San Francisco Housing

Charlene Khun, Helena Thiessen, Benny Chen, and Rongjie Mai

Inspiration

- Living in the Bay Area is expensive
- Understand what factors drive the renting/housing cost in Bay Area
- **Main Problem:** How do various factors such as the location and square footage affect and influence the renting prices in San Francisco?
- Identify ways for renters to get the best value and investors to optimize purchases

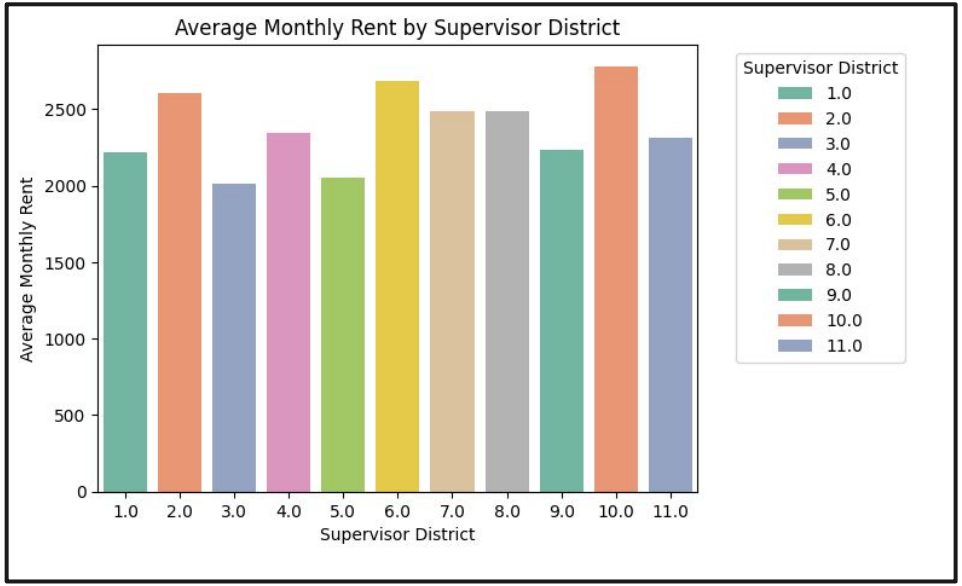
Dataset

- Utilized SF Apartment Rental Inventory dataset from April 8th, 2025
- Information obtained from the submissions of apartment owners
- Over 186K apartment records
- Key variables include (monthly rent, number of bedrooms, number of bathrooms, etc.)

Problem Background (Q1)

How does location influence rental price?

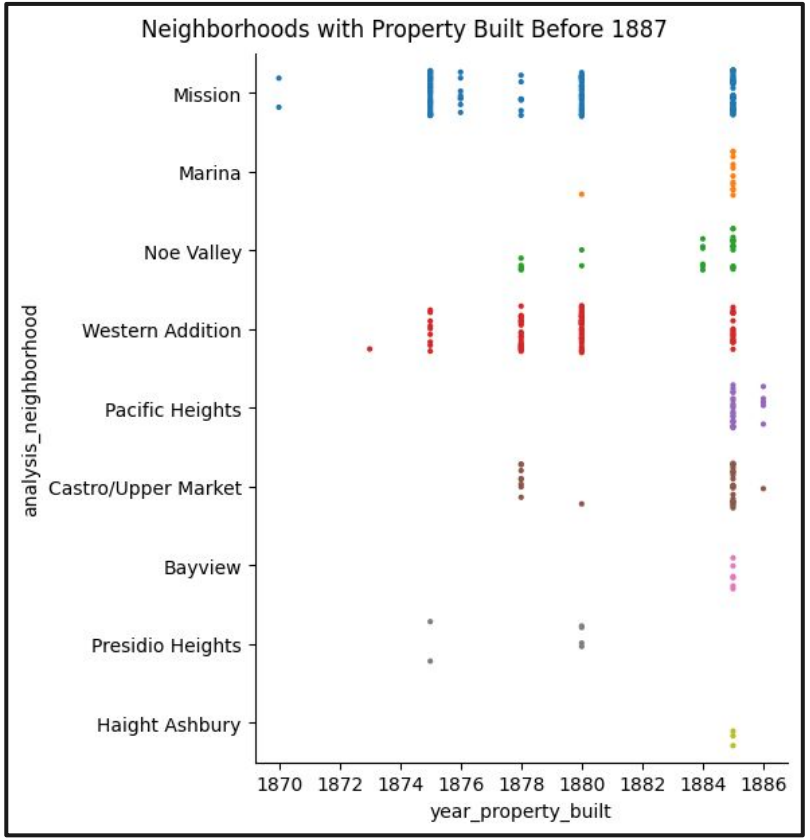
- Used a barplot
- District 3 and 5 are the most affordable
- Districts 2, 6, and 10 are the most expensive



Problem Background (Q2)

Which neighborhood has the oldest buildings (built before 1886)?

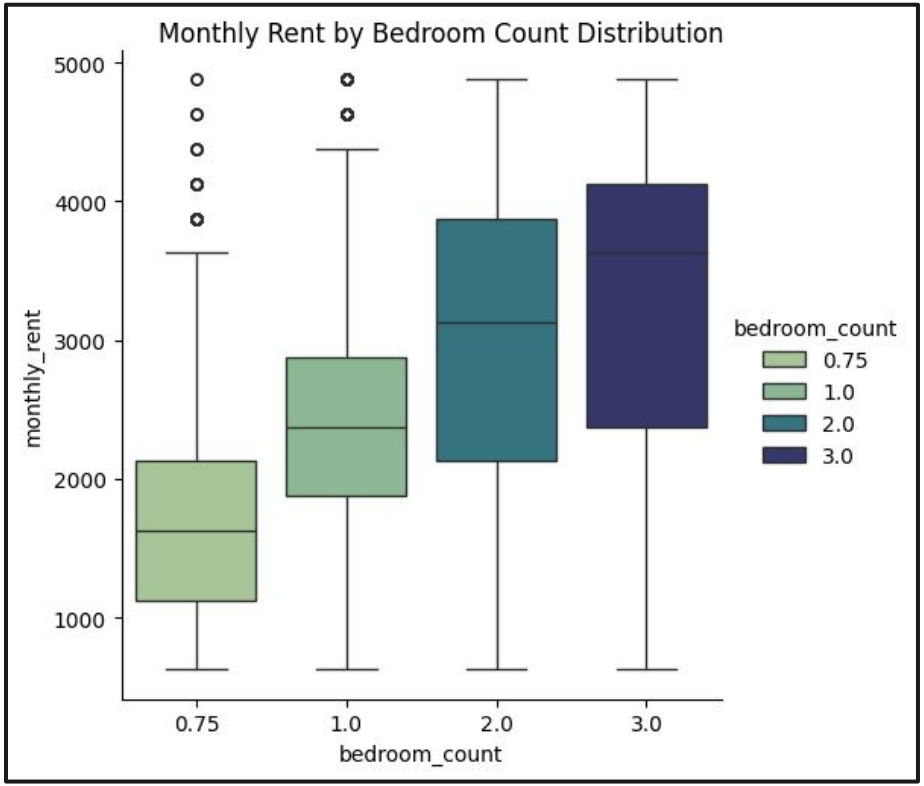
- Used a strip plot
- Mission and Western Addition have the most oldest buildings
- Other neighborhoods have considerably fewer old buildings



Problem Background (Q3)

How does bedroom count influence rent?

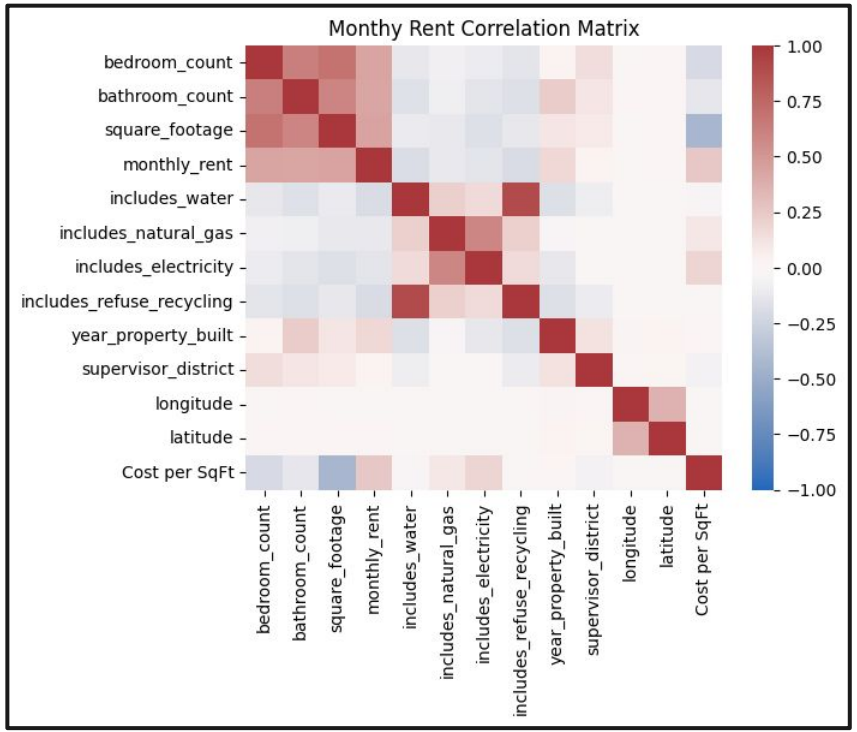
- Used a box plot
- Rent increases as bedroom count increases
- Large increase from one to two bedrooms, but a smaller increase from two to three bedrooms



Problem Background (Q4)

Which factor has the biggest impact on rent?

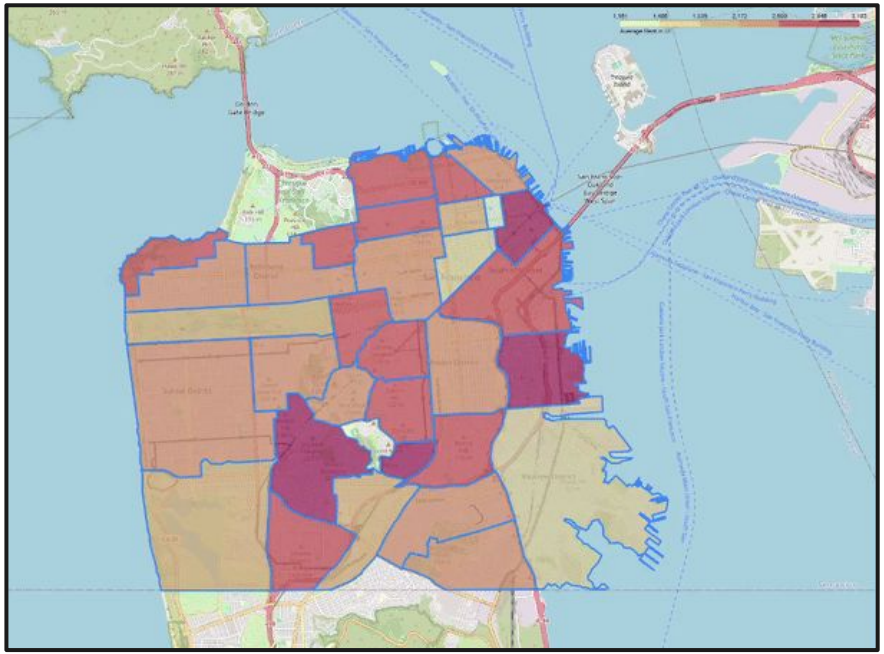
- Used a correlation matrix
- Square footage has the largest positive effect on monthly rent.
- Includes natural gas and includes refuse recycling has the largest negative impact on monthly rent.



Problem Background (Q5)

How do rental attributes differ among neighborhoods?

- Used a choropleth / folium map
- The Financial District has the highest average rent for only a small footage
- Bayview has a balance of high square footage and low average rent (best value)
- Chinatown has the lowest average rent but has very low square footage



Data Cleaning and Preprocessing (Pt 1)

- Imported necessary libraries including Matplotlib, Pandas, Seaborn as well as machine learning related libraries from sklearn
- Drop missing values for important features
- Drop unnecessary columns

```
##imports
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import re
import folium
import geopy
import numpy as np
%matplotlib inline
```

```
##ml imports
from sklearn.model_selection import StratifiedShuffleSplit
from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import cross_val_score
```

Data Cleaning and Preprocessing (Pt 2)

- Standardize inconsistent types and values
  - Map functions to correct non-numeric data
  - Define boolean values
- New columns were added to engineer new features
  - Created cost per square foot, longitude, and latitude

```
##map bedrooms to ints
def mapBed(x):
  if x == "One-Bedroom" or x == "One-bedroom" or x == "1 b
    return 1
  if x == "Two-Bedroom" or x == "Two-bedroom" or x == "2 b
    return 2
  if x == "Three-Bedroom" or x == "3 bedroom" or x == "3":
    return 3
  if x == "Four-Bedroom":
    return 4
  if x == "Studio" or x == "studio" or x == "0" or x == "0
    return 0.75
  if x == "5+" or x == "Five-Bedroom":
    return 5
  if x == "Vacant":
    return None

rent["bedroom_count"] = rent["bedroom_count"].map(mapBed)
```

Map function

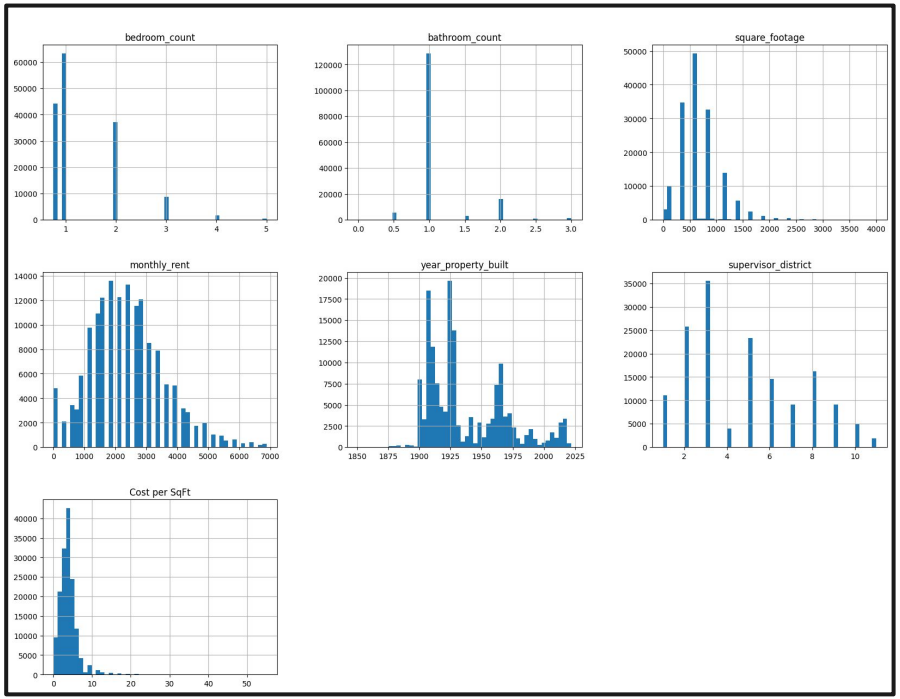
Data Exploration

To identify important features and relationships for our ML Model, we used:

- Histogram
- Choropleth Map
- Correlation Matrix
- Pairplot

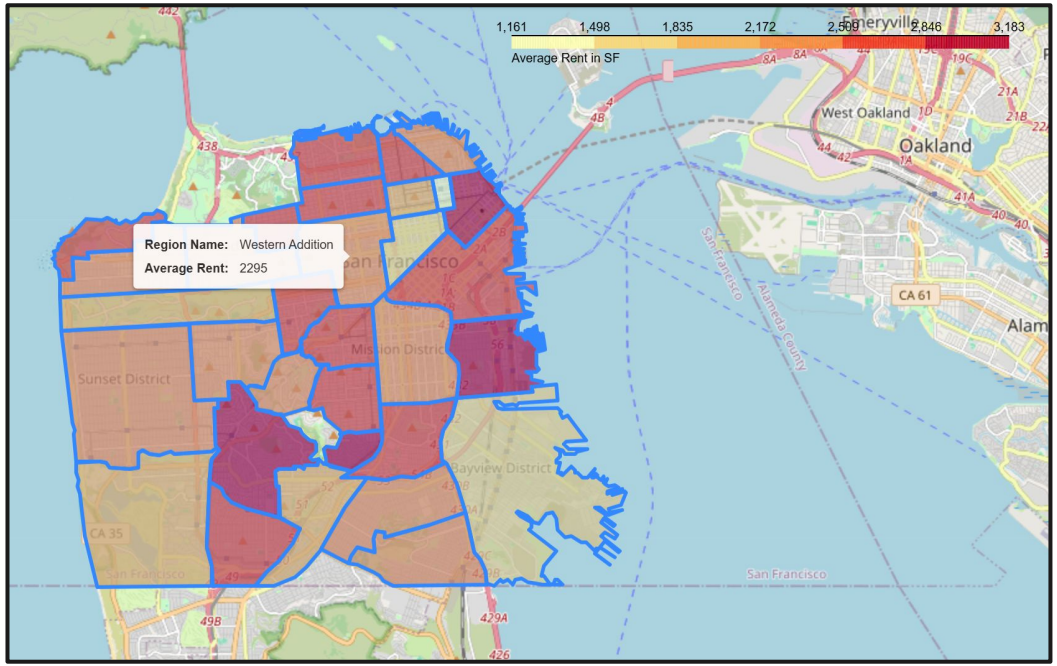
Data Exploration - Histogram

- Gives a sense of the distribution across each numeric category.
    - Allows us to better understand the spread and shape of the data.
- Identify outliers so that we can drop to improve ML model.
    - Ensures that extreme values don't disproportionately influence the analysis.

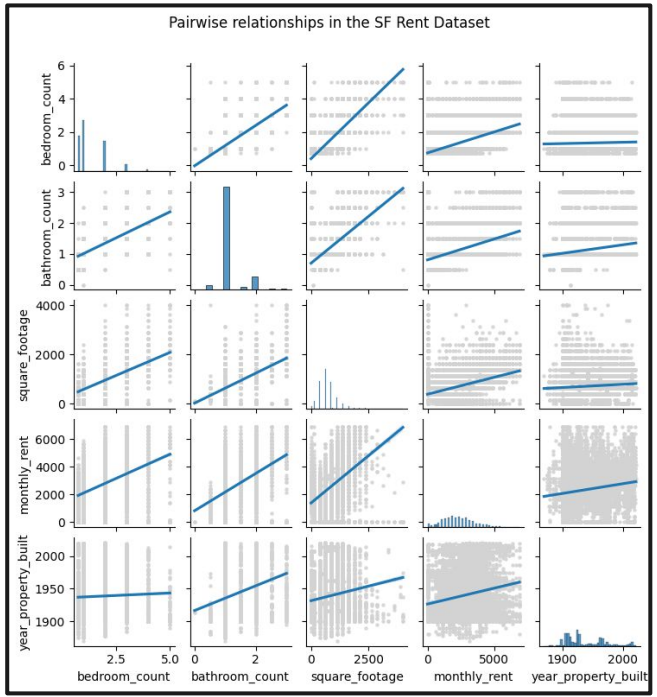


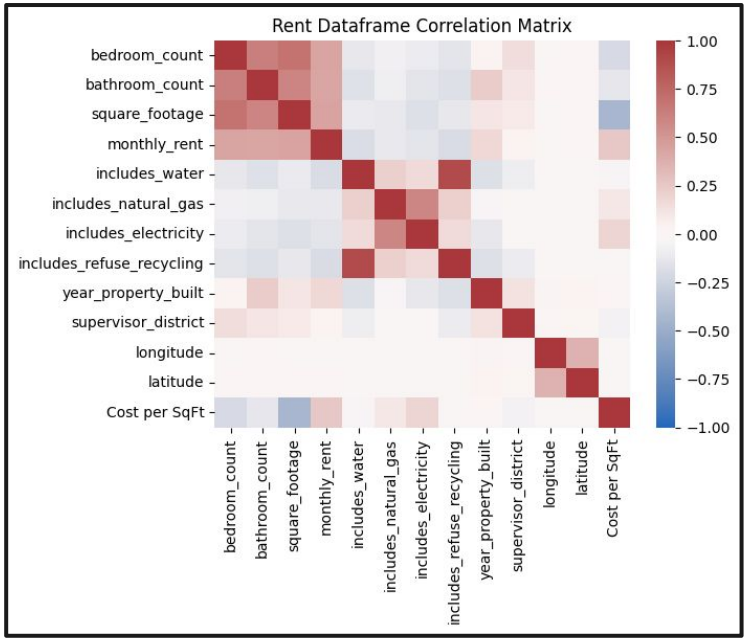
Data Exploration - Choropleth / Folium Map

- Neighborhood (a categorical variable) appears to influence rent prices.
- Legend shows the mean rent in different neighborhoods. Yellow indicates low rent, red indicates high rent.



Data Exploration - Pairplot and Correlation Matrix





Machine Learning Strategies (Pt 1)

- **Target Variable:** Monthly Rent
- Apply ML techniques to estimate rent based on property features
- Results benefit future and current renters and owners

Machine Learning Strategies (Pt 2)

- Split 90-10 into a training set and a test set stratified by year property built.
- Preprocessed the dataset by applying one-hot encoding to the categorical features.
- Used median imputation and ordinal standardization for numerical features.
- Both numerical and categorical features were integrated using a Column Transformer.

```
# convert categorical values into one-hot vectors
cat_encoder = OneHotEncoder()
rent_cat_1hot = cat_encoder.fit_transform(rent_cat)
rent_cat_1hot

cat_attr = ["analysis_neighborhood"]
rent_num = rent.drop('analysis_neighborhood', axis=1)
num_attr = list(rent_num)

num_pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy="median")),
    ('std_scaler', StandardScaler())
    ])

rent_num_tr = num_pipeline.fit_transform(rent_num)
```

```
full_pipeline = ColumnTransformer([
        ("num", num_pipeline, num_attr),
        ("cat", OneHotEncoder(), cat_attr)
    ])

rent_prepared = full_pipeline.fit_transform(rent)
cols=list(rent_num) + cat_encoder.categories_[0].tolist()

## Convert the 2D array into Pandas DataFrame
rent_prep_df = pd.DataFrame.sparse.from_spmatrix(
    rent_prepared,
    columns=cols,
    index=rent.index)
rent_prep_df.head()
```

ML Model Fine Tuning (Pt 1)

- Performing 10 cross validation for each chosen machine learning model and using RMSE, MAE, & R^2 as the evaluation metrics, we found the following:

	Linear Regression	Decision Tree Regression	Random Forest Regression
RMSE	748.00	658.89	639.68
MAE	581.35	466.02	464.93
R^2	0.43	0.56	0.58

- Random Forest Regression achieved the best performance, with the lowest RMSE (score = 639.68), lowest MAE (score = 464.93), and highest R^2 value (0.58) among the models tested.

ML Model Fine Tuning (Pt 2)

- Random forest was then fine-tuned with Grid Search and Random Search. We obtained the following results:

	Grid Search	Random Search
RMSE	541.87	541.39
MAE	394.66	394.27
R^2	0.7011	0.7016

- Even though Grid search performed slightly worse in this iteration, we chose Grid Search based on its more consistent performance in prior iterations and similarly strong training scores.

ML Model Results (Pt 1)

- To collect our model results, the fine-tuned Random Forest Regression model was run on our test data set after applying the preprocessing pipeline.
- Results were assessed through RMSE, MAE values, R^2, confusion matrix, and line graph.

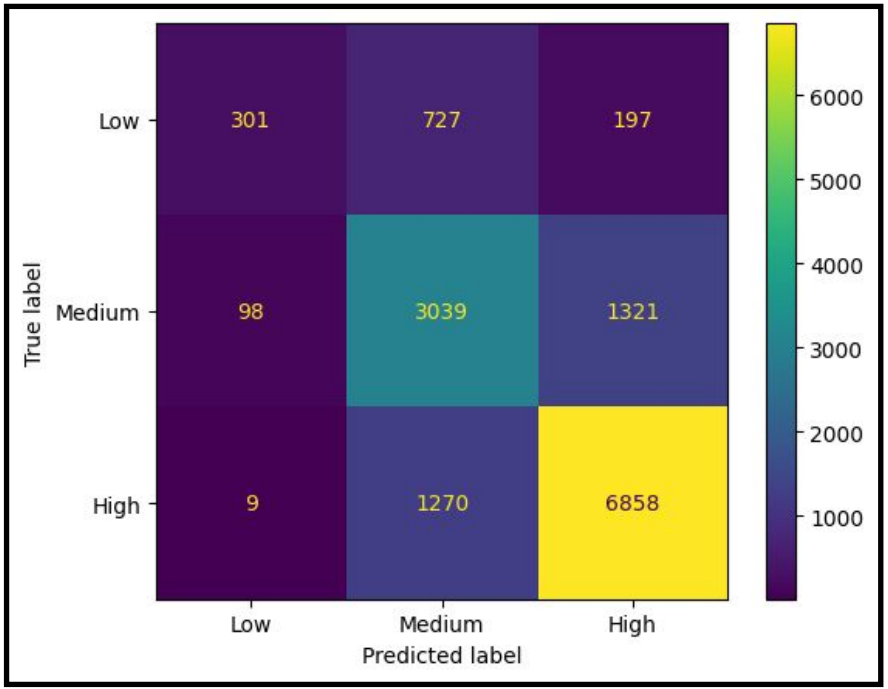
ML Model Results (Pt 2)

	Random Forest Regression
RMSE	621.53
MAE	455.27
R^2	0.6115

- Large margin of error apparent in the RMSE, MAE, and R^2 value.
- Although there were many attempts made at fine-tuning the model to achieve a better result, these scores were ultimately determined to be a consequence of the data supplied.

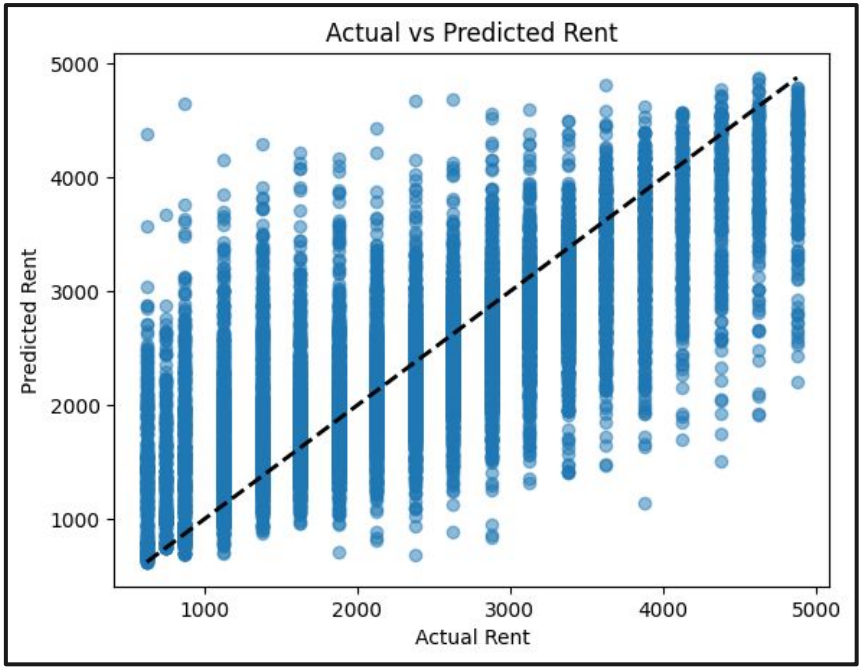
ML Model Results (Pt 3)

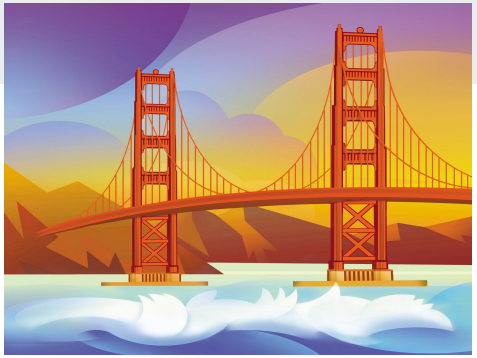
- The model correctly classified 84% of high-cost, 68% of medium-cost, and 24% of low-cost apartments.
- While accurate for high-cost units, it tends to overestimate rent for lower-cost apartments.



ML Model Results (Pt 4)

- The black dashed line represents the ideal predictions
- Upward trend represents positive correlation
- Wide spread data indicates high variability, suggesting a lack of precision





Conclusion

- Our project demonstrates how machine learning can assist in estimating rent but also underscores the limits of purely data-driven approaches in real estate.
- The final evaluation of the model on the test set yielded an RMSE of 621.53, an MAE of 455.27, and an R^2 value of 0.6115, reflecting moderate accuracy and acceptable generalization.
- While the model can reasonably predict rental prices, it is not highly precise, emphasizing the inherent complexity of rental pricing.
- Model improvements may be available through collecting more data (walkability, transit, etc), and exploring advanced models like gradient boosting.

Thank you for listening!