CS 5010 Group Project

David Ackerman (dja2dg) Jeremey Donovan (jdd5dw) Xin Huang (xh2jg)

I. Introduction

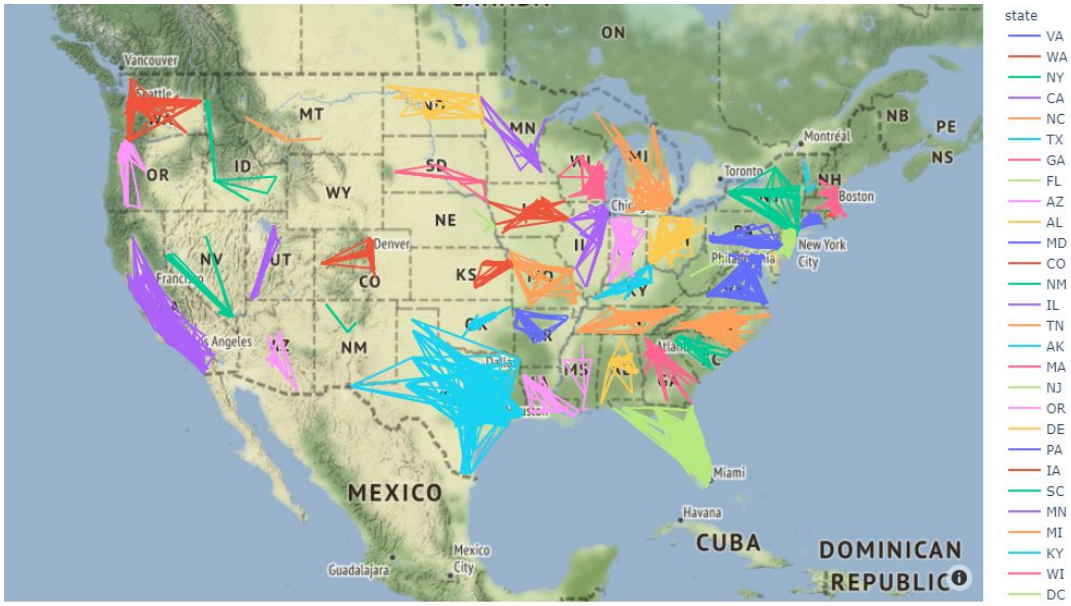
We set out to understand how apartments are priced in the United States. This has both applications for both tenants and landlords. Tenants would benefit from unbiased pricing information when signing or renewing lease agreements. Landlords, especially those with limited property holdings, could ensure they are charging suitable rent. Moreover, landlords could determine which amenities would have the highest return-on-investment.

Beyond pricing, we also sought to understand how amenities vary in the 'average' apartment across cities. This is designed to help people moving to new cities decide where they are most likely to get the housing they desire.

II. The Data

Avoiding the need for web scraping, we were able to locate a recent (2019-12-28) apartment for rent dataset[1] containing 10,000 rows on the UCI Machine Learning Repository. Since the creator had not uploaded the data, we reached out to him (fredrick_nilsson@yahoo.com) directly and he kindly sent us his datafile. This data sets sampled apartment listings across 51 states.

[1] https://archive.ics.uci.edu/ml/datasets/Apartment+for+rent+classified



The attributes we used from the apartments for rent dataset were as follows (with a discussion of how we cleaned the data to address outliers and missing data):

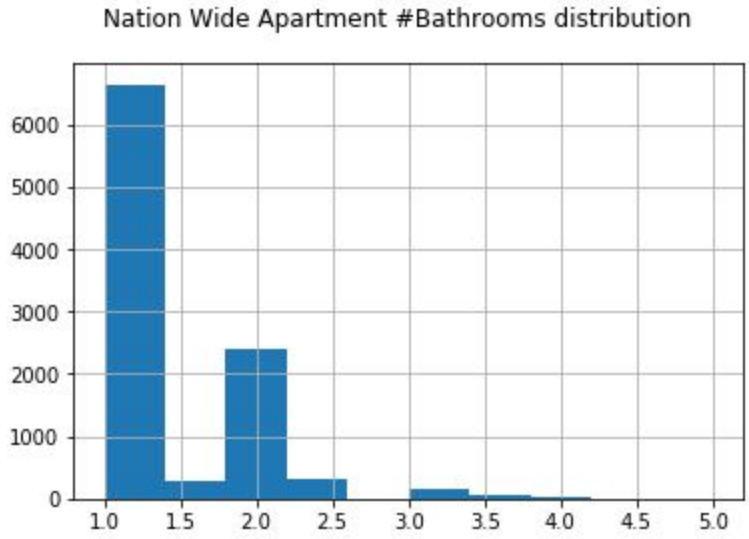
- Amenities
    - 3,549 missing values coded in the dataset as ‘null’; we have retained these records and have assumed ‘none’

Amenity	Frequency		Amenity	Frequency
Parking	3727		AC	662
Dishwasher	3266		Elevator	642
Pool	3238		Tennis	482
Refrigerator	3133		Gated	473
Patio/Deck	2472		Wood Floors	357
Cable or Satellite	1678		Hot Tub	346
Storage	1531		Basketball	318
Gym	1469		TV	207
Internet Access	1441		View	149
Clubhouse	1317		Doorman	29
Garbage Disposal	1210		Alarm	23
Washer Dryer	1077		Golf	23
Fireplace	1065		Luxury	11
Playground	782			

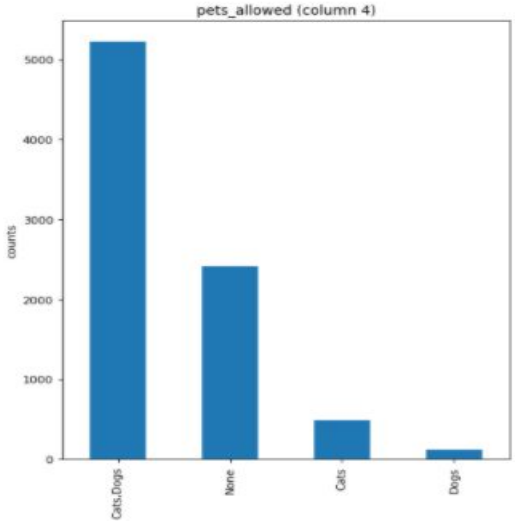
- Bathrooms
    - Ranges from 1 to 8 with a mean of 1.38

- 34 missing values; here, we set bathrooms=bedrooms (based on regression - not shown - with $R^2 = 0.46$)
-

- Bedrooms
  - Ranges from 1 to 9 with a mean of 1.78
  - 205 missing values (of which 7 are blank and 198 coded as “0” which is not possible). We dropped all 205 values.

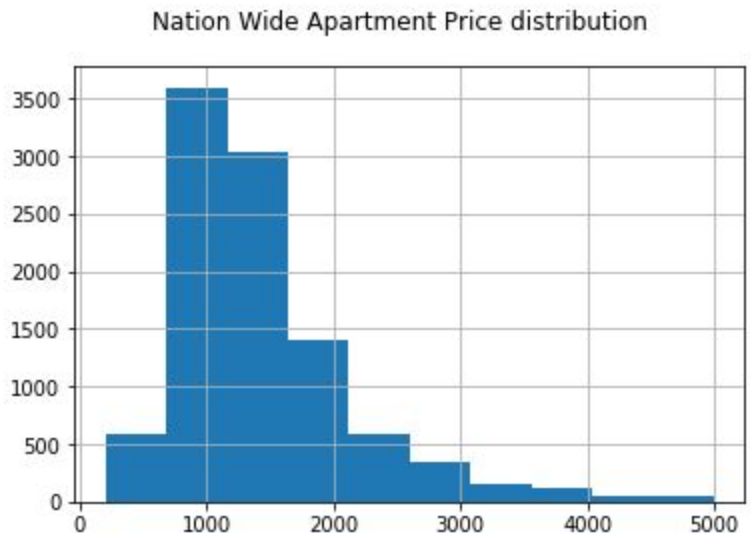


- Pets_Allowed
  - Listed as none, cats, or ‘cats,dogs’. There are also 1748 coded as ‘null’ which we retained by assuming ‘none’.

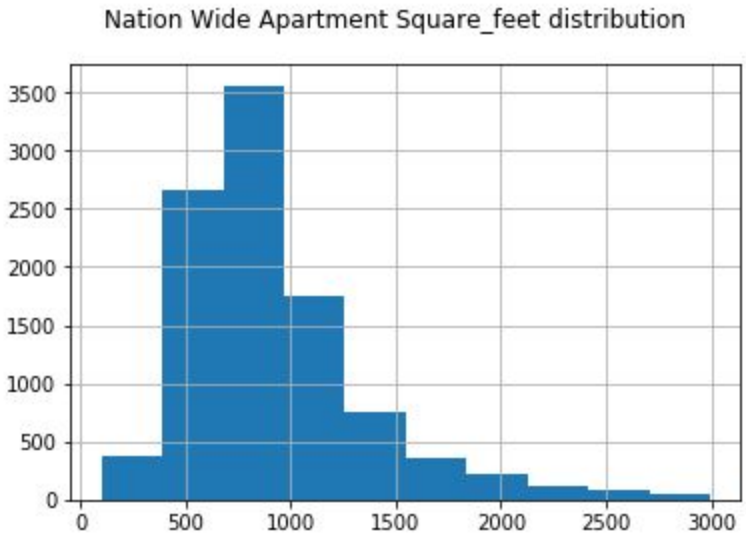


-

- Price
    - Ranges from $200 to $52,500 with a mean of $1486; No missing values
    - The $52,500/month apartment, an outlier, was miscoded since the description lists price as $500/month. We dropped this (actually was dropped because it also had bedrooms = 0)



- Square_feet
    - Ranges from 101 to 40,000 with an average of 946; no missing values

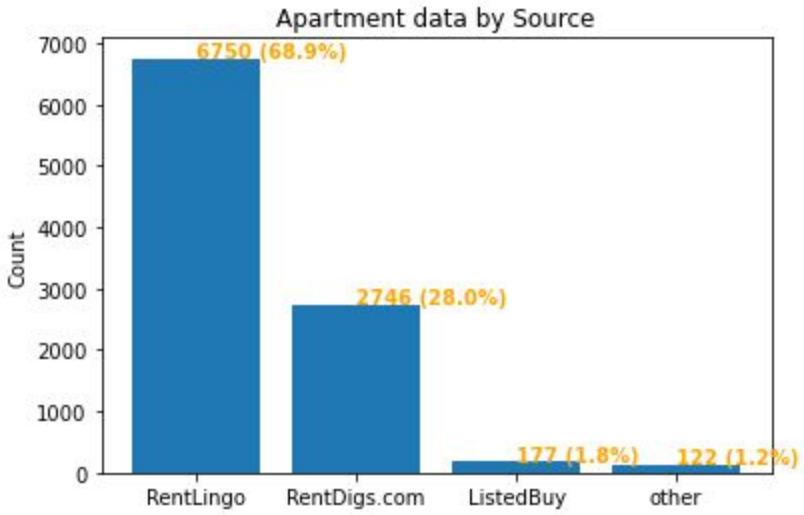


    -
- Latitude & Longitude

- 10 missing values; we replaced the missing values with sentinel values for the North Pole

We also eliminated four records that appeared to be rooms for rent (discussed in Appendix A).

One of the attributes we did not use but still wanted to investigate was the source from which the creator of the data scraped. As shown in the figure below, the vast majority were from RentLingo (69%) and RentDigs.com (28%).



In addition, we supplemented the primary apartment for rent dataset with an additional dataset[2] listing all Starbucks locations in the world in order to explore whether nearby coffee shops might be statistically significant in predicting apartment pricing. This data includes latitude and longitude which helped us in determining how many Starbucks lie within an x (ex: 5) mile radius of each apartment.

III. Experimental Design

[2] https://data.world/data-hut/starbucks-store-location-data

The following is our step-by-step process from obtaining the data to finding results.

1. Obtain data

   We browsed several sources including the UCI Data Repository, Kaggle, and Data.World to find a dataset that would allow us to apply a range of data science techniques. As discussed above, we settled on apartment rent data. Since the data had not been uploaded to the UCI Data Repository, we obtained it directly from the individual who sourced it (that individual employed web-scraping.)
2. Examine & clean the data
   a. The primary source data contained 22 attributes and 10,000 rows. As discussed above and in the Appendix, we retained the 8 of the attributes we felt would be most useful for analysis. We examined and addressed missing data and outliers.
   b. The secondary source data continued Starbucks locations with latitude and longitude. We obtained this from Data.World.
3. Read the two CSV files into 2 dataframes in Python
4. Append a column to the dataframe with the number of Starbucks within an x (ex: 5) mile radius of each apartment.
5. Query and analyze the data (including Unit Testing of our code) in the following ways:
   a. Regression analysis with price as the dependent variable
   b. Visualize and descriptive statistics
   c. Prediction models for determining which city a listing is in based on price, bedrooms, bathrooms, and square feet
   d. Prediction models for the number of bedrooms based on price and bathrooms
   e. Determine general relationship between the number of bedrooms and number of bathrooms in apartments
   f. General bar and scatter plots to understand relationship between variables using df display GUI

6. Write up our results with tables & visualizations

IV. Beyond the Original Specification

Here, we provide a high-level description of the ways we went above and beyond the original project specification. We outline the techniques here and provide deeper analysis and insights in the subsequent Results section.

A. Regression Analysis:

a. Merging data that lacks a primary/foreign key relationship
In particular, we sought to add the # of Starbucks locations within an x mile radius of each apartment in our primary data set. Our first attempt at this used a lambda function that computed the distance from each Starbucks to each apartment on a row-by-row basis. This had a 448 minute run time. We then switched to using "map" but only shortened run time to 446 mins. Finally, we switched to a vectorized approach[3] that cut run time to 9 seconds - a nearly 3,000x improvement!

b. Implementing the Havershine algorithm to compute geographic distance

c. Using forward selection to determine a regression model given a large number of independent variables

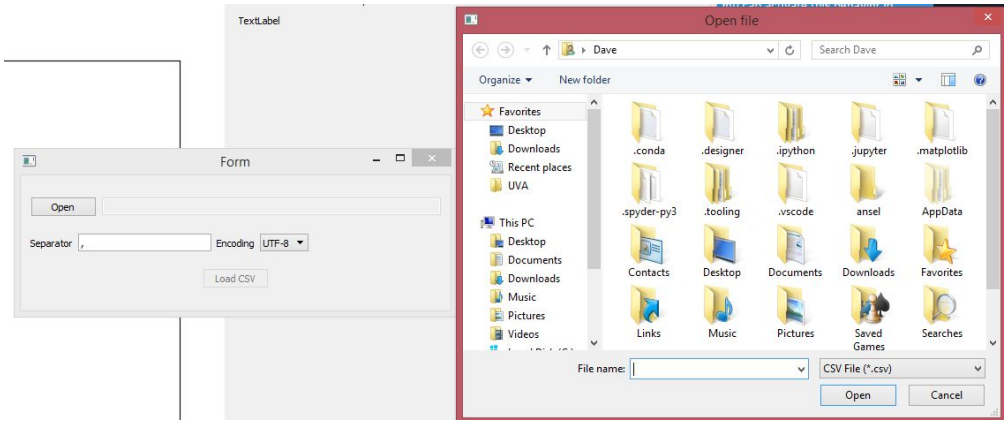
d. Testing for multicollinearity between quantitative independent variables and for association between qualitative independent variables

[3] https://godatadriven.com/blog/the-performance-impact-of-vectorized-operations/

e. Collecting user input: We ask the user to input the radius around which we will count the number of Starbucks near an apartment

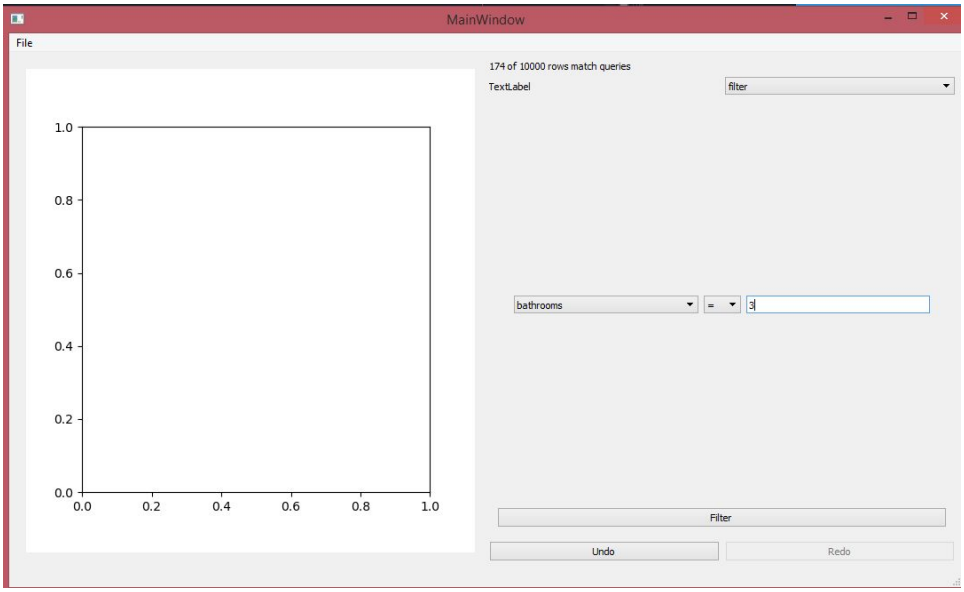
f. Implemented a binary search algorithm that find the maximum R^2 as a function of search radius (i.e. the maximum of an inverse parabola)

B. Visualization-DF Display:
   a. DF Display is a GUI we created making use of PyQT, PyQT designer (included in anaconda), and matplotlib.
   b. The purpose of the GUI is general and not specific to our data. It allows for users to open a csv file and read it into a dataframe, filter by different column values, and plot trends without having to know pandas or even python themselves.
      i. These plots can then be exported from within the application and included as references for further investigation.
   c. It uses pandas to read in a dataframe from a user selected CSV file

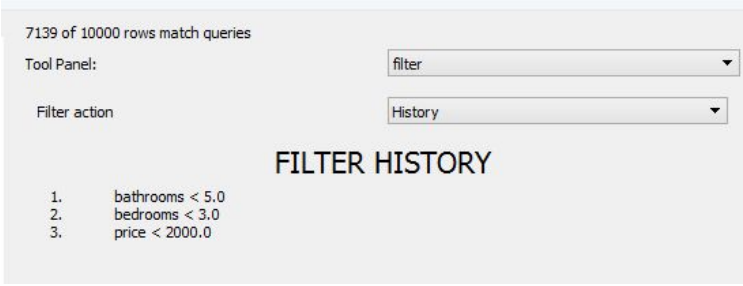


   d. It allows for data to be filtered down using user selected queries on columns.
      i. Allows Redo and undo of queries

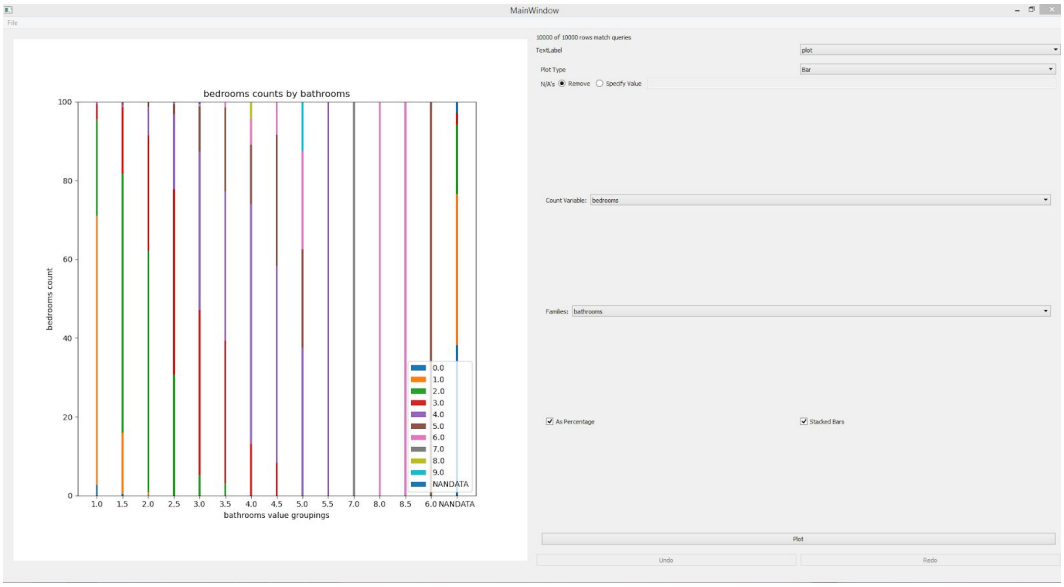
ii. Uses dictionary representations of filter queries for storing filter history, this prevents storing multiple large dataframes in memory.



iii. Provides a convenient history of the filters done on the data



e. Allows creation of bar and scatter plots based on user selected columns and filters. This includes stacked bar charts and normalizing the bars to percentages of total.



i. The plots are then exportable directly from the tool

C. K Nearest Neighbors

a. Use of SKLearn package to create K nearest neighbors model

b. Data split up into training set and prediction set

c. Selection and reselection of features to include in the model to discover highest accuracy model

V. Results

We began with a number of interesting queries as follows:

1. What is the mean price of an apartment?

```
# 1. What is the average price of an apartment
meanPrice=df1['price'].mean(axis=0,skipna=True)
print ("\nThe mean apartment price is: {:.1f}".format(meanPrice))
```

```
The mean apartment price is: 1483.1
```

2. What percentage of apartments have wood floors

```
# 2. What percentage of apartments have wood floors
print ("\nPercent of apartments with wood floors: {:.2%}"
       .format(df1[df1['Wood Floors']==1].shape[0]/df1.shape[0]))
```

```
Percent of apartments with wood floors: 3.58%
```

3. What is the median number of starbucks nearby

```
# 3. What is the median number of starbucks nearby
print("\nThe median number of Starbucks within a {} mile radius is: {}"
      .format(radius_to_starbucks_in_miles,
              df1['starbucksCount'].median(axis=0,skipna=True)))
```

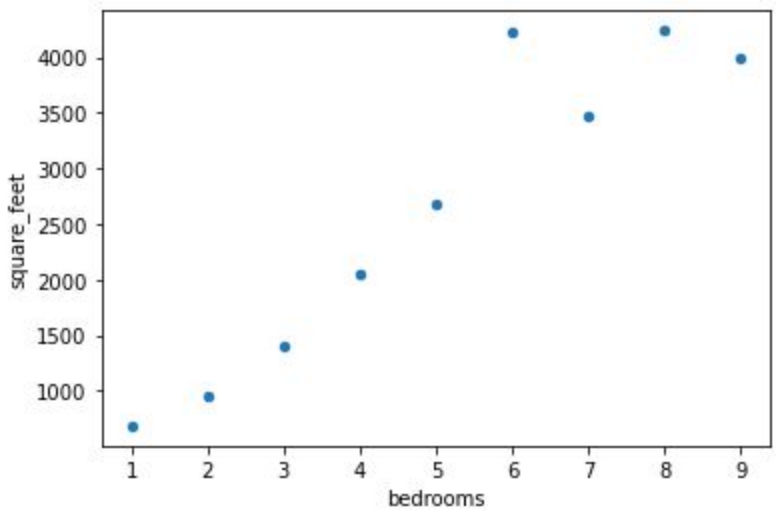
```
The median number of Starbucks within a 5.0 mile radius is: 9.0
```

4. What is the average sq ft by # number of bedrooms

```
# 4. What is the average sq ft by # number of bedrooms
print("\nThe average sq ft by # of bedrooms is as follows:")
df_bedroom_sqft=df1.sort_values(by=['bedrooms']).groupby('bedrooms',as_index=False).square_feet.mean()
print(df_bedroom_sqft)

df_bedroom_sqft.plot(kind='scatter',x='bedrooms',y='square_feet')
```

```
The average sq ft by # of bedrooms is as follows:
   bedrooms  square_feet
0       1.0   683.282396
1       2.0   959.648028
2       3.0  1408.784483
3       4.0  2046.700495
4       5.0  2684.674157
5       6.0  4234.466667
6       7.0  3471.000000
7       8.0  4240.000000
8       9.0  4000.000000
```



Next we built a regression model to predict price for the potential regressors, including many categorical variables for amenities.

We were worried about multicollinearity of quantitative independent variables as well as association between qualitative independent variables. The correlation matrix (shown below) reveals a high correlation between bedrooms and bathrooms. We left both variables in to allow our forward selection model to choose the better of the two.

```
Investigating multicollinearity in quant predictors...
             bedrooms  bathrooms  square_feet
bedrooms     1.000000   0.710458     0.586645
bathrooms    0.710458   1.000000     0.632872
square_feet  0.586645   0.632872     1.000000
```

Using the Cramer's V test to explore categorical variable association, we found (shown below) high association between (Dishwasher, Refrigerator) and between (cats, dogs). We dropped regressors for Dogs, and Refrigerator.

```
Investigating associate in categorical predictors...
... outputting categorical variable pairs with Cramers V above 0.7
high association: Dishwasher , Refrigerator 0.72
high association: Cats , Dogs 0.88
```

Next, we built a regression model using forward selection given a radius of 5 miles from each apartment to determine the number of local Starbuck stores.

```
Determining linear model using forward selection...
price ~ bathrooms + starbucksCount + square_feet + Elevator + Cats + Garbage_Disposal + Doorman
+ Playground + bedrooms + AC + Hot_Tub + Wood_Floors + Washer_Dryer + Storage + Gym + Fireplace
+ View + Parking + Cable_or_Satellite + Luxury + Golf + Pool
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.420
Model:                            OLS   Adj. R-squared:                  0.418
Method:                 Least Squares   F-statistic:                     321.3
Date:                Fri, 30 Oct 2020   Prob (F-statistic):               0.00
Time:                        11:54:33   Log-Likelihood:                -78412.
No. Observations:                9795   AIC:                         1.569e+05
Df Residuals:                    9772   BIC:                         1.570e+05
Df Model:                          22
Covariance Type:            nonrobust
```

The resulting coefficients with radius = 5 miles were as follows:

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	145.6747	24.200	6.020	0.000	98.237	193.112
bathrooms	520.6151	18.448	28.221	0.000	484.454	556.776
starbucksCount	17.0673	0.361	47.257	0.000	16.359	17.775
square_feet	0.3273	0.015	21.679	0.000	0.298	0.357
Elevator	304.6546	32.144	9.478	0.000	241.646	367.663
Cats	-97.6353	15.478	-6.308	0.000	-127.976	-67.295
Garbage_Disposal	-143.3122	26.703	-5.367	0.000	-195.656	-90.969
Doorman	-853.8678	143.880	-5.935	0.000	-1135.902	-571.833
Playground	-126.6859	29.730	-4.261	0.000	-184.963	-68.409
bedrooms	41.3237	11.970	3.452	0.001	17.860	64.787
AC	88.7052	33.462	2.651	0.008	23.114	154.297
Hot_Tub	105.0208	41.865	2.509	0.012	22.958	187.084
Wood_Floors	97.6203	41.005	2.381	0.017	17.242	177.999
Washer_Dryer	-66.8581	28.647	-2.334	0.020	-123.013	-10.704
Storage	42.7472	22.376	1.910	0.056	-1.114	86.608
Gym	61.8848	25.682	2.410	0.016	11.542	112.227
Fireplace	-49.3220	25.758	-1.915	0.056	-99.812	1.168
View	117.8042	61.835	1.905	0.057	-3.405	239.013
Parking	28.6728	17.574	1.632	0.103	-5.776	63.122
Cable_or_Satellite	-32.4069	24.518	-1.322	0.186	-80.467	15.653
Luxury	270.0727	219.299	1.232	0.218	-159.798	699.943
Golf	166.7793	152.876	1.091	0.275	-132.890	466.448
Pool	-20.6811	18.988	-1.089	0.276	-57.901	16.539

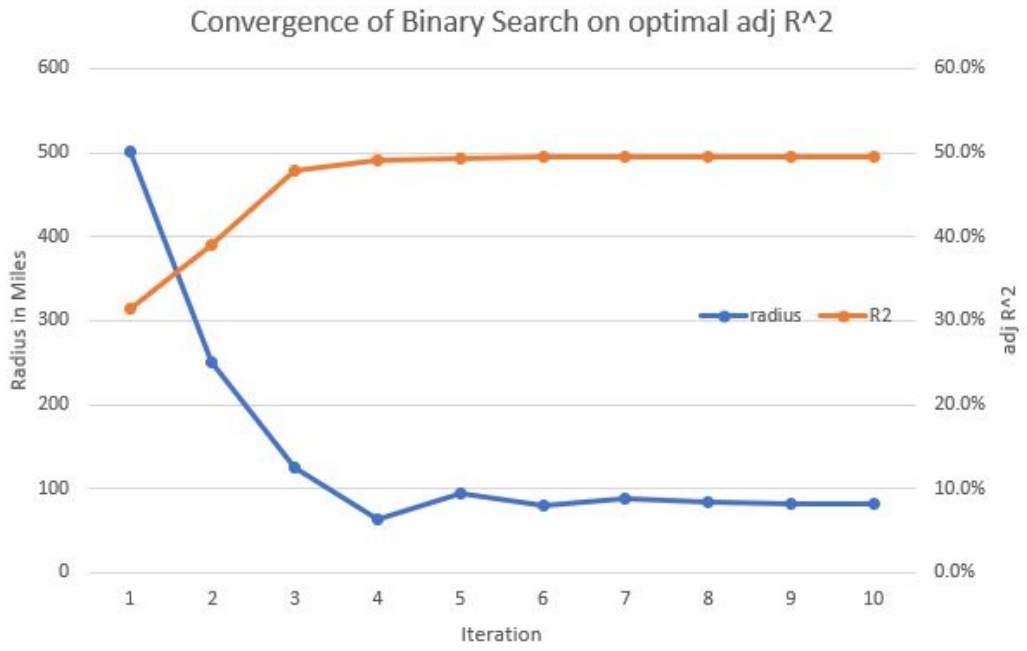
We were particularly interested in whether or not the number of local Starbucks would be a statistically significant predictor. Indeed, forward selection found it to be the second strongest factor (behind bathrooms). Each local Starbucks within a 5 mile radius is associated with a $17.07 increase in monthly rent. We do not necessarily think there is a causal relationship here; instead, our hypothesis is that Starbucks are cited highly desirable locations. Moreover, there are more Starbucks locations in areas with higher population density and it is the population that ‘causes’ the increase in demand.

As noted above, the adj R^2 with radius=5 miles is 41.8%. By comparison, we repeated the forward selection algorithm excluding the starbucksCount variable and found adj R^2 drops to 28.6%. This strengthened our conviction in using the Starbucks count as a predictor.

In order to optimize the apartment pricing formula, we created a binary search algorithm to find radius that maximizes adj R^2. We expect adj R^2 to be an inverse

parabola as a function of radius; when radius=0, we know adj R^2 = 28.6%; we ran with a radius of 1000 miles and found an adj R^2 = 30.1%. With a large radius, the number of Starbucks should be irrelevant. (Note: This approach would not be appropriate if adj R^2 as a function of radius was expected to have multiple local maxima rather than one global maximum).

Scanning integer radii between 1 and 1000 miles, our binary search algorithm found the maximum in 10 iterations, matching our expectation of log2(1000)=10. The converge was rapid as shown below:



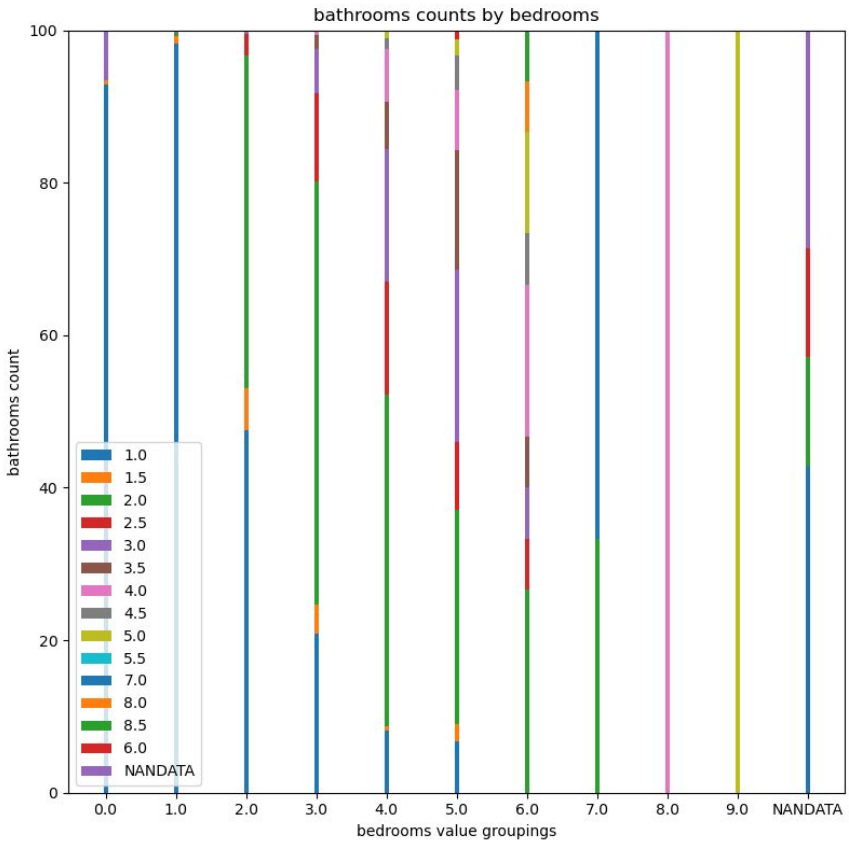
Our binary search algorithm found the optimal radius to be 81 miles with an adj R^2 = 49.57%. This radius was much further than we expected - our hypothesis was that the radius should be within walking distance if the relationship were more causal. This further confirms our suspicion that the Starbucks count is a proxy for other factors, especially population density.

The resulting apartment pricing model coefficients with a optimized radius of 81 miles is as follows:

	coef
Intercept	-15.3706
bathrooms	431.3190
starbucksCount	1.9937
square_feet	0.3483
Playground	-200.1977
Elevator	318.9453
bedrooms	62.2086
Fireplace	-79.1653
AC	-106.3822
Wood_Floors	161.9751
Doorman	-392.7397
View	173.4662
Garbage_Disposal	-99.0940
Dishwasher	60.4391
Basketball	-97.9349
Washer_Dryer	-60.4141
Gated	-73.1136
Internet_Access	61.1978
Cats	-25.3492
Cable_or_Satellite	-49.2822
Golf	186.5140
Pool	-32.6662
Clubhouse	38.7049
Parking	23.3612

Results from Advanced Techniques

We wanted to see how to fill any missing bathroom data. We used DFDisplay to create stacked bar charts of the distribution of bathrooms for each value of bedrooms.



From the graph, we saw that the value of bathrooms was typically equal to or lower than the number of bathrooms, and thus for missing data, we could set the number of bathrooms equal to the number of bedrooms unless the data was missing for both columns.

For our first K nearest neighbors model, we attempted to predict if the city the listing was in was for New York, San Francisco, or Other. We used 1000 rows as our training data, and then ran the prediction on the remaining data. This yielded a prediction accuracy of 91%, however since our data had so many different cities, we could have had a 99.3% accuracy just by guessing”Other” for every entry. This result showed that we needed to try to model something else.

```
Attempt to use KNN to predict if a listing is in New York, San Francisco, or other
The Accuracy of the predicted model: 0.910062535531552
The Accuracy if we only guessed other would be: 0.9930642410460488
```

We attempted to make a second prediction model and instead predict the number of bedrooms for a listing given the price, location, and presence of listed amenities. Those 3 columns were used as the features of the training set for the first 1000 listings. Using those features for the rest of the data, the model predicted the number of bedrooms with a 41.99% accuracy.

```
Now we will attempt to predict bedrooms instead
Accuracy of bedroom predictions: 0.4198976691301876
```

Following these attempts at models, we inferred that we needed a more balanced dataset to get better prediction accuracy. We used groupby and size to find the most common cities appearing in our dataset.

	cityname	counts
68	Austin	509
347	Dallas	216
1238	San Antonio	180
630	Houston	178
788	Los Angeles	150
269	Chicago	140
805	Madison	120
1130	Portland	104
365	Denver	104
677	Kansas City	99

From these results, we immediately noticed that the top 4 cities were all in Texas. With this knowledge, we created a subset dataframe only containing those 4 cities and attempted to create a K nearest neighbor model for predicting which Texas city an apartment listing was for. The features used were price and square feet. The model obtained a 44.32% accuracy, a modest improvement over just guessing the most common city, Austin, which would have yielded a 40.86% accuracy.

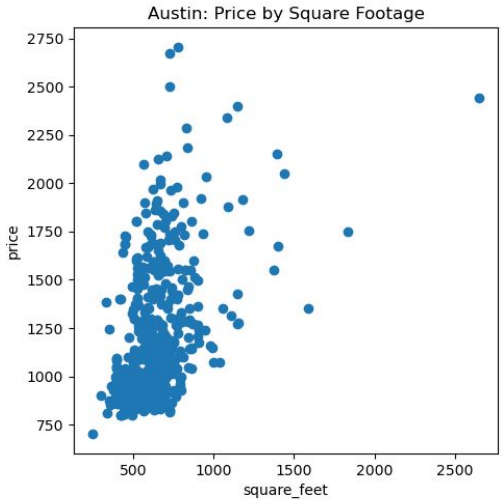
```
The top 4 cities are all in Texas, lets make a dataframe with only the Texas entries
Now lets try to predict which city a listing is for in Texas
Accuracy: 0.44316730523627074
Accuracy if guessing most common city: 0.4086845466155811
```

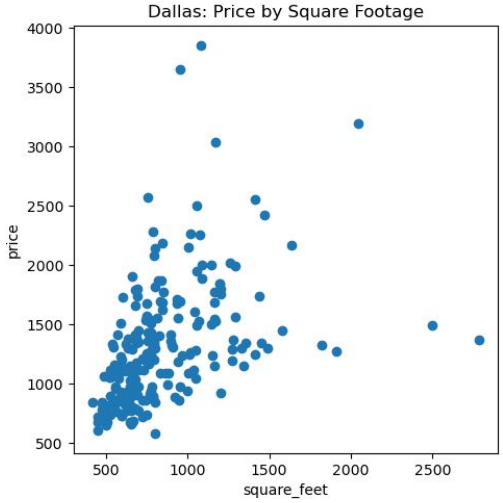
Leaning on the improvement of the model found by including nearby starbucks, we added the number of starbucks in a 81 mile radius as a feature of

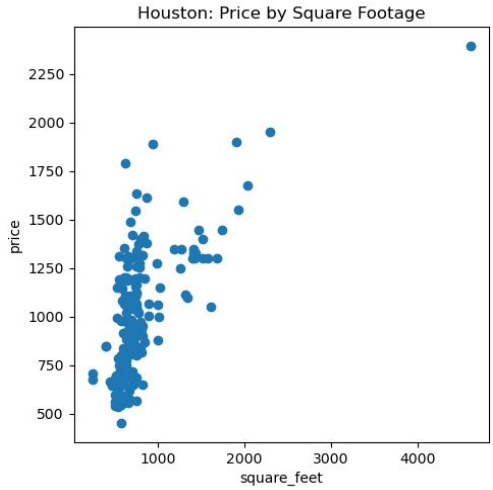
the data. This inclusion had a huge impact on accuracy, raising the accuracy of the predictions to 56.58%

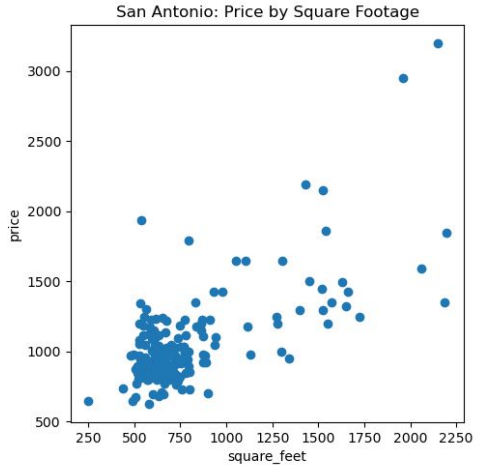
```
Now lets try to predict which city a listing is for in Texas with  starbucks in 81.0 mile
radius
Accuracy: 0.565772669220945
Accuracy if guessing most common city: 0.4086845466155811
```

DFDisplay was used to create plots for the four cities of price against square feet. These plots show that the general pattern for each city is slightly different, which is why without the nearby Starbucks as a predictor, the model was able to produce results better than random chance.









VI. Testing

We created unit tests for the following:

1. sb_in_range_vec_TestCases: Unit tests for our function that computes whether two geographic locations are within a given radius of each other. We tested if it works for:
    a. Two scalar locations within the desired radius
    b. Two scalar locations outside of the desired radius
    c. A set of vectors given as locations (this was most important because it gave us a 3000x increase in run speed!

2. radius_is_valid_TestCases: Unit tests for checking if an input is a positive (non-zero, non-negative) float. As a result of this testing, we modified the code to accept string and float/integer input types
    a. Test_false_when_blank
    b. test_false_when_string
    c. Test_false_when_negative
    d. Test_false_when_zero
    e. Test_true_when_pos_integer
    f. Test_true_when_pos_float

Sample output from our unit tests are provided here: (note: the function radius is valid is verbose hence the “try again,”s)

```
Python 3.8.3 (default, Jul  2 2020, 17:30:36) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.18.1 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/bestr/OneDrive/_UVA/_CS 5010/Group Project/
uva_group_proj_UnitTests.py', wdir='C:/Users/bestr/OneDrive/_UVA/_CS 5010/Group Project')
.......... try again, the radius must be a number
... try again, the radius must be a number
... try again, the radius must be >0
... try again, the radius must be >0
[1 1]

----------------------------------------------------------------------
Ran 9 tests in 0.007s

OK

In [2]:
```

VII. Conclusions

A. Key Findings and Use Cases

- Given an adj R^2 = 49.57% (moderately high), we would recommend our model as directionally correct for landlords and tenants to use when signing & renewing leases.

- In the presence of other factors (to confirm we would recommend simple linear regression), the following are positively correlated with pricing and are therefore recommended improvements for landlords to make: bathrooms; citing near Starbucks; larger sq. ft; elevator; bedrooms; wood floor; view; dishwasher; internet access. Notably, most of these are 'interior to the apartment' improvements.
- In the presence of other factors (to confirm we would recommend simple linear regression), the following are negatively correlated with pricing and therefore not recommended: playground; fireplace; AC (investigate further); doorman (investigate further); garbage disposal; basketball; washer/dryer (investigate further); gated (investigate further). Most of these are external and/or generate noise (garbage disposal) or risk (fireplace). We suspect those we labeled as "investigate further" would turn positive in simple linear regression and are thus conflated with other factors in the multiple regression.

B. Future Improvements (if we had more time)
    1. Integrate additional predictors into our regression model Including but not limited to[4]: population density; city ranking; median household income; racial composition; crime rates; etc.
    2. More carefully explore non-linearity and/or interactions between independent variables in our regression model
    3. Test the validity of our pricing model with out-of-sample data
    4. Implement a vectorized version of the geodesic algorithm (more accurate than Haversine)
    5. Explore more sophisticated regression models beyond forward selection to predict price
    6. Implement natural language processing to extract keywords from the "body" attribute (a free form text description of each apartment)

[4] https://simplemaps.com/data/us-cities

VIII. Appendix

A. Unused Attributes from the apartments for rent data set

- Id: unique identifier of apartment
  - We verified there were no duplicates & no missing values
- Category: category of classified
  - There are no missing values
  - There are three categories: housing/rent/apartment (n=9999); housing/rent/home (n=2); housing/rent/short-term (n=2). The latter two categories appear to be rooms for rent so we will exclude them from the analysis.
- title = title text of apartment
- body = body text of apartment
- currency (all = 'USD')
- Fee (all = 'No')
- Has_Photo
  - Three categories: thumbnail (n=8907); yes (n=909); no (n=184). There is no missing data.
  - Since "no" is uncommon, we excluded from analysis though our hypothesis is that apartments without photos may have lower prices all else being equal.
- Price_Type (all = 'monthly')
- Address; City_Name; State: Unnecessary since we have latitude and longitude
- source = origin of classified
- time = when classified was created (notably, we were unable to decode this)

B. Links to CS 5010 Resources

- Project Instructions PDF