1) Describe your data set, why you chose it, and what you are trying to predict with it

Before choosing my dataset, I had the idea of predicting cryptocurrency prices, but it was advised that times series prediction was out of the scope of this course. Then, I looked into predicting Inspection Scores from Restaurants in Manhattan, and again, it was not a good choice because the dataset did not have enough features to come up with a good analysis. Finally, I took the professor's recommendation to work with the dataset from StreetEasy about Rental Prediction in Manhattan. This one was a good choice because it has enough features, and it is large enough to run the ML models introduced in the class. Also, I decided to work with it because I was curious about the fluctuations in the rent prices in Manhattan. The dataset is composed of 3,539 rows and 18 columns, and my target variable is predicting the rent prices in Manhattan.

```
manhattan = pd.read_csv('manhattan.csv')
manhattan.shape
#below we can see our dataset has 3539 rows and 18 columns

(3539, 18)
```

2) Detail what you did to clean your data and any changes in the data representation that you applied. Discuss any challenges that arose during this process.

Before proceeding with the data cleaning, I divided the dataset into the training and test set as indicated in the assignment instructions. My "y" variable as I mentioned before was the "rent", and my "X" was the everything, but "rent" since this variable was drop to because it is the target variable. Then, I used pandas.isnull().sum() to find the null values. Luckily, this

dataset did not have any nulls values, and I did not have to use IMPUTER in scikilearn for value replacement. In the pictures below, we can see the features of the dataset without any null values.

```
X_train_cleaned.isnull().sum() # not missing values

rental_id            0
bedrooms             0
bathrooms            0
size_sqft            0
min_to_subway        0
floor                0
building_age_yrs     0
no_fee               0
has_roofdeck         0
has_washer_dryer     0
has_doorman          0
has_elevator         0
has_dishwasher       0
has_patio            0
has_gym              0
neighborhood         0
borough              0
dtype: int64
```

```
X_train.info() #using info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 2123 entries, 1491 to 2732
Data columns (total 17 columns):
rental_id          2123 non-null int64
bedrooms           2123 non-null float64
bathrooms          2123 non-null int64
size_sqft          2123 non-null int64
min_to_subway      2123 non-null int64
floor              2123 non-null float64
building_age_yrs   2123 non-null int64
no_fee             2123 non-null int64
has_roofdeck       2123 non-null int64
has_washer_dryer   2123 non-null int64
has_doorman        2123 non-null int64
has_elevator       2123 non-null int64
has_dishwasher     2123 non-null int64
has_patio          2123 non-null int64
has_gym            2123 non-null int64
neighborhood       2123 non-null object
borough            2123 non-null object
dtypes: float64(2), int64(13), object(2)
```

After making looking at the shape of the data, I decided to drop the "borough" feature since the entire dataset is about Manhattan. I dropped this column from both my training and test sets. Another important step in the data exploration was using pandas.describe() to get the summary statistics of the data. It was another way to confirm that all my columns had the same number of count values. Then, I looked into the mean values of important features such as "bedrooms", "bathrooms", "has_elevator", "has_gym", and others since by intuition I thought the might critical features influencing the rent prices in the dataset.

```
X_train.describe() #using describe()
```

	rental_id	bedrooms	bathrooms	size_sqft	min_to_subway	floor
count	2123.000000	2123.000000	2123.000000	2123.000000	2123.000000	2123.000000
mean	5280.276967	1.357984	1.375412	944.876119	4.879416	11.898022
std	3316.279919	0.966555	0.603866	472.785072	5.285583	11.069380

To my surprise, the mean number of bedrooms and bathrooms was very similar with 1.35 for bedrooms and 1.37 for bathrooms. With this information we can assume most apartments in Manhattan have 1 and 2 bedrooms and bathrooms. Also, it was interesting to notice that the mean number for "min_to_subway" was 4.87. This means most apartments in Manhattan are relatively close to the subway stations.

3) Discuss what you learned by visualizing your data

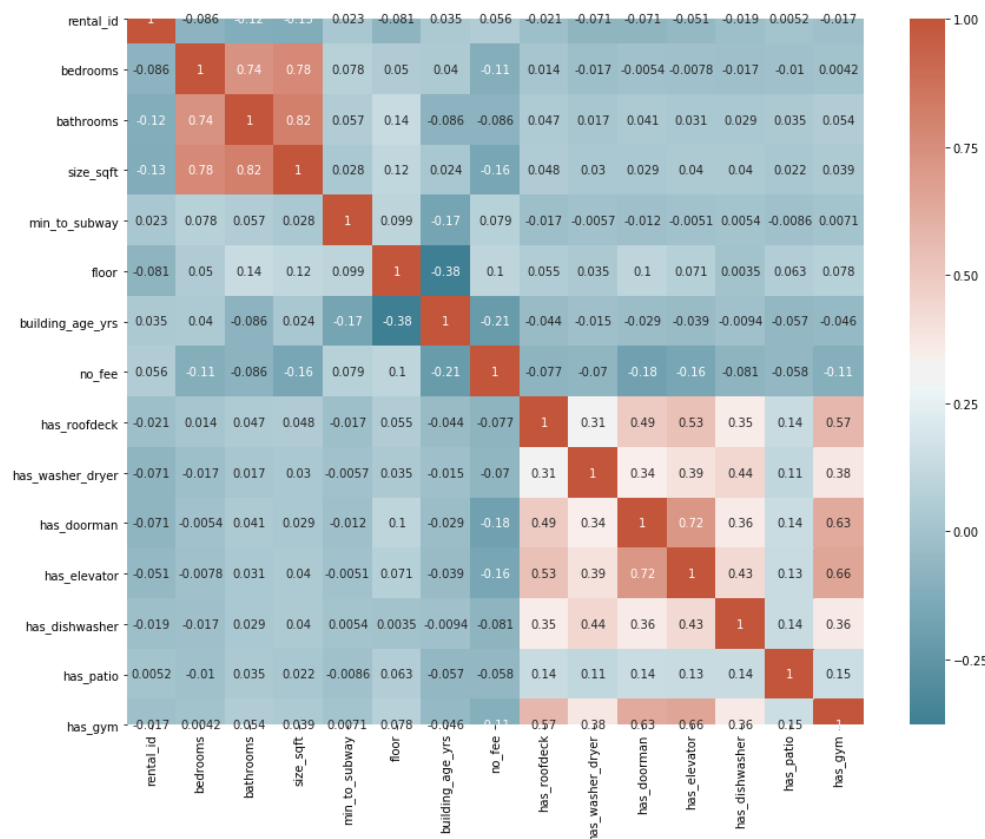In the picture we can see how the data looks when using pandas.hist():
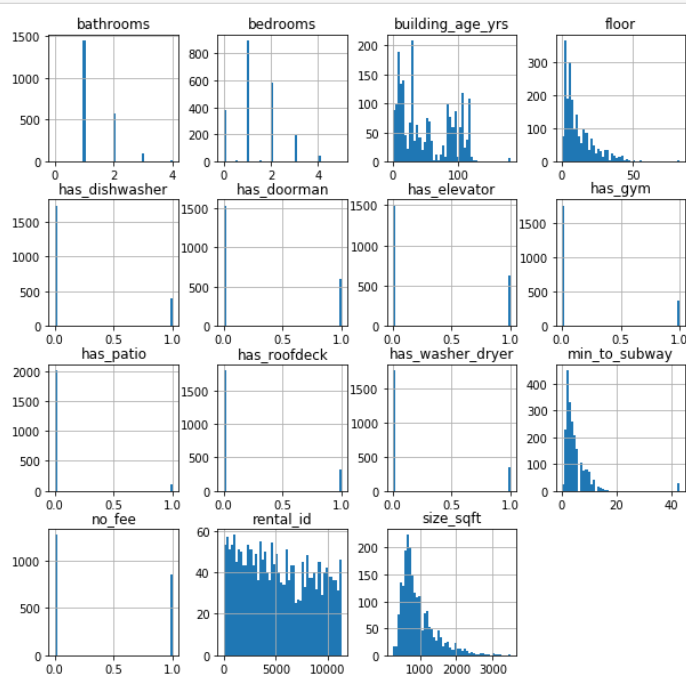
Here I can confirm my assumptions regarding the number of bedrooms and bathrooms in the dataset. According to the graph, most apartments have 1 and 2 bedrooms, as the same with the number of bathrooms. Looking at the "size_sqft" histogram we can see the data is positive skew to the right, and most apartments are around 1000 sqft with a mean value of 944. Also, it is surprising to see that the frequency of "building_age_yrs" is very high near to 100. We have around 25% of the dataset with buildings going over 100 years old where the mean is 51 years old. Another important fact to note is that very few apartments have gyms, elevators, doormen, patios, or dishwashers. Which makes sense because most of the apartments are relatively old. Years ago, having this

type of luxury was not a priority since it can be a factor in reducing the usable living space for the apartments.

I printed a correlation matrix to explore the significant levels among the features and the result was very intuitive. "Bedrooms", "bathrooms", and "size_sqft" were highly correlated among each other, which makes sense because the size will influence the number of bedrooms and bathrooms in the apartments. Also, surprisingly "has_doorman" and "has_elevator" had a high correlation

together with "has_gym" and "has_elevator".

Beside using the correlation matrix and the histogram, I also used the scatter_matrix from pandas, and the result was similar to the conclusions already mentioned above.

4) Describe your experiments with the two supervised learning algorithms you chose. This should include a brief description, in your own words, of what the algorithms do and the parameters that you adjusted. You should also report the relative performance of the algorithms on predicting your target attribute, reflecting on the reasons for any differences in performance between models and parameter settings.

The 2 supervised machine learning models I chose are *decision tree regressor* and *lasso regression*. I will start by defining how the *decision tree regressor* works. Decision trees can either by used for classification and regression problems. The methodology behind it works similar to an if/else statement where the "decision nodes" are the questions and the "leaves" are the answers. Then, the data is classified in groups and the algorithm learns from these decisions and makes the group classifications. I decided to use the decision tree regressor is because of its main advantages. It is very easy to understand by non-experts analyst like myself and picking one of the parameters such as max_depth, max_leaf_nodes, or min_samples_leaf is sufficient to prevent overfitting. Its work to mention that the disadvantage of this model is that it can result in poor generalizing performance even when the use of "pre-pruning" or choosing parameters.