# CS 5010 Group Project

# Analyzing Apartment Data Across the US

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

# Introduction

# We had 2 research objectives

1. Predict apartment price given attributes from two independent data sets
   Application: To help lessors set and renters pay fair market prices

2. Understand how attributes of the ‘average’ apartment varies by city
   Application: To help people moving to new cities decide where they are most likely to get the housing they desire.

# The Data

We combined data from two sources

# Our primary source contained 22 attributes for 10,000 apartments

UCI

**Machine Learning Repository**

Center for Machine Learning and Intelligent Systems

## Apartment for rent classified Data Set

*Download*: Data Folder, Data Set Description

**Abstract**: This is a dataset of classified for apartments for rent in USA.

| | | | | | |
|---|---|---|---|---|---|
| **Data Set Characteristics:** | Multivariate | **Number of Instances:** | 10000 | **Area:** | Business |
| **Attribute Characteristics:** | N/A | **Number of Attributes:** | 22 | **Date Donated** | 2019-12-26 |
| **Associated Tasks:** | Classification, Regression, Clustering | **Missing Values?** | N/A | **Number of Web Hits:** | 11780 |

**Source:**

Collected from Internet 2019-12-28 for an Machine learning task and I want to share this dataset with all who is interested to use it.
For any questions about the dataset feel free to contact me on fredrick_nilsson '@' yahoo.comthe names, email addresses, institutions, and other contact information of the donors and creators of the data set.

**Data Set Information:**

The dataset contains of 10'000 or 100'000 rows and of 22 columns The data has been cleaned in the way that
column price and square_feet never is empty but the dataset is saved as it was created.

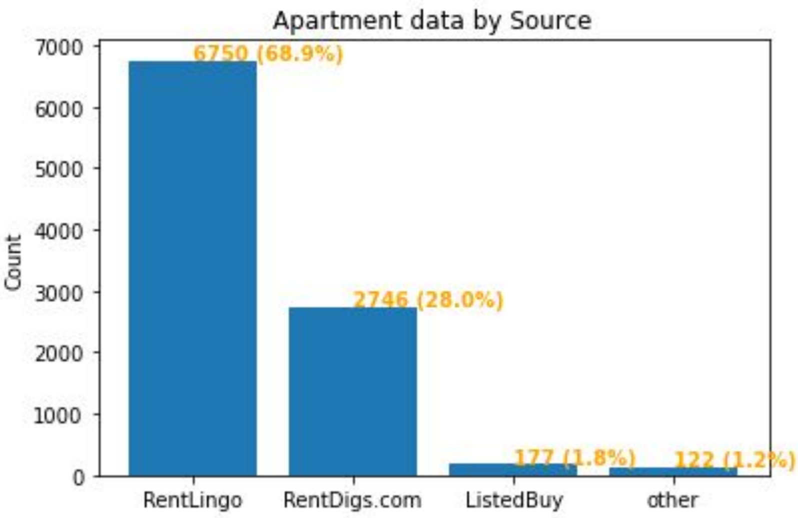
Can be used for different machine learning tasks such as clustering, classification and also regression for the squares feet column

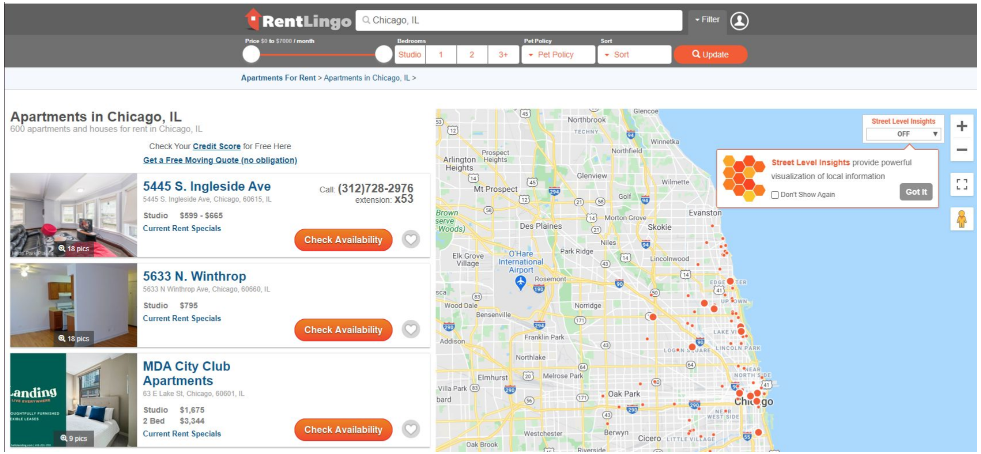
**Attribute Information:**

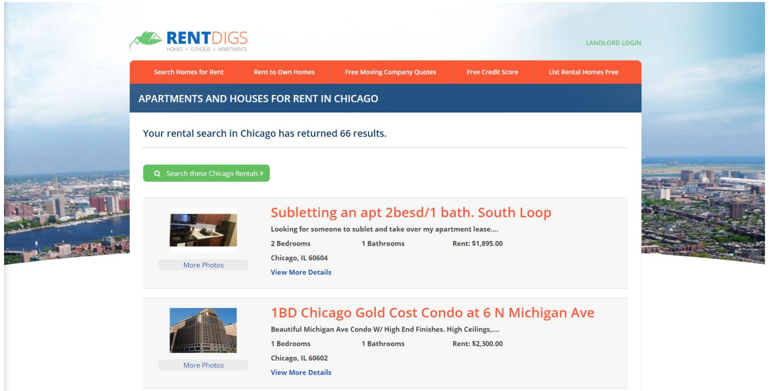
Provide information
id = unique identifier of apartment
category = category of classified
title = title text of apartment
body = body text of apartment
amenities = like AC, basketball,cable, gym, internet access, pool, refrigerator etc.
bathrooms = number of bathrooms
bedrooms = number of bedrooms
currency = price in current
fee = fee
has_photo = photo of apartment
pets_allowed = what pets are allowed dogs/cats etc.
price = rental price of apartment
price_display = price converted into display for reader
price_type = price in USD
square_feet = size of the apartment
address = where the apartment is located
cityname = where the apartment is located
state = where the apartment is located
latitude = where the apartment is located
longitude = where the apartment is located
source = origin of classified
time = when classified was created
bout each attribute in your data set.

| 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 | | |

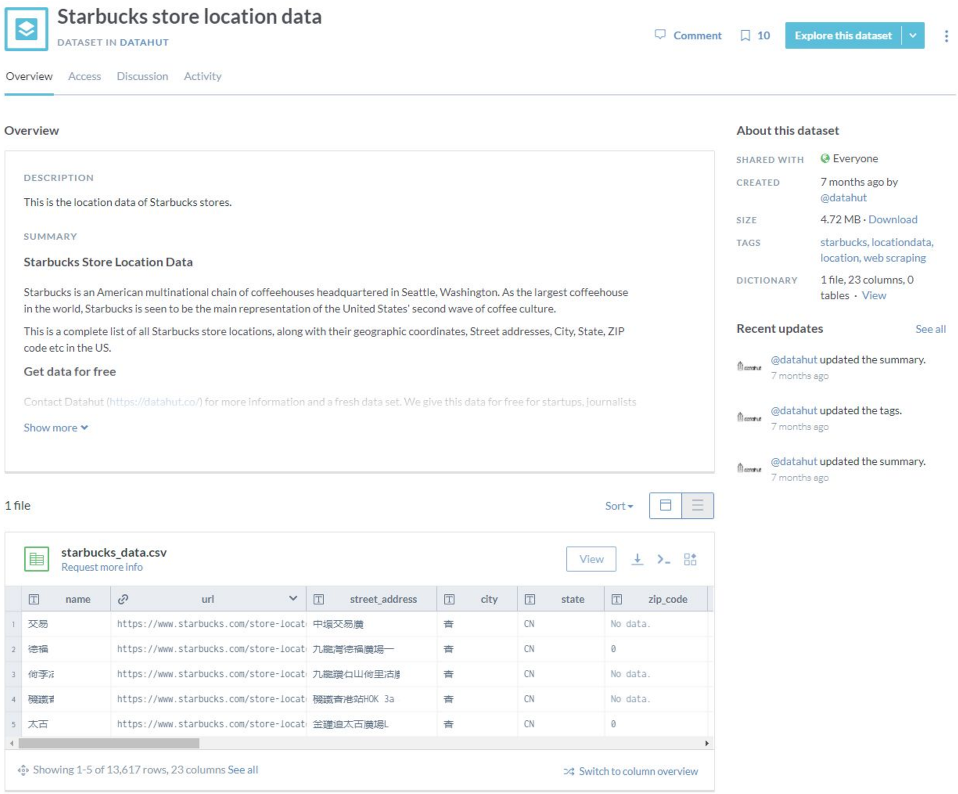
# 97% of the apartment data was scraped from 2 sites



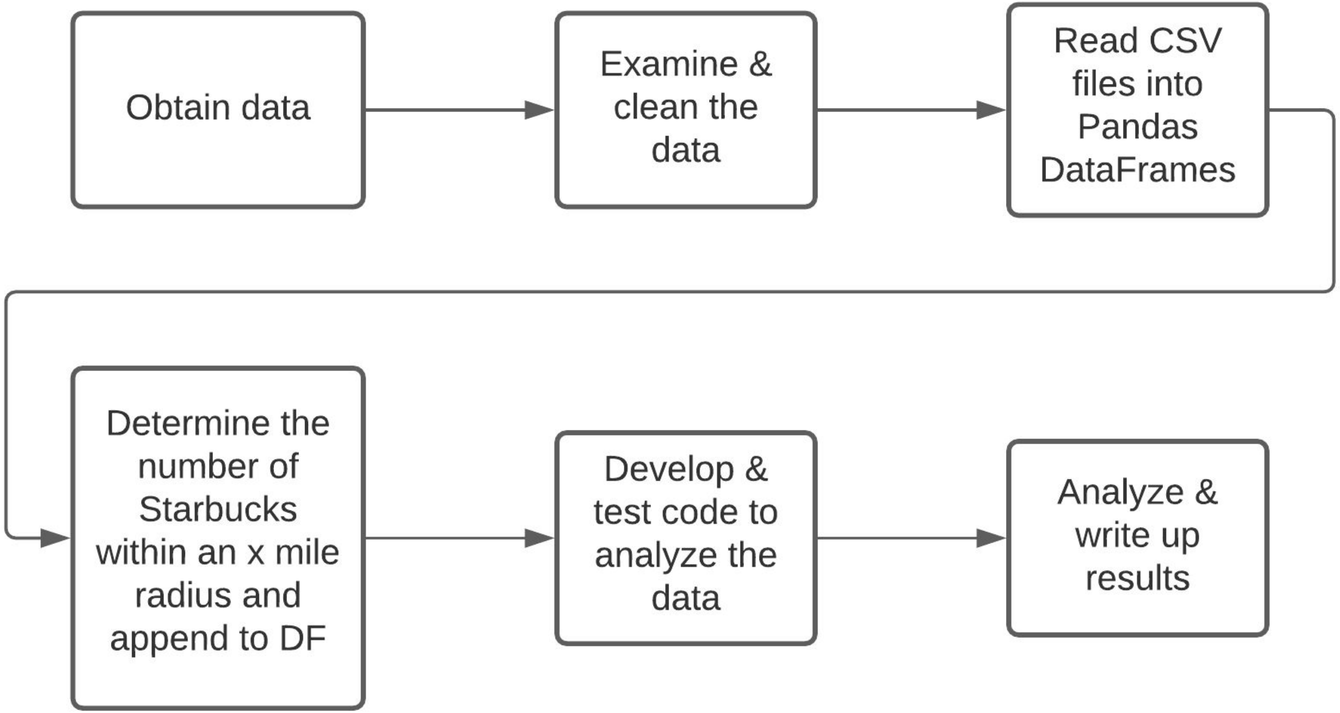



# We added the data from 13,617 Starbucks locations using data from Data.World



# EXPERIMENTAL DESIGN


Obtain data
Examine & clean the data
Read CSV files into Pandas DataFrames
Determine the number of Starbucks within an x mile radius and append to DF
Develop & test code to analyze the data
Analyze & write up results

# BEYOND THE ORIGINAL SPECIFICATIONS:

# We went beyond the original specification in 4 ways

1. Merging multiple datasets using vector-based logic
2. Regression analysis to predict apartment pricing
3. DataFrame visualization via GUI
4. K-Nearest Neighbors analysis

# 1. Merging multiple datasets using vector-based logic

To determine the number of Starbucks within an x mile radius of each apartment, we started with a row-based lambda function:

But this had a 448 minute run time!

```
df1['starbucksCount']+=df1.apply(lambda row:
starbucksInRange(row['latitude'],row['longitude'],row['sLat'],row['sLon'],row['radius_to_starbucks_in_miles']),axis=1)
```

We tried map() but that had little impact → 446 min run time!

```
df1['starbucksCount']+=list(map(starbucksInRange,df1['latitude'].values,df1['longitude'].values,df1['sLat'].values,df1['sLon'].values,df1['ra
dius_to_starbucks_in_miles'].values))
```

# Vectorized computation yielded a 3,000x performance improvement

```python
def sb_in_range_vec(lat, lon, pcode_lat, pcode_lon,rad_in_miles):
    """
    Find the distance between (lat,lon) and the reference point
    (pcode_lat,pcode_lon).
    Source: https://godatadriven.com/blog/the-performance-impact-of-vectorized-operations/
    """
    RAD_FACTOR = pi / 180.0  # degrees to radians for trig functions
    lat_in_rad = lat * RAD_FACTOR
    lon_in_rad = lon * RAD_FACTOR
    pcode_lat_in_rad = pcode_lat * RAD_FACTOR
    pcode_lon_in_rad = pcode_lon * RAD_FACTOR
    delta_lon = lon_in_rad - pcode_lon_in_rad
    delta_lat = lat_in_rad - pcode_lat_in_rad
    # Next two lines is the Haversine formula
    inverse_angle = (sin(delta_lat / 2) ** 2 + cos(pcode_lat_in_rad) *
                     cos(lat_in_rad) * sin(delta_lon / 2) ** 2)
    haversine_angle = 2 * arcsin(sqrt(inverse_angle))
    #EARTH_RADIUS = 6367  # kilometers
    EARTH_RADIUS = 3958  # miles
    distance=haversine_angle * EARTH_RADIUS
    in_range=(distance<=rad_in_miles)
    return in_range.astype(int)
```

This runs in 9 seconds given ~10,000 apartments and 13K Starbucks locations!

# 2. Regression analysis to predict apartment pricing

We implemented the forward selection algorithm to determine the optimal regression model given a large number of independent variables

In addition, we created a binary search algorithm to find the Starbucks search radius that maximizes adj. $R^2$ (i.e the maximum of an inverse parabola)

# 3. DataFrame visualization via GUI

# 4. K-Nearest Neighbors analysis

## Background

1. Large Original dataset (10,000 rows before cleaning)
2. Used SkLearn Package
3. Divided original data into training set (10%) and prediction set (90%)
4. Used k=3

# KNN: Can we predict a major city?

1. Divided our data into 3 categories: “New York”, “San Francisco”, “Other”
2. Used “hasAmenities” (1 if amenities present, 0 if not) and “price” as the features to look for neighbors for

```
#%%
print("\nAttempt to use KNN to predict if a listing is in New York, San Francisco, or other")
data=df1
data["location"] = ["other"] * len(data) #Create a basic data column prepopulated with other
data.loc[data['cityname'] == 'New York', 'location'] = "New York"
data.loc[data['cityname'] == 'San Francisco', 'location'] = "San Francisco"

#We use a labelencoder to convert our 3 categories into numbers
le = preprocessing.LabelEncoder()
encoded_location=le.fit_transform(data["location"])
list(le.classes_)
data["encodedloc"] = encoded_location
data["hasAmenities"] = [1] * len(data)
data.loc[pd.isna(data['amenities']), 'hasAmenities'] = 0

#The features of the neighbors are hasAmenities and Price
features=list(zip(data["hasAmenities"][0:1000],data["price"][0:1000]))
model = KNeighborsClassifier(n_neighbors=3)

# Train the model using the training sets, the first 1000 listings
model.fit(features,data["encodedloc"][0:1000])

#predict for the test set
actuals = list(zip(data["hasAmenities"][1000:9795],data["price"][1000:9795]))
predicted = model.predict(actuals)
correct_values = data["encodedloc"][1000:9795].tolist()

test_all_other = [2] * len(correct_values)
print("The Accuracy of the predicted model:", metrics.accuracy_score(correct_values, predicted)
print("The Accuracy if we only guessed other would be:", metrics.accuracy_score(correct_values,
#Having so much "other blows out the dataset, need to reevaluate
```

**Results:**

The Accuracy of the predicted model: 0.910062535531552

That’s great, however:

The Accuracy if we only guessed other would be: 0.9930642410460488

# KNN: Problems

Our data was incredibly unbalanced with “Other” = 9706/9795

**Solution:** Try to predict the number of the number of bedrooms now based on location, amenities, and price.

Accuracy: **0.4198976691301876, 41.99%**

Need to reevaluate our solution.

# KNN: Need to ask better Questions

```
#Let's find the top cities
sorted_counts_frame = data.groupby("cityname").size().reset_index(name='counts').sort_values(by=['counts'],ascending=False)
print(sorted_counts_frame.head(10))
```

```
         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
```

- First let's find the top cities in our dataframe
- We noticed the top 4 cities were all in Texas
- Filtered down the dataframe to only those 4 cities
- Divide into training and prediction set and attempt to predict city of the listing based on price and square feet
- Accuracy: 0.44316730523627074
- Accuracy if only guess Austin: 0.4086845466155811

# KNN: Coffee to the rescue

- Add the # of starbucks in an 81 mile radius to the apartment as a predictor
- ```
  features=list(zip(texas_data["price"][0:300],texas_data["square_feet"][0:300],texas_data["starbucksCount"][0:300]))
  ```
- Rerun the model
- Accuracy: 0.565772669220945
- Accuracy without starbucks: 0.44316730523627074
  ```

# KNN Takeaways

1. Machine learning and other modeling is a powerful analytical tool, but requires asking the right questions
2. Beware of biased and unbalanced data
3. Consider what a variable is really showing about a data set
   a. E.g. starbucks count being a pseudo-city identifier

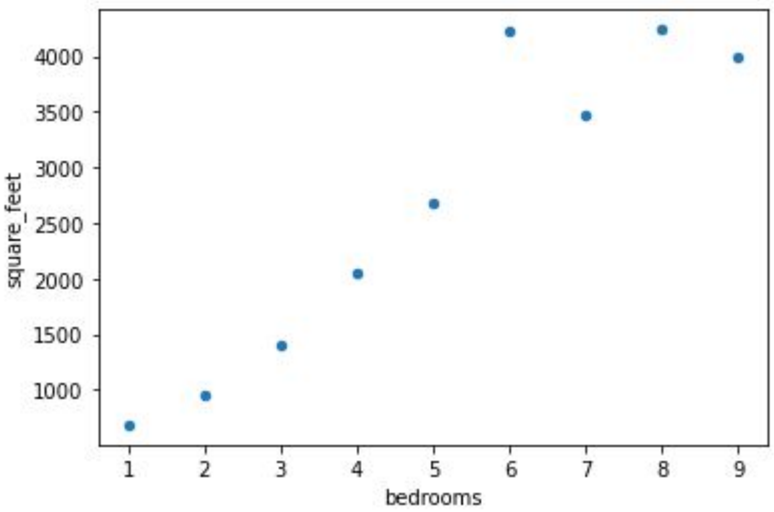
# RESULTS

# We began with a number of queries to understand the data. One was sq. ft by # 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
```



# We were also worried about multicollinearity for quant variables & association for qual variables

```
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
```

```
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
```

# Our regression model with Starbucks radius=5 miles gave an adj. R2 of 41.8%

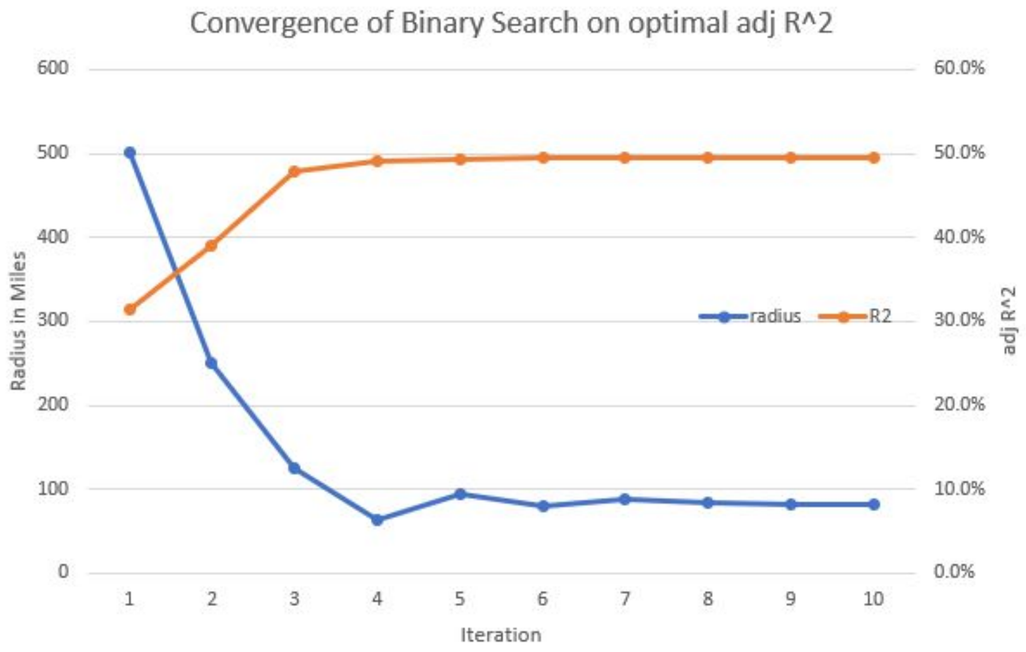
```
Determining linear model using forward selection...
price ~ bathrooms + starbucksCount + square_feet + Elevator + Cats + Garbage_Disposal + Doorman
+ Playground + bedrooms + AC + Hot_Tub + Wood_Floors + Washer_Dryer + Storage + Gym + Fireplace
+ View + Parking + Cable_or_Satellite + Luxury + Golf + 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
```

```
Determining R^2 without starbucksCount using forward selection...
Rsq without starbucksCount = 28.62%
```

# And the Starbucks count was the 2nd most important attribute (based on p-value)!

| | 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 found 81 miles to be the Starbucks radius that maximes $R^2$



# TESTING

# We created unit tests for our most critical functions

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

```
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]:
```

# CONCLUSIONS

1. Given an adj R2 = 49.57% (moderately high), we would recommend our model as directionally correct for landlords and tenants to use when signing & renewing leases.

2. 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.

3. 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.

Thank you