Progress Report

Project Introduction

We are analyzing data about apartments at Chicago Zip code 60607 from two website zillow.com and apartments.com. Our goal is to predict rent of the apartment from various features like area, transit score provided.

Data cleaning

The data was extracted from two websites zillow.com and appartments.com.

Data frame 1

The shape of dataset 1 was:

```
df1.shape
```

```
(200, 8)
```

The null values in each column were:

```
df1.isna().sum()
```

```
Address              0
Bedrooms            10
Bathrooms            8
Rent                 0
Area(sq.ft)         23
Price per sq.ft     23
Type of house      200
Transit Score       92
dtype: int64
```

First, we take that part of dataframe in which rent value was not 0.

```
df1 = df1[df1['Rent'] != 0]
```

The data in Transit score column was in form of string and need to extract specific value. The column had following values:

```
79 / 100  (Excellent Transit)    15
74 / 100  (Excellent Transit)    12
75 / 100  (Excellent Transit)     8
100 / 100  (Rider's Paradise)     8
72 / 100  (Excellent Transit)     7
76 / 100  (Excellent Transit)     7
87 / 100  (Excellent Transit)     6
71 / 100  (Excellent Transit)     5
78 / 100  (Excellent Transit)     5
83 / 100  (Excellent Transit)     5
88 / 100  (Excellent Transit)     2
86 / 100  (Excellent Transit)     2
73 / 100  (Excellent Transit)     2
81 / 100  (Excellent Transit)     1
97 / 100  (Rider's Paradise)      1
77 / 100  (Excellent Transit)     1
84 / 100  (Excellent Transit)     1
90 / 100  (Rider's Paradise)      1
92 / 100  (Biker's Paradise)      1
Name: Transit Score, dtype: int64
```

We changed the type of column to string, replace space code (\xa0) with actual space and then split element and fetch transit score. The values after this cleaning were:

```
79.0     15
74.0     12
75.0      8
100.0     8
72.0      7
76.0      7
87.0      6
71.0      5
78.0      5
83.0      5
88.0      2
86.0      2
73.0      2
81.0      1
97.0      1
77.0      1
84.0      1
90.0      1
92.0      1
Name: Transit Score, dtype: int64
```

To define type of house we created a function which check number of bedrooms and rent value and assign it one of the four types. The limits were as following:

```
def get_house_type(num_bedrooms, price):
    if num_bedrooms >= 4 and price >= 1000000:
        return "Luxury"
    elif 2 <= num_bedrooms <= 3 and 500000 <= price < 1000000:
        return "Residence"
    elif num_bedrooms <= 2 and price < 500000:
        return "Single Family"
    else:
        return "Other"
```

```
df1["Type of house"] = df1.apply(lambda row: get_house_type(row["Bedrooms"], row["Rent"]), axis=1)
```

To deal with missing values in transit score we replace them median of the column.

```
median_score = df1['Transit Score'].median()
df1['Transit Score'].fillna(median_score, inplace=True)
```

We drop the rest of the rows with missing values, and we get data with no missing values of shape:

```
df1.dropna(inplace=True)
df1.shape
```

```
(137, 8)
```

Data frame 2

The shape of dataset 2 was:

```
data2.shape
```

```
(696, 8)
```

There was one row in which transit value was “-“. We first replaced it with null value n=and then replaced null value with median of column.

```
df2['Transit Score'] = df2['Transit Score'].replace('-', np.nan)
```

```
median_score = df2['Transit Score'].median()
df2['Transit Score'].fillna(median_score, inplace=True)
```

The null values in total area were:

```
df2['Total Area'].isna().sum()
```

```
138
```

We drop the rows which have 0 area.

```
df2.dropna(subset=['Total Area'], inplace=True)
```

```
df2.shape
```

```
(558, 8)
```

For type of house we repeat the same process of applying function.

```
def get_house_type(num_bedrooms, price):
    if num_bedrooms >= 4 and price >= 1000000:
        return "Luxury"
    elif 2 <= num_bedrooms <= 3 and 500000 <= price < 1000000:
        return "Residence"
    elif num_bedrooms <= 2 and price < 500000:
        return "Single Family"
    else:
        return "Other"
```

The column Bedroom had one string value.

```
df2['Bedrooms'].value_counts()
```

```
2         253
1         168
3          83
Studio     40
4          11
5           3
Name: Bedrooms, dtype: int64
```

The studio apartment contains only one bedroom. So, we replaced it with 1.

```
df2['Bedrooms'] = df2['Bedrooms'].replace("Studio", 1)
```

```
df2['Bedrooms'] = df2['Bedrooms'].astype(float)
```

We drop extra null columns. Then we rename column so that dataframe have same name of columns.

```
df1 = df1.rename(columns={'Area(sq.ft)': 'Total Area'})
```

```
df1 = df1.rename(columns={'Price per sq.ft': 'Price per sqft'})
```

```
df2 = df2.rename(columns={'Price per Sq.Ft.': 'Price per sqft'})
```

```
df1.columns
```

```
Index(['Address', 'Bedrooms', 'Bathrooms', 'Rent', 'Total Area',
       'Price per sqft', 'Type of house', 'Transit Score'],
      dtype='object')
```

```
df2.columns
```

```
Index(['Address', 'Bedrooms', 'Bathrooms', 'Rent', 'Total Area',
       'Price per sqft', 'Transit Score', 'Type of house'],
      dtype='object')
```

Then we merged both dataframe into single dataframe and saved it in csv file and use that dataframe for further application.

Data Analysis

The data has total 8 columns:

```
merged_df.columns
```

```
Index(['Address', 'Bedrooms', 'Bathrooms', 'Rent', 'Total Area',
       'Price per sqft', 'Type of house', 'Transit Score'],
      dtype='object')
```

The shape of data is:

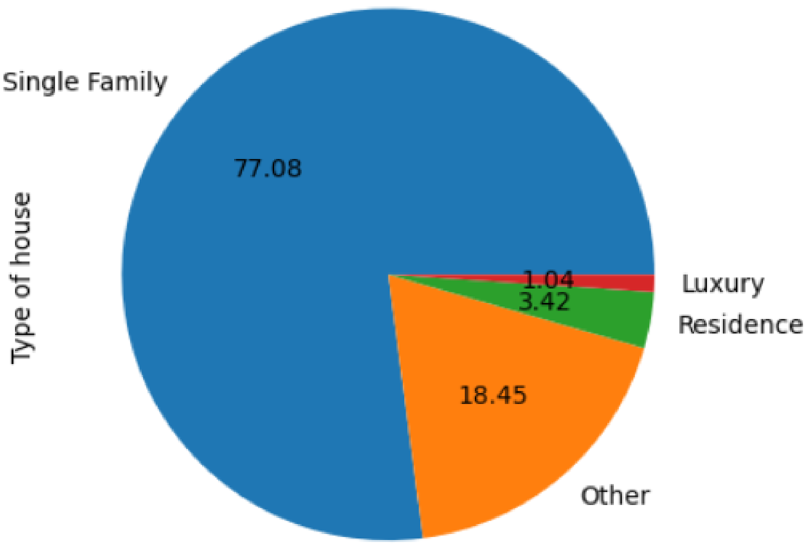
```
merged_df.shape
```

```
(695, 8)
```

The most of apartments were single.

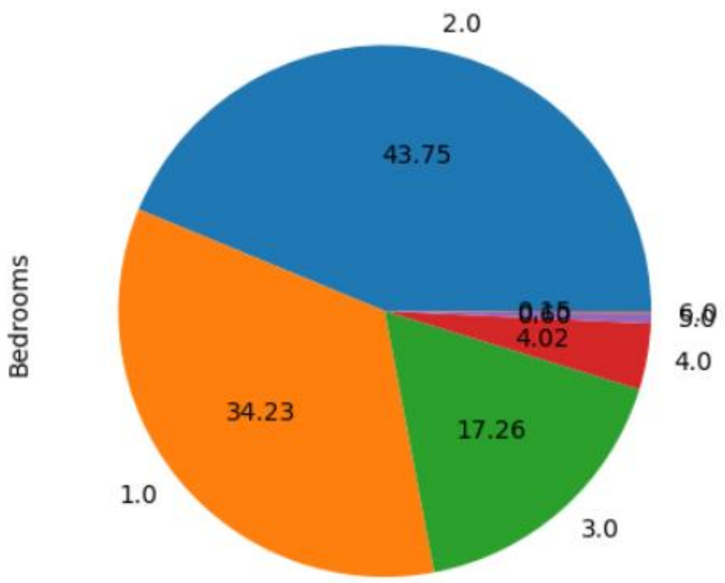
```
data['Type of house'].value_counts()
```

```
Single Family    518
Other            124
Residence         23
Luxury             7
Name: Type of house, dtype: int64
```

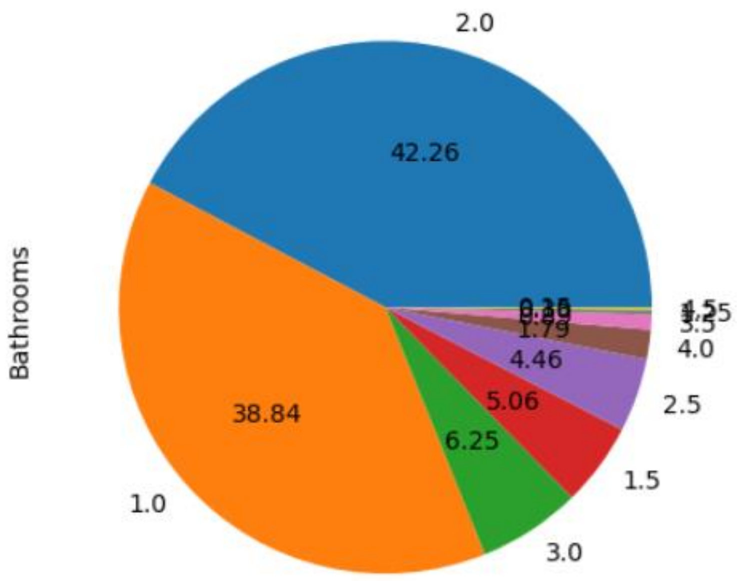


The maximum number of bedrooms was 6 and minimum number of bedrooms was 1.

Most of the apartments have 3 bedrooms.

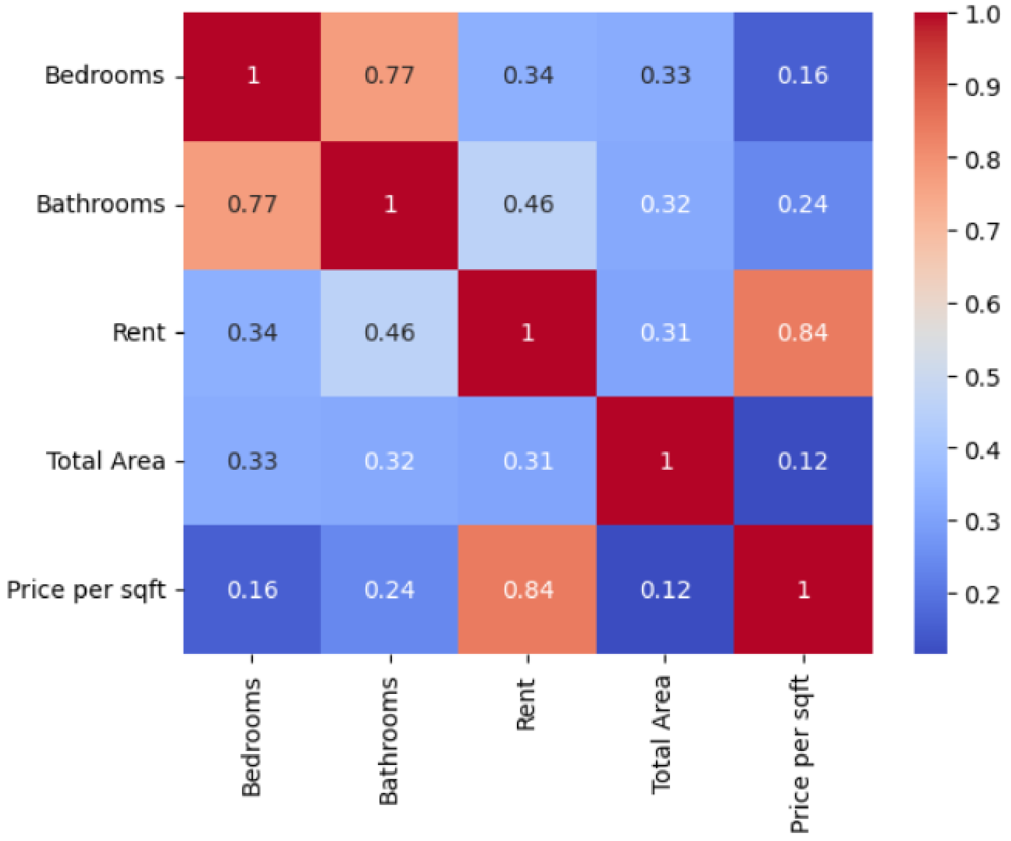


Most of them have 2 bathrooms:

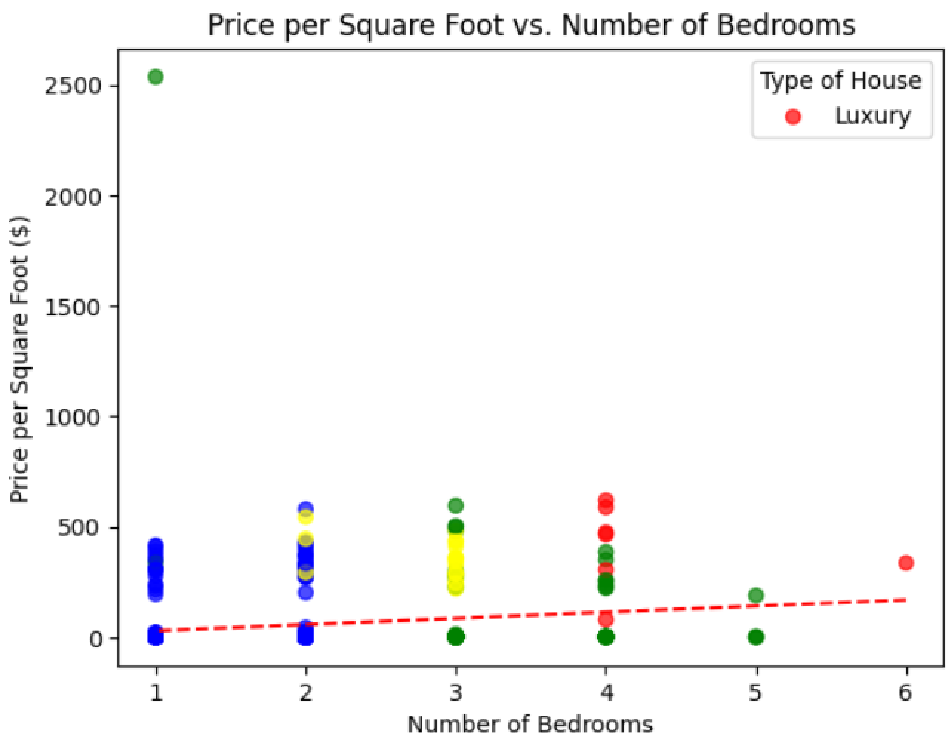


Visualization

To see whether variables are related to rent or not, we draw an heatmap of data frame.



To see relationship between price per sq feet and number of bedrooms, we draw linear regression line between them.



ML Analysis

Standardization

The y feature (Rent) was separated as y and rest of variables as X. The address variable was dropped as it has no relation.

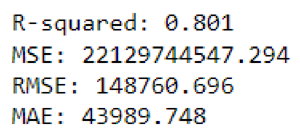
The data was split into train and test set with ratio 80, 20 respectively. To normalize data standard scalar on Scikit learn is used. The train set is fitted and transformed, and test set is transformed.

Training and Testing

The linear regression instance was initialized, and train data was fit and evaluated on the test dataset.

Results:

Following were results after evaluation:



```
R-squared: 0.801
MSE: 22129744547.294
RMSE: 148760.696
MAE: 43989.748
```

Reflection

Hardest Part

The hardest part was to extract data from the websites and then clean it. We wrote scrappers to extract data from it and then clean it with exploration and cleaning techniques.

Concrete Results

We have well trained model with a god R value of 0.81 or 81%.

Next Step

We will improve our model reducing its error and increasing R value.