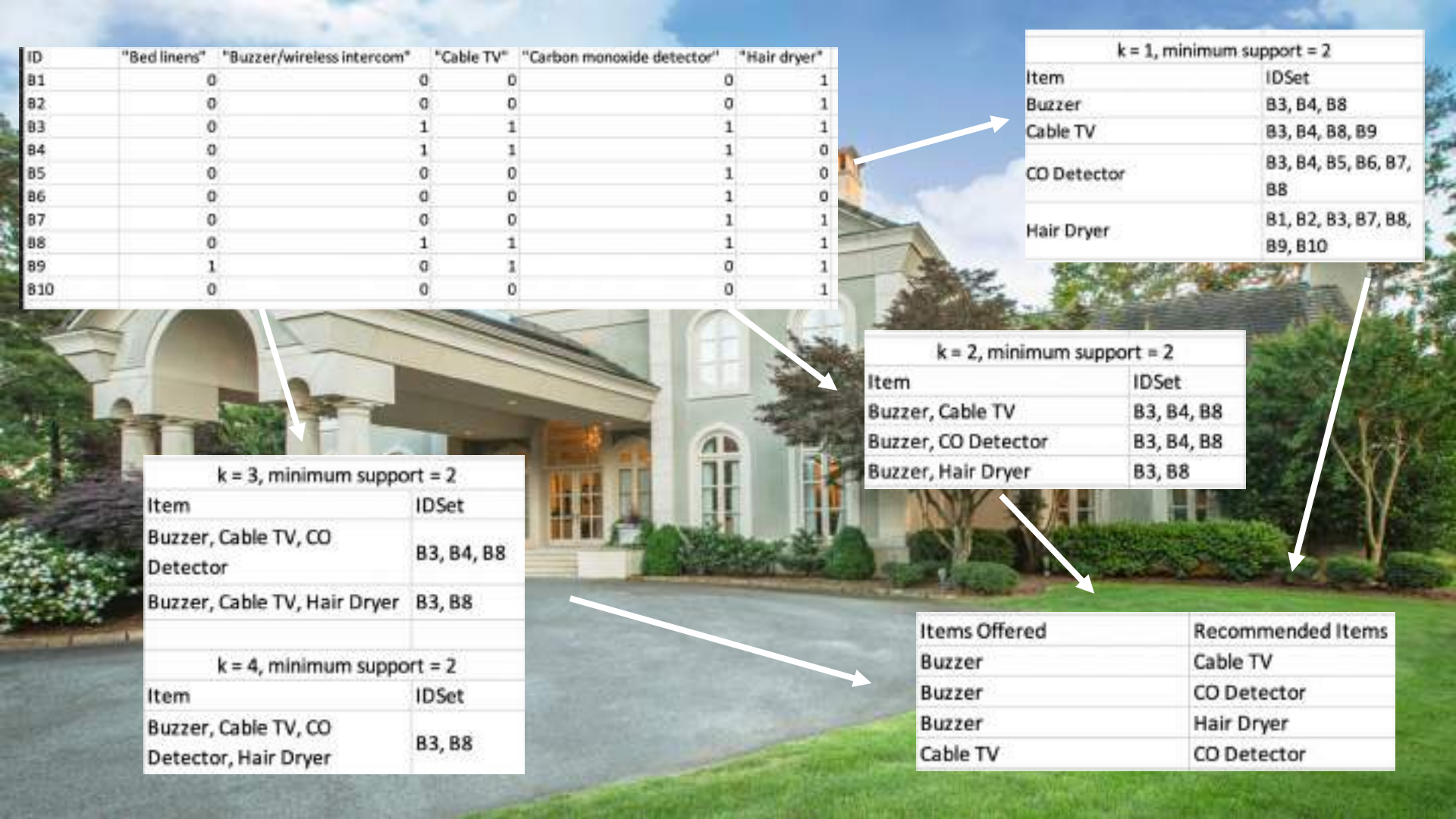
$= \frac{2}{9}$

Introduction to Apriori Algorithm

Compared to Association Rules, the number of itemsets the Apriori Algorithm examines is reduced.

Apriori algorithm uses frequent itemsets to generate association rules. Frequent itemset is an itemset whose support value is greater than a threshold value.

Disadvantages: Computationally expensive for large datasets (or when support value is low)



ID	"Bed linens"	"Buzzer/wireless intercom"	"Cable TV"	"Carbon monoxide detector"	"Hair dryer"
B1	0	0	0	0	1
B2	0	0	0	0	1
B3	0	1	1	1	1
B4	0	1	1	1	0
B5	0	0	0	1	0
B6	0	0	0	1	0
B7	0	0	0	1	1
B8	0	1	1	1	1
B9	1	0	1	0	1
B10	0	0	0	0	1

k = 1, minimum support = 2	
Item	IDSet
Buzzer	B3, B4, B8
Cable TV	B3, B4, B8, B9
CO Detector	B3, B4, B5, B6, B7, B8
Hair Dryer	B1, B2, B3, B7, B8, B9, B10

k = 2, minimum support = 2	
Item	IDSet
Buzzer, Cable TV	B3, B4, B8
Buzzer, CO Detector	B3, B4, B8
Buzzer, Hair Dryer	B3, B8

k = 3, minimum support = 2	
Item	IDSet
Buzzer, Cable TV, CO Detector	B3, B4, B8
Buzzer, Cable TV, Hair Dryer	B3, B8

k = 4, minimum support = 2	
Item	IDSet
Buzzer, Cable TV, CO Detector, Hair Dryer	B3, B8

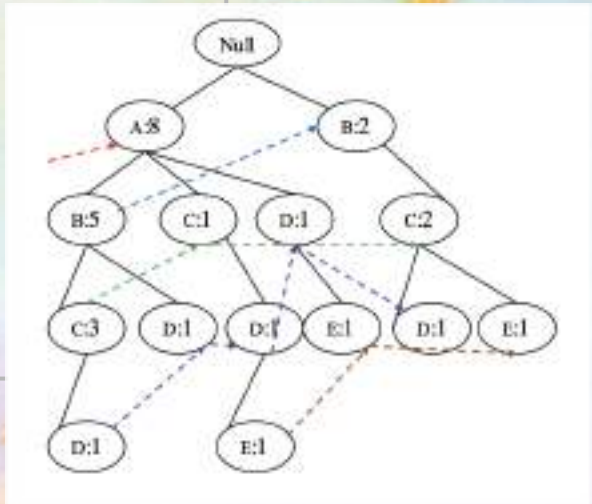
Items Offered	Recommended Items
Buzzer	Cable TV
Buzzer	CO Detector
Buzzer	Hair Dryer
Cable TV	CO Detector

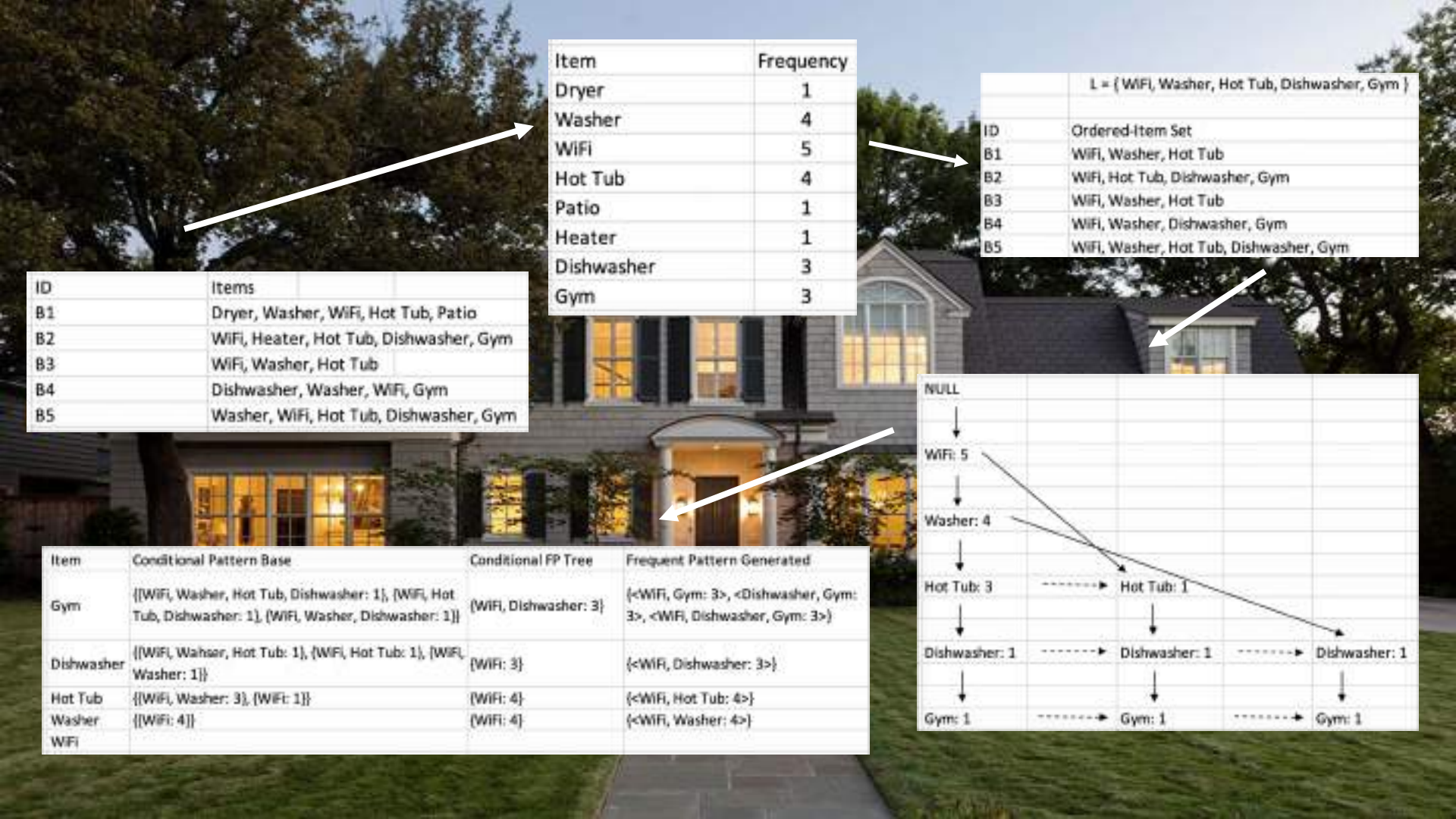
Introduction to FP-Growth Algorithm

FP growth technique uses pattern fragment growth to mine the frequent patterns from large database.

An extended prefix tree structure is used for storing crucial and compressed information about frequent patterns. FP growth discovers the frequent itemsets without candidate itemset generation.

Advantages: Reduced computation time and cost





Item	Frequency
Dryer	1
Washer	4
WiFi	5
Hot Tub	4
Patio	1
Heater	1
Dishwasher	3
Gym	3

ID	Items
B1	Dryer, Washer, WiFi, Hot Tub, Patio
B2	WiFi, Heater, Hot Tub, Dishwasher, Gym
B3	WiFi, Washer, Hot Tub
B4	Dishwasher, Washer, WiFi, Gym
B5	Washer, WiFi, Hot Tub, Dishwasher, Gym

	L = { WiFi, Washer, Hot Tub, Dishwasher, Gym }
ID	Ordered-Item Set
B1	WiFi, Washer, Hot Tub
B2	WiFi, Hot Tub, Dishwasher, Gym
B3	WiFi, Washer, Hot Tub
B4	WiFi, Washer, Dishwasher, Gym
B5	WiFi, Washer, Hot Tub, Dishwasher, Gym

Item	Conditional Pattern Base	Conditional FP Tree	Frequent Pattern Generated
Gym	{{WiFi, Washer, Hot Tub, Dishwasher: 1}, {WiFi, Hot Tub, Dishwasher: 1}, {WiFi, Washer, Dishwasher: 1}}	{WiFi, Dishwasher: 3}	{<WiFi, Gym: 3>, <Dishwasher, Gym: 3>, <WiFi, Dishwasher, Gym: 3>}
Dishwasher	{{WiFi, Wahser, Hot Tub: 1}, {WiFi, Hot Tub: 1}, {WiFi, Washer: 1}}	{WiFi: 3}	{<WiFi, Dishwasher: 3>}
Hot Tub	{{WiFi, Washer: 3}, {WiFi: 1}}	{WiFi: 4}	{<WiFi, Hot Tub: 4>}
Washer	{{WiFi: 4}}	{WiFi: 4}	{<WiFi, Washer: 4>}
WiFi			

Find Amenities w/ High Review Scores

1. Divided into two parts:
    a. Across entire dataset
    b. Within selected bins
2. Using different algorithms within FPM


First Part:
Across All
Dataset

Data Selection

```
cols = [c for c in df_amenities.columns if c.lower()[:14] != 'property_type_']
cols = [c for c in df_amenities.columns if c.lower()[:19] != 'translation missing']
```

```
df_amenities = df_amenities[cols]
```

```
df_amenities = df_amenities.drop(df_amenities.loc[:, 'accommodates':'instant_bookable_t'].columns, axis = 1)
```

```
df_amenities = df_amenities.drop(df_amenities.loc[:, 'first_review_Year':'numReviews_buckets2'].columns,
                                 axis = 1)
```

```
df_amenities = df_amenities.drop(['Internet', 'Shampoo', 'Hangers', 'Bath towel', 'Bed linens',
                                  'Body soap', 'Firm matress', 'Grab-rails for shower and toilet',
                                  'Hand soap', 'Washer / Dryer', 'Wide clearance to shower and toilet'], axis=1)
```

```
df_amenities.head(4)
```

	mooth athway front door	24-hour check-in	Accessible-height bed	Accessible-height toilet	Air conditioning	Air purifier	BBQ grill	Baby bath	Baby monitor	Babysitter recommendations	Bathtub with shower chair	Beach essentials	Buzzer/wireless intercom	Cable TV
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	1	0	0	0	0	0	0	0	0	0
3	0	0	0	0										

```
len(df_amenities.columns)
```

```
117
```

```
len(df_amenities[(df_amenities['Wireless Internet'] == 1)])
```

```
11793
```

```
len(df_amenities[(df_amenities['Internet'] == 1)])
```

```
7828
```

```
len(df_amenities[(df_amenities['Wireless Internet'] == 1) & (df_amenities['Internet'] == 1)])
```

```
7798
```

```
df_bin2.count()
```

```
25458
```

```
df_bin2.filter(df_bin2['numReviews_buckets2'] == 0).count()
```

```
11887
```

```
df_bin2.filter(df_bin2['review_scores_rating'] > 0).select('review_scores_rating').describe().show()
```

```
+-------+--------------------+
|summary|review_scores_rating|
+-------+--------------------+
|  count|               19741|
|   mean|   94.01154956689125|
| stddev|   7.978032677622043|
|    min|                20.0|
|    max|               100.0|
+-------+--------------------+
```

Continue

```
df.createOrReplaceTempView('df')
```

```
spark.sql('select percentile(cast(review_scores_rating as BIGINT), 0.5) from df').collect()
```

```
[Row(percentile(CAST(review_scores_rating AS BIGINT), CAST(0.5 AS DOUBLE), 1)=94.0)]
```

```
df_amenities = df_amenities.filter(df_amenities['review_scores_rating'] > 94)
```

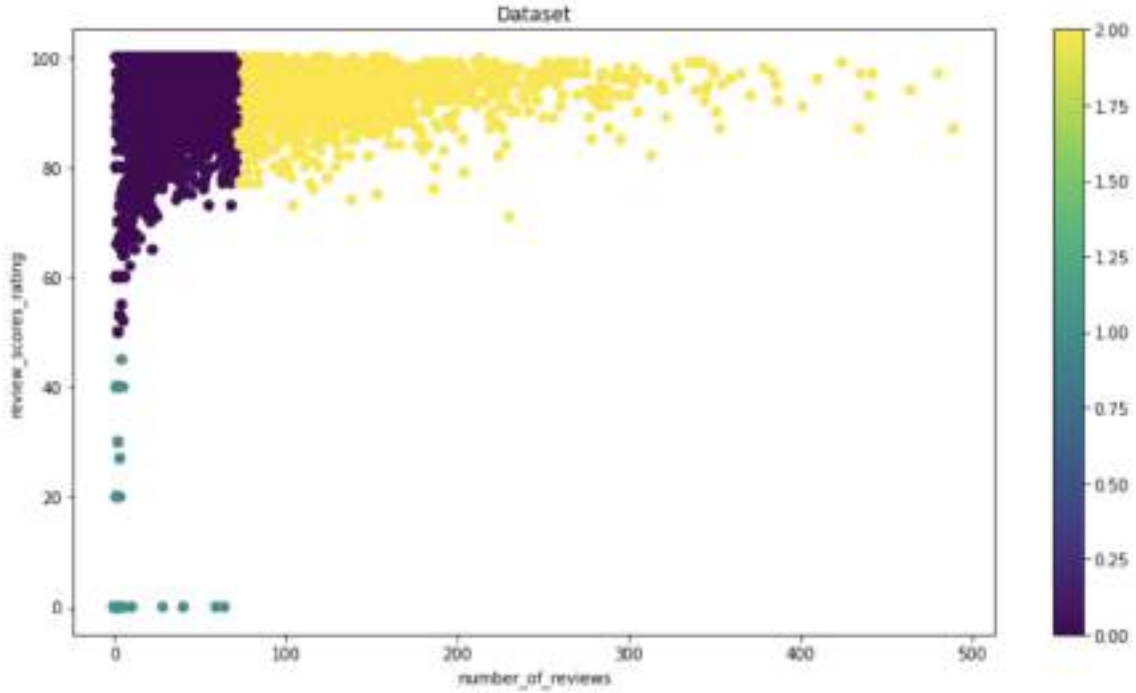
```
df_amenities.count()
```

```
12053
```

```
x1 = np.array(pd_transformed['number_of_reviews'])
x2 = np.array(pd_transformed['review_scores_rating'])

f, ax = plt.subplots(figsize=(13,7))
plt.title('Dataset')
plt.scatter(x1, x2, c=c)
plt.colorbar()
plt.xlabel('number_of_reviews')
plt.ylabel('review_scores_rating')
plt.show();
```

K-means?



Apriori Algorithm

```
from mlxtend.frequent_patterns import apriori, association_rules
```

```
frq_items = apriori(df_amenities, min_support=0.7, use_colnames = True)
```

```
frq_items['length'] = frq_items['itemsets'].apply(lambda x: len(x))
frq_items = frq_items.sort_values(by='support', ascending = False)
frq_items[(frq_items['length'] == 1)].head(20)
```

	support	itemsets	length
2	0.978429	(Wireless Internet)	1
4	0.933544	(Heating)	1
5	0.917116	(Kitchen)	1
3	0.899693	(Essentials)	1
1	0.873475	(Smoke detector)	1
0	0.753422	(Air conditioning)	1
6	0.743217	(TV)	1

```
frq_items[(frq_items['length'] == 2)].head(30)
```

	support	itemsets	length
15	0.922675	(Wireless Internet, Heating)	2
16	0.906164	(Kitchen, Wireless Internet)	2
14	0.889073	(Wireless Internet, Essentials)	2
20	0.867502	(Kitchen, Heating)	2
10	0.862607	(Smoke detector, Wireless Internet)	2
18	0.852983	(Heating, Essentials)	2
19	0.833900	(Kitchen, Essentials)	2
12	0.830913	(Smoke detector, Heating)	2
11	0.814403	(Smoke detector, Essentials)	2
13	0.811250	(Kitchen, Smoke detector)	2
7	0.745209	(Wireless Internet, Air conditioning)	2
17	0.737161	(TV, Wireless Internet)	2
8	0.727122	(Heating, Air conditioning)	2
21	0.711441	(TV, Heating)	2
9	0.704638	(Kitchen, Air conditioning)	2

Apriori Algorithm

```
frq_items[(frq_items['length'] == 3)].head(30)
```

	support	itemsets	length
31	0.858790	(Kitchen, Wireless Internet, Heating)	3
29	0.844769	(Wireless Internet, Essentials, Heating)	3
30	0.825438	(Kitchen, Wireless Internet, Essentials)	3
24	0.822368	(Smoke detector, Wireless Internet, Heating)	3
23	0.806355	(Smoke detector, Wireless Internet, Essentials)	3
25	0.802788	(Kitchen, Smoke detector, Wireless Internet)	3
33	0.793578	(Kitchen, Heating, Essentials)	3
26	0.779142	(Smoke detector, Heating, Essentials)	3
28	0.773915	(Kitchen, Heating, Smoke detector)	3
27	0.757654	(Kitchen, Essentials, Smoke detector)	3
22	0.719489	(Wireless Internet, Heating, Air conditioning)	3
32	0.706214	(TV, Wireless Internet, Heating)	3

```
frq_items[(frq_items['length'] == 4)].head(30)
```

	support	itemsets	length
38	0.786858	(Kitchen, Wireless Internet, Essentials, Heating)	4
34	0.772339	(Smoke detector, Wireless Internet, Essentials...	4
36	0.766780	(Kitchen, Smoke detector, Wireless Internet, H...	4
35	0.750850	(Kitchen, Smoke detector, Wireless Internet, E...	4
37	0.725877	(Kitchen, Heating, Essentials, Smoke detector)	4

```
frq_items[(frq_items['length'] == 5)].head(30)
```

	support	itemsets	length
39	0.720153	(Kitchen, Smoke detector, Wireless Internet, E...	5

Association Rules

```
rules = association_rules(frq_items, metric ="lift", min_threshold = 1.0)
```

```
rules = rules.sort_values(['confidence', 'lift'], ascending =[False, False])
```

```
rules.head()
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
195	(TV, Heating)	(Wireless Internet)	0.711441	0.978429	0.706214	0.992653	1.014538	0.010120	2.936106
158	(Kitchen, Heating, Essentials, Smoke detector)	(Wireless Internet)	0.725877	0.978429	0.720153	0.992113	1.013986	0.009933	2.735190
138	(TV)	(Wireless Internet)	0.743217	0.978429	0.737161	0.991851	1.013718	0.009976	2.647076
64	(Kitchen, Heating, Essentials)	(Wireless Internet)	0.793578	0.978429	0.786858	0.991532	1.013392	0.010098	2.547289
90	(Smoke detector, Heating, Essentials)	(Wireless Internet)	0.779142	0.978429	0.772339	0.991268	1.013123	0.010004	2.470451

```
rules[(rules['antecedent_len'] == 3) & (rules['confidence'] > 0.70)
      & (rules['lift'] > 1.0)].sort_values('support', ascending = False).head(10)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	antecedent_len

```
rules[(rules['antecedent_len'] == 4) & (rules['confidence'] > 0.70)
      & (rules['lift'] > 1.0)].sort_values('support', ascending = False).head(10)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	antecedent_len
158	(Kitchen, Heating, Essentials, Smoke detector)	(Wireless Internet)	0.725877	0.978429	0.720153	0.992113	1.013986	0.009933	2.735190	4
156	(Kitchen, Wireless Internet, Essentials, Smoke...	(Heating)	0.750850	0.933544	0.720153	0.959116	1.027393	0.019201	1.625490	4
157	(Kitchen, Wireless Internet, Heating, Smoke de...	(Essentials)	0.766780	0.899693	0.720153	0.939191	1.043901	0.030286	1.649532	4
160	(Smoke detector, Wireless Internet, Essentials...	(Kitchen)	0.772339	0.917116	0.720153	0.932431	1.016699	0.011828	1.226656	4
159	(Kitchen, Wireless Internet, Essentials, Heating)	(Smoke detector)	0.786858	0.873475	0.720153	0.915226	1.047798	0.032851	1.492486	4
	Essentials)	Internet)								
104	(Kitchen, Wireless Internet, Heating)	(Smoke detector)	0.858790	0.873475	0.766780	0.892861	1.022193	0.016648	1.180934	3
103	(Kitchen, Heating, Smoke detector)	(Wireless Internet)	0.773915	0.978429	0.766780	0.990780	1.012624	0.009559	2.339744	3

FP-Growth

```
from mlxtend.frequent_patterns import fpgrowth

fp_frqitems = fpgrowth(df_amenities, min_support=0.7, use_colnames = True)

fp_frqitems['length'] = fp_frqitems['itemsets'].apply(lambda x: len(x))
fp_frqitems[(fp_frqitems['length'] == 1)].sort_values('support', ascending = False).head(20)
```

	support	itemsets	length
5	0.978429	(Wireless Internet)	1
0	0.933544	(Heating)	1
1	0.917116	(Kitchen)	1
2	0.899693	(Essentials)	1
3	0.873475	(Smoke detector)	1
6	0.753422	(Air conditioning)	1
4	0.743217	(TV)	1

```
fp_frqitems[(fp_frqitems['length'] == 2)].head(20)
```

	support	itemsets	length
7	0.922675	(Wireless Internet, Heating)	2
8	0.867502	(Kitchen, Heating)	2
9	0.906164	(Kitchen, Wireless Internet)	2
11	0.852983	(Heating, Essentials)	2
12	0.833900	(Kitchen, Essentials)	2
13	0.889073	(Wireless Internet, Essentials)	2
18	0.830913	(Smoke detector, Heating)	2
19	0.814403	(Smoke detector, Essentials)	2
20	0.811250	(Kitchen, Smoke detector)	2
21	0.862607	(Smoke detector, Wireless Internet)	2
33	0.711441	(TV, Heating)	2
34	0.737161	(TV, Wireless Internet)	2
36	0.745209	(Wireless Internet, Air conditioning)	2
37	0.727122	(Heating, Air conditioning)	2
38	0.704638	(Kitchen, Air conditioning)	2

```
fp_frqitems[(fp_frqitems['length'] == 3)].\
sort_values('support', ascending = False).head(20)
```

	support	itemsets	length
10	0.858790	(Kitchen, Wireless Internet, Heating)	3
14	0.844769	(Wireless Internet, Essentials, Heating)	3
16	0.825438	(Kitchen, Wireless Internet, Essentials)	3
22	0.822368	(Smoke detector, Wireless Internet, Heating)	3
24	0.806355	(Smoke detector, Wireless Internet, Essentials)	3
28	0.802788	(Kitchen, Wireless Internet, Smoke detector)	3
15	0.793578	(Kitchen, Heating, Essentials)	3
23	0.779142	(Smoke detector, Heating, Essentials)	3
26	0.773915	(Kitchen, Heating, Smoke detector)	3
27	0.757654	(Kitchen, Essentials, Smoke detector)	3
39	0.719489	(Wireless Internet, Heating, Air conditioning)	3
35	0.706214	(TV, Wireless Internet, Heating)	3

```
fp_frqitems[(fp_frqitems['length'] == 4)].\
sort_values('support', ascending = False).head(20)
```

	support	itemsets	length
17	0.786858	(Kitchen, Wireless Internet, Essentials, Heating)	4
25	0.772339	(Smoke detector, Wireless Internet, Essentials...	4
29	0.766780	(Kitchen, Wireless Internet, Heating, Smoke de...	4
31	0.750850	(Kitchen, Wireless Internet, Essentials, Smoke...	4
30	0.725877	(Kitchen, Heating, Essentials, Smoke detector)	4

```
fp_frqitems[(fp_frqitems['length'] == 5)].\
sort_values('support', ascending = False).head(20)
```

	support	itemsets	length
32	0.720153	(Kitchen, Smoke detector, Wireless Internet, E...	5

```
spark_test = spark.read.format('com.databricks.spark.csv').options(header='true',
                                inferschema='true').load('df_amenities.csv')
```

```
spark_test.select('amenities_list').show(3)
```

```
+--------------------+
|      amenities_list|
+--------------------+
|, Cable TV, Eleva...|
|, Carbon monoxide...|
|, Air conditionin...|
+--------------------+
only showing top 3 rows
```

Spark ML

```
spark_test = spark_test.toPandas()
```

```
spark_test['amenities_list'] = spark_test['amenities_list'].apply(lambda x: str(x))\
.apply(lambda x: x.split(',')).apply(lambda x: list(x))
```

```
spark_test['amenities_list'] = spark_test['amenities_list'].\
apply(lambda row: [val for val in row if val != ''])
```

```
spark_test = spark.createDataFrame(spark_test)
```

```
spark_test['amenities_list'].head(5)
```

```
0    [ Cable TV,  Elevator in building,  Fire extinguisher,  First aid kit,  Free parking on premises,  Hair dr
yer,  Laptop friendly workspace,  Safety...
1    [ Carbon monoxide detector,  Family/kid friendly,  First aid kit,  Hair dryer,  Laptop friendly workspace,
Smoke detector,  Wireless Internet,  E...
2    [ Air conditioning,  Carbon monoxide detector,  Family/kid friendly,  Fire extinguisher,  Hair dryer,  Hot
tub,  Laptop friendly workspace,  Smoke...
3    [ Air conditioning,  Carbon monoxide detector,  Family/kid friendly,  Hair dryer,  Pets allowed,  Private
entrance,  Smoke detector,  Wireless Int...
4    [ Air conditioning,  Carbon monoxide detector,  Fire extinguisher,  First aid kit,  Free parking on premis
es,  Hair dryer,  Laptop friendly worksp...
Name: amenities_list, dtype: object
```

Spark ML

```
df_amenities.head(2)
```

Gym	Heating	Iron	Keypad	Kitchen	Lockbox	Microwave	Other	Oven	Pool	Refrigerator	Smartlock	Stove	TV	Washer	Waterfront	amenities_list
0	1	1	0	1	0	0	0	0	0	0	0	0	1	1	0	, Cable TV, Elevator in building, Fire extingui...
0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	, Carbon monoxide detector, Family/kid friendl...

```
spark_frq[(spark_frq['length'] == 1)].sort_values('freq', ascending = False).head(20)
```

	items	freq	length
16	[Wireless Internet]	11793	1
22	[Heating]	11252	1
24	[Kitchen]	11054	1
32	[Essentials]	10844	1
0	[Smoke detector]	10528	1
17	[Air conditioning]	9081	1
28	[TV]	8958	1

```
model.associationRules.sort(['confidence', 'lift'], ascending =[False, False]).show(10, truncate = False)
```

```
+----------------------------------------------------+----------------------+------------------+------------------+
|antecedent                                          |consequent            |confidence        |lift              |
+----------------------------------------------------+----------------------+------------------+------------------+
|[ TV,  Heating]                                     |[ Wireless Internet]|0.9926530612244898|1.014538060454403 |
|[ Smoke detector,  Essentials,  Kitchen,  Heating]|[ Wireless Internet]|0.9921133843867871|1.0139864853738612|
|[ TV]                                               |[ Wireless Internet]|0.9918508595668676|1.0137181726752695|
|[ Essentials,  Kitchen,  Heating]                   |[ Wireless Internet]|0.9915316257187663|1.0133919006858552|
|[ Smoke detector,  Essentials,  Heating]            |[ Wireless Internet]|0.9912682355446705|1.0131227035546437|
|[ Smoke detector,  Essentials,  Kitchen]            |[ Wireless Internet]|0.9910205869469996|1.0128695950540307|
|[ Smoke detector,  Kitchen,  Heating]               |[ Wireless Internet]|0.9907804459691252|1.0126241596935357|
|[ Essentials,  Heating]                             |[ Wireless Internet]|0.9903705865188211|1.0122052640813493|
|[ Smoke detector,  Essentials]                      |[ Wireless Internet]|0.990118174409128 |1.011947287047674 |
|[ Kitchen,  Heating]                                |[ Wireless Internet]|0.9899579188982403|1.0117834983872203|
+----------------------------------------------------+----------------------+------------------+------------------+
```

Kitchen, Wireless Internet, Heating, Smoke Detector, Essentials


Second Part:
Within Binned
Group

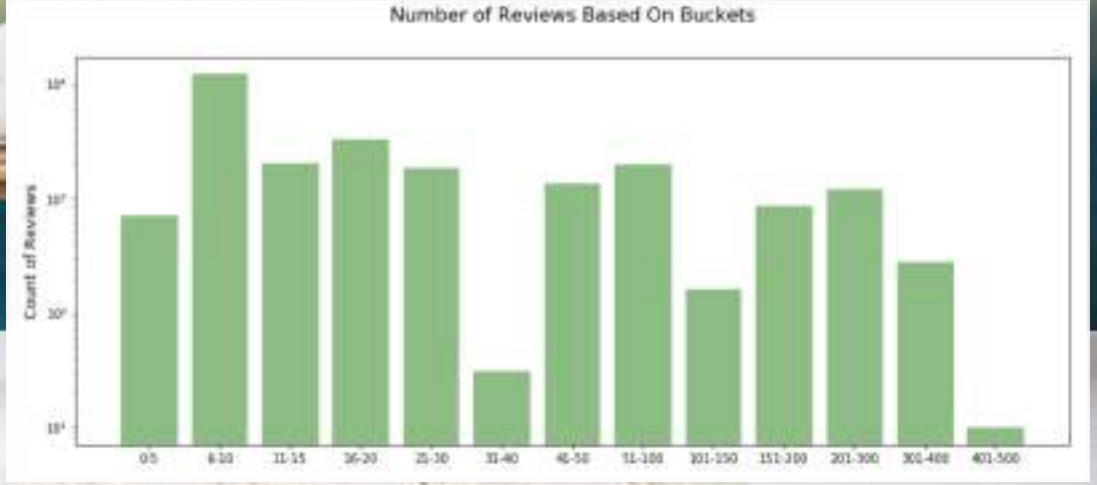
Data Selection

```
bucketizer = Bucketizer(splits=[0, 5, 10, 15, 20, 30, 40, 50, 100, 150,
                                200, 300, 400, 500, float('Inf') ],
                        inputCol="number_of_reviews", outputCol="numReviews_buckets2")
df_bin2 = bucketizer.setHandleInvalid("keep").transform(df)
```

```
df_bin2.select('number_of_reviews', 'review_scores_rating', 'numReviews_buckets2').show(10)
```

number_of_reviews	review_scores_rating	numReviews_buckets2
6	97.0	1.0
2	80.0	0.0
2	100.0	0.0
7	94.0	1.0
0	0.0	0.0
8	98.0	1.0
6	97.0	1.0
3	93.0	0.0
15	92.0	3.0
0	0.0	0.0

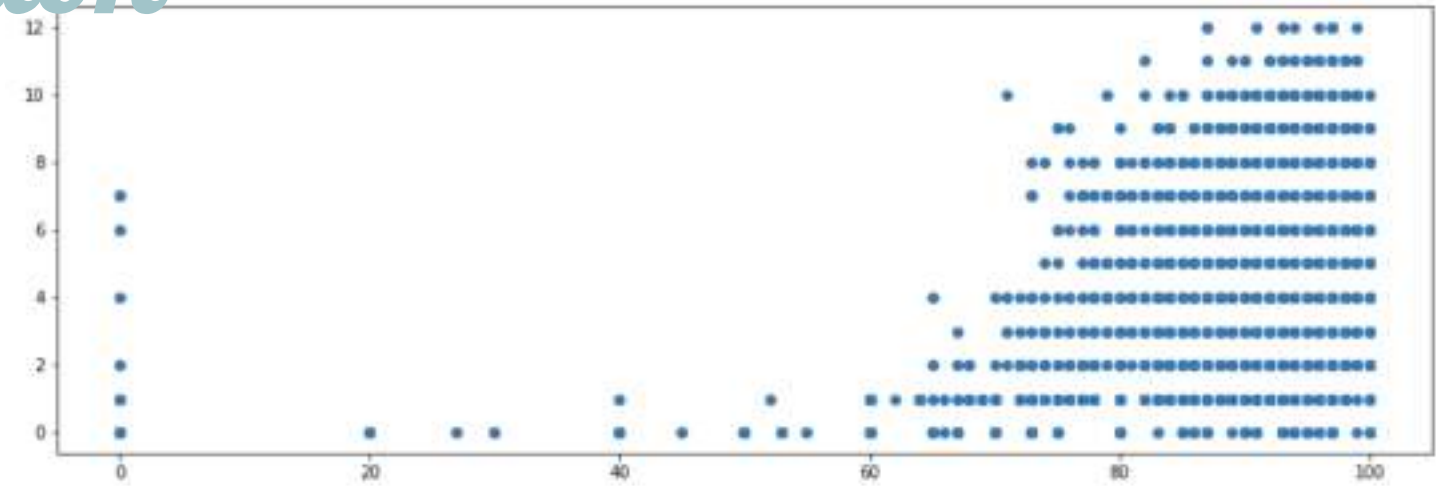
only showing top 10 rows



Data Selection

numReviews_buckets2	avg(review_scores_rating)
0.0	48.6364936485236
1.0	94.12430080795525
2.0	94.11538461538461
3.0	94.40423921271764
4.0	94.24225663716814
5.0	94.49457881567973
6.0	94.22014051522248
7.0	93.89588497768963
8.0	93.7661406025825
9.0	93.97491039426524
10.0	94.3167701863354
11.0	94.51612903225806
12.0	93.8

numReviews_buckets2	stddev_samp(review_scores_rating)
0.0	47.53936021586476
1.0	7.106592833154599
2.0	6.036332707932597
3.0	5.075508890374119
4.0	5.3064313232572
5.0	4.4363157158242315
6.0	5.51483277348511
7.0	5.325614570938444
8.0	4.2849108613129765
9.0	4.311113450442866
10.0	4.008463484067878
11.0	3.845958638553247
12.0	4.263540521418527



Data Selection

```
df_bin_amenities = df_bin2.filter(df_bin2['numReviews_buckets2'] > 2)
```

```
df_bin_amenities = df_norm.toPandas()
```

```
df_bin_amenities = df_bin_amenities[((df_bin_amenities['numReviews_buckets2'] == 3.0) &
                                     (df_bin_amenities['review_scores_rating'] > 99.47)) |
                                    ((df_bin_amenities['numReviews_buckets2'] == 4.0) &
                                     (df_bin_amenities['review_scores_rating'] > 99.54)) |
                                    ((df_bin_amenities['numReviews_buckets2'] == 5.0) &
                                     (df_bin_amenities['review_scores_rating'] > 98.92)) |
                                    ((df_bin_amenities['numReviews_buckets2'] == 6.0) &
                                     (df_bin_amenities['review_scores_rating'] > 99.73)) |
                                    ((df_bin_amenities['numReviews_buckets2'] == 7.0) &
                                     (df_bin_amenities['review_scores_rating'] > 99.21)) |
                                    ((df_bin_amenities['numReviews_buckets2'] == 8.0) &
                                     (df_bin_amenities['review_scores_rating'] > 98.04)) |
                                    ((df_bin_amenities['numReviews_buckets2'] == 9.0) &
                                     (df_bin_amenities['review_scores_rating'] > 98.28)) |
                                    ((df_bin_amenities['numReviews_buckets2'] == 10.0) &
                                     (df_bin_amenities['review_scores_rating'] > 98.31)) |
                                    ((df_bin_amenities['numReviews_buckets2'] == 11.0) &
                                     (df_bin_amenities['review_scores_rating'] > 98.35)) |
                                    ((df_bin_amenities['numReviews_buckets2'] == 12.0) &
                                     (df_bin_amenities['review_scores_rating'] > 98.06))]
```

```
df_bin_amenities.count()
```

```
numReviews_buckets2     650
review_scores_rating    650
z-score                 650
dtype: int64
```

Binned Unnormalized

Binned Unnormalized (High Buckets)

```
df_high_bucket = df_high_bucket[((df_high_bucket['numReviews_buckets2'] == 8.0) &
                                 (df_high_bucket['review_scores_rating'] > 98.04)) |
                                ((df_high_bucket['numReviews_buckets2'] == 9.0) &
                                 (df_high_bucket['review_scores_rating'] > 98.28)) |
                                ((df_high_bucket['numReviews_buckets2'] == 10.0) &
                                 (df_high_bucket['review_scores_rating'] > 98.31)) |
                                ((df_high_bucket['numReviews_buckets2'] == 11.0) &
                                 (df_high_bucket['review_scores_rating'] > 98.35)) |
                                ((df_high_bucket['numReviews_buckets2'] == 12.0) &
                                 (df_high_bucket['review_scores_rating'] > 98.06))]
```

```
len(df_high_bucket)
```

```
87
```

```
df_norm.groupBy('numReviews_buckets2').agg(mean("z-score"),
stddev("z-score")).sort('numReviews_buckets2', ascending = True).show(10)
```

```
+--------------------+--------------------+--------------------+
|numReviews_buckets2|        avg(z-score)|stddev_samp(z-score)|
+--------------------+--------------------+--------------------+
|                 0.0|5.929657557617728...|  1.0000420654117332|
|                 1.0|-8.83210361417153...|  1.0001554122318808|
|                 2.0|3.200319002158346...|  1.0002531325189254|
|                 3.0|-6.19909540472002...|  1.0003787161658209|
|                 4.0|3.035919598311269...|  1.0002766634442195|
|                 5.0|5.540929590789772...|  1.0004172752111498|
|                 6.0|8.320172549884078...|  1.0005859947763391|
|                 7.0|9.564320910206604...|  1.0002479851247048|
|                 8.0|4.077720147834147...|  1.0007181329471326|
|                 9.0|-9.23196206856761...|  1.0017969466424617|
+--------------------+--------------------+--------------------+
only showing top 10 rows
```

```
df_norm.filter(df_norm['z-score'] > 1.0).count()
```

```
4581
```

Binned Normalized Z-Score

Apriori Algorithm

(Binned Normalized)

	support	itemsets	length
28	0.831041	(Kitchen, Heating, Wireless Internet)	3
27	0.810303	(Essentials, Wireless Internet, Kitchen)	3
26	0.800262	(Essentials, Heating, Wireless Internet)	3
22	0.782580	(Kitchen, Smoke detector, Wireless Internet)	3
21	0.771011	(Smoke detector, Heating, Wireless Internet)	3
20	0.765553	(Essentials, Smoke detector, Wireless Internet)	3
29	0.761842	(Kitchen, Heating, Essentials)	3
25	0.738048	(Kitchen, Heating, Smoke detector)	3
24	0.731281	(Kitchen, Smoke detector, Essentials)	3
23	0.726916	(Essentials, Heating, Smoke detector)	3

	support	itemsets	length
2	0.964637	(Wireless Internet)	1
5	0.917704	(Kitchen)	1
4	0.898494	(Heating)	1
3	0.877538	(Essentials)	1
1	0.847632	(Smoke detector)	1
0	0.742851	(Air conditioning)	1
6	0.714473	(TV)	1

	support	itemsets	length
15	0.901986	(Kitchen, Wireless Internet)	2
14	0.883650	(Heating, Wireless Internet)	2
13	0.863130	(Essentials, Wireless Internet)	2
19	0.843047	(Kitchen, Heating)	2
9	0.830605	(Smoke detector, Wireless Internet)	2
18	0.821436	(Kitchen, Essentials)	2
17	0.810968	(Essentials, Heating)	2
12	0.795678	(Kitchen, Smoke detector)	2
11	0.783890	(Heating, Smoke detector)	2
10	0.777123	(Essentials, Smoke detector)	2
7	0.731718	(Air conditioning, Wireless Internet)	2
16	0.704868	(TV, Wireless Internet)	2
8	0.701594	(Air conditioning, Heating)	2

	support	itemsets	length
33	0.753329	(Essentials, Heating, Wireless Internet, Kitchen)	4
32	0.727352	(Kitchen, Smoke detector, Heating, Wireless In...	4
31	0.721676	(Essentials, Smoke detector, Wireless Internet...	4
30	0.717311	(Essentials, Smoke detector, Heating, Wireless...	4

Apriori Algorithm

Binned Unnormalized

	support	itemsets	length
5	0.992308	(Wireless Internet)	1
7	0.969231	(Heating)	1
6	0.936923	(Essentials)	1
4	0.930769	(Smoke detector)	1
9	0.886154	(Kitchen)	1
2	0.815385	(Hair dryer)	1
1	0.803077	(Carbon monoxide detector)	1
10	0.792308	(TV)	1
8	0.769231	(Iron)	1
3	0.767692	(Laptop friendly workspace)	1
0	0.760000	(Air conditioning)	1

Unnormalized (High Buckets)

	support	itemsets	length
5	1.000000	(Wireless Internet)	1
7	0.965517	(Heating)	1
6	0.919540	(Essentials)	1
4	0.908046	(Smoke detector)	1
2	0.885057	(Hair dryer)	1
8	0.873563	(Iron)	1
9	0.873563	(Kitchen)	1
3	0.839080	(Laptop friendly workspace)	1
0	0.816092	(Carbon monoxide detector)	1

Binned Normalized

```
bin_frqitems[(bin_frqitems['length'] == 4)].head(20)
```

	support	itemsets	length
33	0.753329	(Essentials, Heating, Wireless Internet, Kitchen)	4
32	0.727352	(Kitchen, Smoke detector, Heating, Wireless In...	4
31	0.721676	(Essentials, Smoke detector, Wireless Internet...	4
30	0.717311	(Essentials, Smoke detector, Heating, Wireless...	4

Association Rules

Binned Normalized

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	antecedent_len
63	(Essentials, Heating, Kitchen)	(Wireless Internet)	0.761842	0.964837	0.753329	0.988825	1.025075	0.018428	3.164576	3
62	(Essentials, Heating, Wireless Internet)	(Kitchen)	0.800262	0.917704	0.753329	0.941353	1.025770	0.018926	1.403250	3
64	(Essentials, Wireless Internet, Kitchen)	(Heating)	0.810303	0.898484	0.753329	0.929688	1.034718	0.025276	1.443644	3
65	(Kitchen, Heating, Wireless Internet)	(Essentials)	0.831041	0.877538	0.753329	0.906488	1.032990	0.024059	1.309590	3
90	(Kitchen, Heating, Smoke detector)	(Wireless Internet)	0.738048	0.964837	0.727352	0.985507	1.021638	0.015404	2.440079	3
93	(Heating, Wireless Internet, Smoke detector)	(Kitchen)	0.771011	0.917704	0.727352	0.943375	1.027973	0.019793	1.453355	3

Binned Unnormalized (High Buckets)

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	antecedent_len
43	(Essentials, Wireless Internet, Smoke detector)	(Heating)	0.896552	0.965517	0.873563	0.974359	1.009158	0.007927	1.344828	3
44	(Essentials, Heating, Wireless Internet)	(Smoke detector)	0.896552	0.908046	0.873563	0.974359	1.073028	0.059453	3.586207	3
45	(Heating, Wireless Internet, Smoke detector)	(Essentials)	0.885057	0.919540	0.873563	0.987013	1.073377	0.059717	6.195402	3
64	(Essentials, Wireless Internet, Smoke detector)	(Hair dryer)	0.896552	0.885057	0.862069	0.961538	1.086414	0.068569	2.988506	3
66	(Essentials, Wireless Internet, Hair dryer)	(Smoke detector)	0.873563	0.908046	0.862069	0.986842	1.086775	0.068833	6.988506	3

FP-Growth

Binned Normalized

	support	itemsets	length
14	0.753329	(Kitchen, Heating, Wireless Internet, Essentials)	4
27	0.727352	(Kitchen, Smoke detector, Heating, Wireless Internet)	4
23	0.721676	(Kitchen, Smoke detector, Wireless Internet, Essentials)	4
24	0.717311	(Essentials, Smoke detector, Heating, Wireless Internet)	4

Binned Unnormalized (High Buckets)

	support	itemsets	length
36	0.839080	(Wireless Internet, Hair dryer, Essentials, Smoke detector, Heating)	5
84	0.839080	(Iron, Wireless Internet, Hair dryer, Essentials, Smoke detector)	5
92	0.827586	(Iron, Wireless Internet, Hair dryer, Essentials, Heating)	5
96	0.827586	(Iron, Wireless Internet, Essentials, Smoke detector, Heating)	5
148	0.816092	(Laptop friendly workspace, Wireless Internet, Hair dryer, Essentials, Smoke detector)	5
97	0.816092	(Iron, Hair dryer, Essentials, Smoke detector, Heating)	5
98	0.816092	(Iron, Wireless Internet, Hair dryer, Smoke detector, Heating)	5

Binned Normalized

Internet,
Essentials,
Heating, Kitchen,
Smoke Detector

Binned Overall & Binned (High Buckets)

Internet, Heating,
Essentials,
Smoke Detector, Hair
Dryer, Iron, Laptop
Friendly Workspace

Hmmmm.....

```
df_low = df_bin2[df_bin2['review_scores_rating'] < 10]

df_low.count()
```

```
5717
```

	support	itemsets	length
0	0.910093	(Wireless Internet)	1
1	0.886304	(Kitchen)	1
2	0.823159	(Heating)	1
3	0.794123	(Essentials)	1
4	0.757740	(Smoke detector)	1
5	0.700542	(Air conditioning)	1

	support	itemsets	length
9	0.746545	(Kitchen, Heating, Wireless Internet)	3
12	0.718384	(Kitchen, Wireless Internet, Essentials)	3

	support	itemsets	length
2	0.978429	(Wireless Internet)	1
4	0.933544	(Heating)	1
5	0.917116	(Kitchen)	1
3	0.899693	(Essentials)	1
1	0.873475	(Smoke detector)	1
0	0.753422	(Air conditioning)	1
6	0.743217	(TV)	1

	support	itemsets	length
5	0.978429	(Wireless Internet)	1
0	0.933544	(Heating)	1
1	0.917116	(Kitchen)	1
2	0.899693	(Essentials)	1
3	0.873475	(Smoke detector)	1
6	0.753422	(Air conditioning)	1
4	0.743217	(TV)	1

	support	itemsets	length
2	0.964637	(Wireless Internet)	1
5	0.917704	(Kitchen)	1
4	0.898494	(Heating)	1
3	0.877538	(Essentials)	1
1	0.847632	(Smoke detector)	1
0	0.742851	(Air conditioning)	1
6	0.714473	(TV)	1

	support	itemsets	length
5	1.000000	(Wireless Internet)	1
7	0.966517	(Heating)	1
6	0.919540	(Essentials)	1
4	0.908046	(Smoke detector)	1
2	0.885057	(Hair dryer)	1
8	0.873563	(Iron)	1
9	0.873563	(Kitchen)	1
3	0.839080	(Laptop friendly workspace)	1
0	0.816092	(Carbon monoxide detector)	1


Additional
Investigation


USA
WHAAAA?!?!
Alaska
Hawaii
Pacific Ocean
Atlantic Ocean
Gulf of Mexico

Apriori Algorithm

Unbinned

East Coast

	support	itemsets	length
40	0.882174	(Kitchen, Heating, Wireless Internet)	3
38	0.850520	(Essentials, Heating, Wireless Internet)	3
27	0.841886	(Air conditioning, Heating, Wireless Internet)	3
39	0.836611	(Essentials, Wireless Internet, Kitchen)	3
28	0.823341	(Kitchen, Air conditioning, Wireless Internet)	3
33	0.813749	(Smoke detector, Heating, Wireless Internet)	3

West Coast

	support	itemsets	length
37	0.826313	(Essentials, Heating, Wireless Internet)	3
30	0.818127	(Smoke detector, Heating, Wireless Internet)	3
40	0.817528	(Kitchen, Heating, Wireless Internet)	3
29	0.816530	(Essentials, Smoke detector, Wireless Internet)	3
38	0.802156	(Essentials, Wireless Internet, Kitchen)	3

FP-Growth Binned Unnormalized

East Coast

	support	itemsets	length
37	0.845361	(Wireless Internet, Essentials, Kitchen, Air conditioning, Heating)	5
53	0.835052	(Wireless Internet, Hair dryer, Essentials, Air conditioning, Heating)	5
64	0.814433	(Wireless Internet, Hair dryer, Kitchen, Air conditioning, Heating)	5
85	0.814433	(Wireless Internet, Essentials, Smoke detector, Air conditioning, Heating)	5

West Coast

	support	itemsets	length
37	0.848168	(Wireless Internet, Hair dryer, Essentials, Smoke detector, Heating)	5
85	0.837696	(Laptop friendly workspace, Wireless Internet, Hair dryer, Essentials, Smoke detector)	5
53	0.832461	(Iron, Wireless Internet, Hair dryer, Essentials, Smoke detector)	5
61	0.821990	(Iron, Wireless Internet, Hair dryer, Essentials, Heating)	5
65	0.821990	(Iron, Wireless Internet, Essentials, Smoke detector, Heating)	5

FP-Growth Binned Normalized

	support	itemsets	length
9	0.854575	(Kitchen, Heating, Wireless Internet)	3
13	0.821015	(Kitchen, Wireless Internet, Essentials)	3
14	0.808230	(Essentials, Heating, Wireless Internet)	3

East Coast

	support	itemsets	length
11	0.786566	(Kitchen, Wireless Internet, Essentials)	3
36	0.786024	(Kitchen, Heating, Wireless Internet)	3
37	0.777898	(Essentials, Heating, Wireless Internet)	3
16	0.774648	(Kitchen, Smoke detector, Wireless Internet)	3

West Coast

Conclusion

1. The number of reviews do not have a significant impact on review score rating.
2. Sets of amenities do not provide a clear indication of what amenity, or amenities are needed to guarantee a high review score.
3. For Airbnb hosts looking to increase their review scores:
    a. There seems to be a“mandatory” set of amenities that are needed.

Conclusion

4. The seemingly “mandatory” set of amenities show up in “poor” review scored AirBnB’s.
5. We now know what information we need to come to more definitive conclusion.
    a. Specific information about why a review score is bad.
        i. Business trip vs. vacation vs. personal travel
        ii. Bad neighborhood (factors indicating why)

References

1. https://towardsdatascience.com/plotting-business-locations-on-maps-using-multiple-plotting-libraries-in-python-45a00ea770af
2. https://www.pinterest.com/pin/163044448983653898/
3. https://www.pinterest.com/pin/87820261463065433/
4. https://paintingvalley.com/architecture-sketch-wallpaper
5. https://www.elledecor.com/design-decorate/g9096337/beautiful-views/
6. https://pdfs.semanticscholar.org/3bc3/6381f77ec371b0931d3276a3f4f58d186427.pdf
7. https://www.thesleepjudge.com/white-bedroom-ideas/
8. https://www.youtube.com/watch?v=guVvtZ7ZClw
9. https://www.kdnuggets.com/2016/04/association-rules-apriori-algorithm-tutorial.html/2
10. https://www.muralswallpaper.com/shop-murals/black-brick-wallpaper-mural/
11. https://bluemarbleprivate.com/insight/arijiju-sirai-luxury-safari-lodges
12. https://beyond-universe.fandom.com/wiki/Corridor_of_All-Creator
13. https://brickhunter-store.myshopify.com/collections/wall-thin-brick/products/old-chicago-blanc-flat
14. https://www.pinterest.com/pin/196188127497314831/
15. https://waliicorners.com/product/custom-fashion-wallpaper-geometric-gold-lines-modern-minimalistic-luxury-abstract-background-wall-papers-home-decor/
16. https://www.pinterest.com/pin/181340322471070745/
17. http://rasbt.github.io/mlxtend/user_guide/frequent_patterns/association_rules/
18. http://rasbt.github.io/mlxtend/api_subpackages/mlxtend.frequent_patterns/#fpgrowth