Isaac Pacheco
Isaia Pacheco
December 10th, 2025

Apartment Rental Rental Predictions and Mispricing

Objectives

1) Understand how property features and location influence market value.
2) Build a model to predict the rental prices of apartments in the United States.
3) Build another model to classify listings as fair or mispriced.
4) Assist renters, property managers, and owners in making data-driven decisions

Data Sources

Our dataset came from the UC Irvine Machine Learning Repository and it was:

- Not well documented - features were ambiguous (i.e., time)
- Not cleaned - features contain multiple types and erroneous values
- Mostly complete

Our dataset contained:

- 99,826 total records of apartment, condo, housing, and mansion listings
- 23 features but the features of most interest to us were:
    - Structural Features
        - Bedrooms
        - Bathrooms
        - Square feet
    - Spatial Features
        - Latitude
        - Longitude
        - City Name
        - State
    - Price: in dollars
    - Price_Type: Month or Weekly
    - Time: We originally planned to use this feature to merge other datasets on housing indexes but since there was a lack of documentation, we failed to identify what time meant and decided not to make an assumption (e.g., list post datetime, database insertion datetime

    - Source: Listing source
    - Category: whether the listing is an apartment, condo, home, etc.

The dataset was valuable in many ways because it contained:

- Large sample of apartment rental listings
- Representation from all 50 states
- Low-income to luxury rentals
- Various listing sources

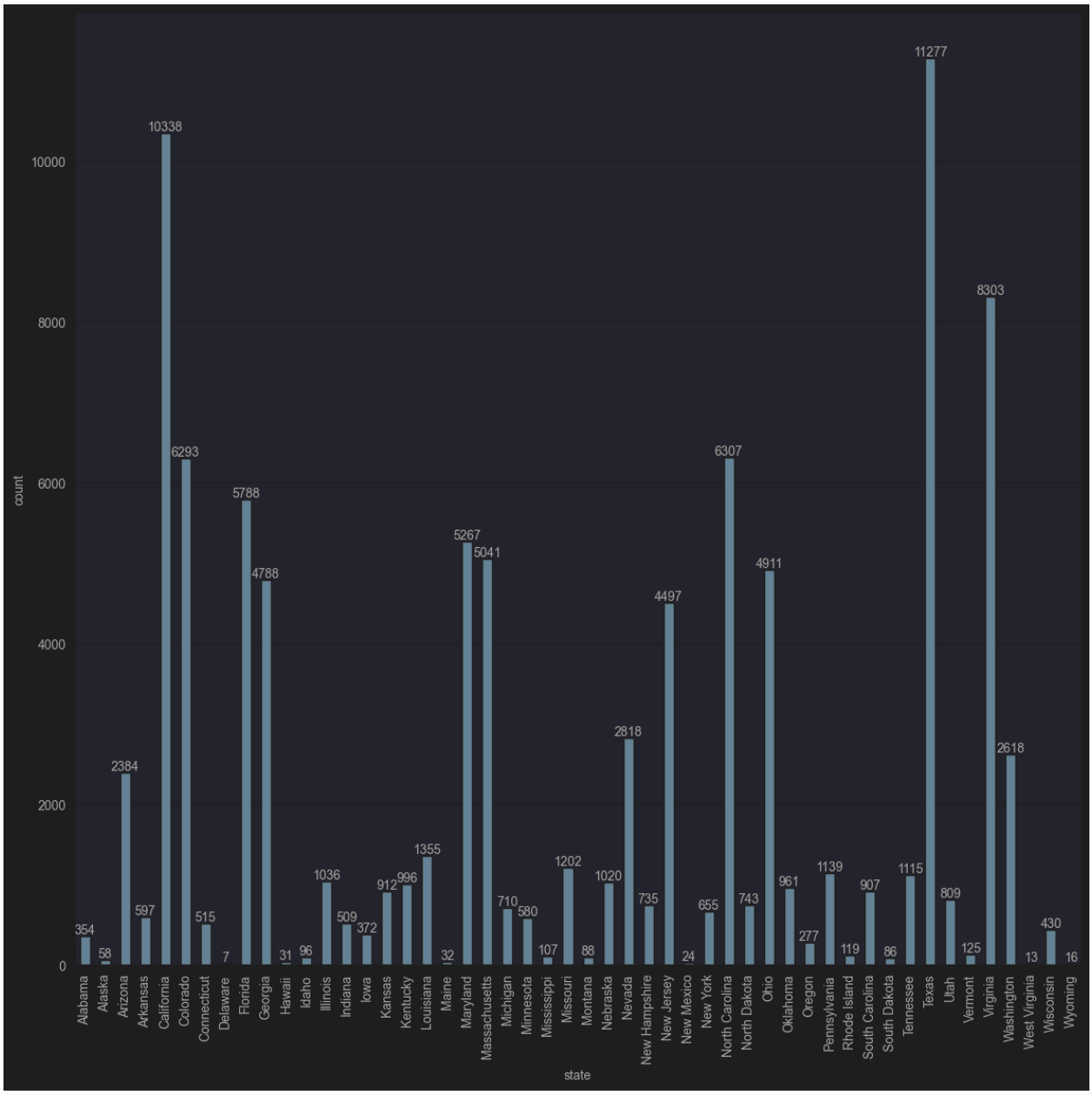
However the dataset contained biasness as well due to the following:

- Class imbalance amongst each state and city
- Possible synthetic listings - we were unable to verify the validity of all sources because some sources could not be found or we discontinued.
- A small number of listings categorized as apartments did not represent common apartments in context of price, count of bedrooms and bathrooms, square footage and skewed the data.

	category	count
0	gym	1
1	housing,rent	7
2	housing,rent,apartment	99762
3	housing,rent,commercial,retail	42
4	housing,rent,condo	3
5	housing,rent,home	4
6	housing,rent,other	1
7	housing,rent,short_term	4
8	parking,patio,deck	1

	price_type	count
0	monthly	99817
1	weekly	3

	source	count
0	rentdigs.com	91239
1	rentlingo	6924
2	listedbuy	571
3	gosection8	437
4	realrentals	269



Upon exploring the descriptive statistics of the raw data we found outliers that skewed the data and were not representative of apartments for example:

- Apartments with 0 bedrooms (that is a studio)
- Apartments with 9 bedrooms or 9 bathrooms (not an apartment, probably a mansion)
- Priced at $100/month (extremely low for an apartment)
- Priced at $52,5000/month (extremely high for an apartment)
- Square footage of 40,000 (extremely high for an apartment)

Possible reasons for outliers:

- Input errors
- Massive properties (not really representative a common apartment)

	id	uci_id	bathrooms	bedrooms	price	price_display	square_feet	latitude	longitude	time
count	99361.00	99361.00	99311.00	99239.00	99360.00	99360.00	99361.00	99361.00	99361.00	99361.00
mean	49946.62	5358206852.03	1.42	1.73	1524.94	1525.02	956.13	36.94	-91.57	1559659121.08
std	28806.28	184458113.33	0.53	0.75	902.76	902.80	387.16	4.61	15.84	11036904.75
min	0.00	5121046168.00	1.00	0.00	100.00	100.00	106.00	19.57	-159.37	1544174418.00
25%	25007.00	5197948613.00	1.00	1.00	1012.00	1012.00	730.00	33.74	-104.82	1550831636.00
50%	49852.00	5508672674.00	1.00	2.00	1350.00	1350.00	900.00	37.21	-84.55	1568745002.00
75%	74859.00	5509007380.00	2.00	2.00	1795.00	1795.00	1115.00	39.96	-77.61	1568767011.00
max	99825.00	5669438542.00	9.00	9.00	52500.00	52500.00	40000.00	64.83	-68.78	1577391425.00

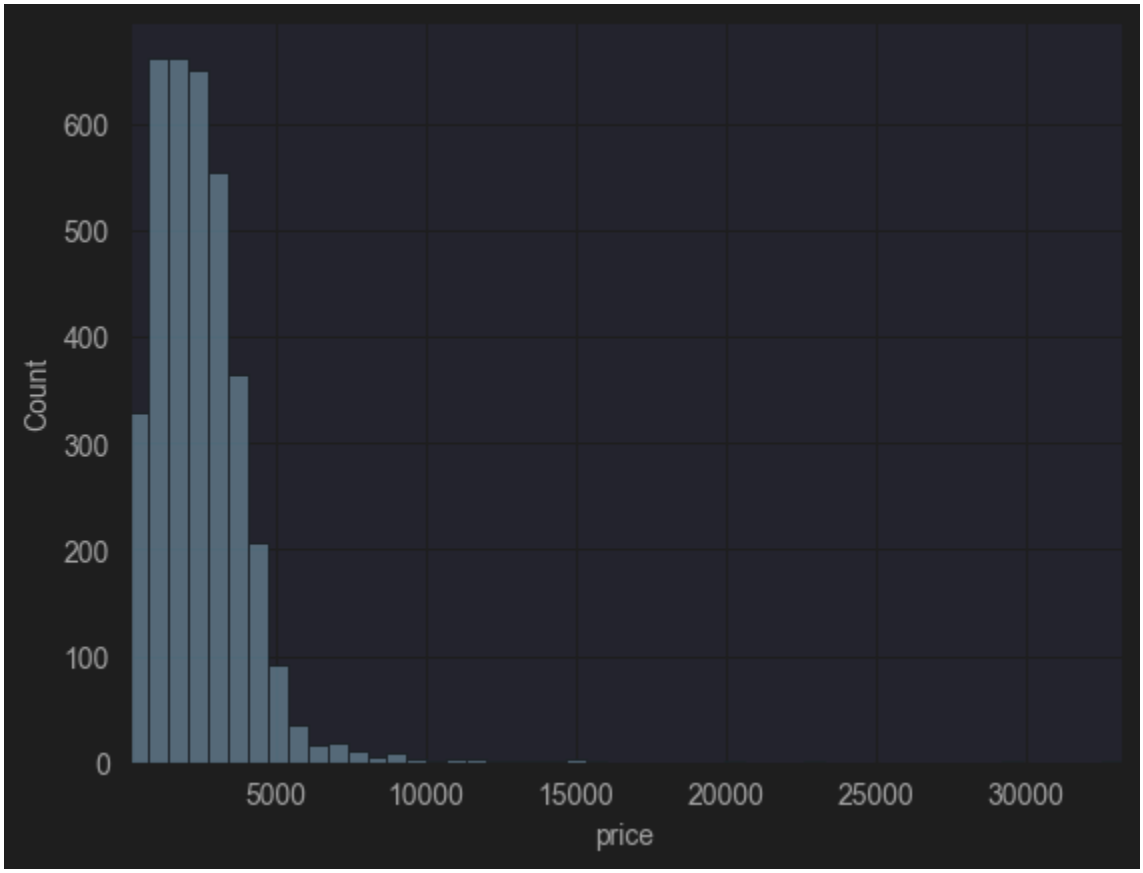
Handling Skewness of Bedrooms and Bathrooms

- We defined that the range of bedrooms and bathrooms that a typical apartment contained was between any combination of 1-3.
- We decided this was reasonable because when you start exceeding these numbers we begin to gain skepticism whether the listing is actually a townhouse or home, or mansion - properties that aren't an apartment.
- Any listing that did not fall within this range was filtered out.

Handling Skewness of Price

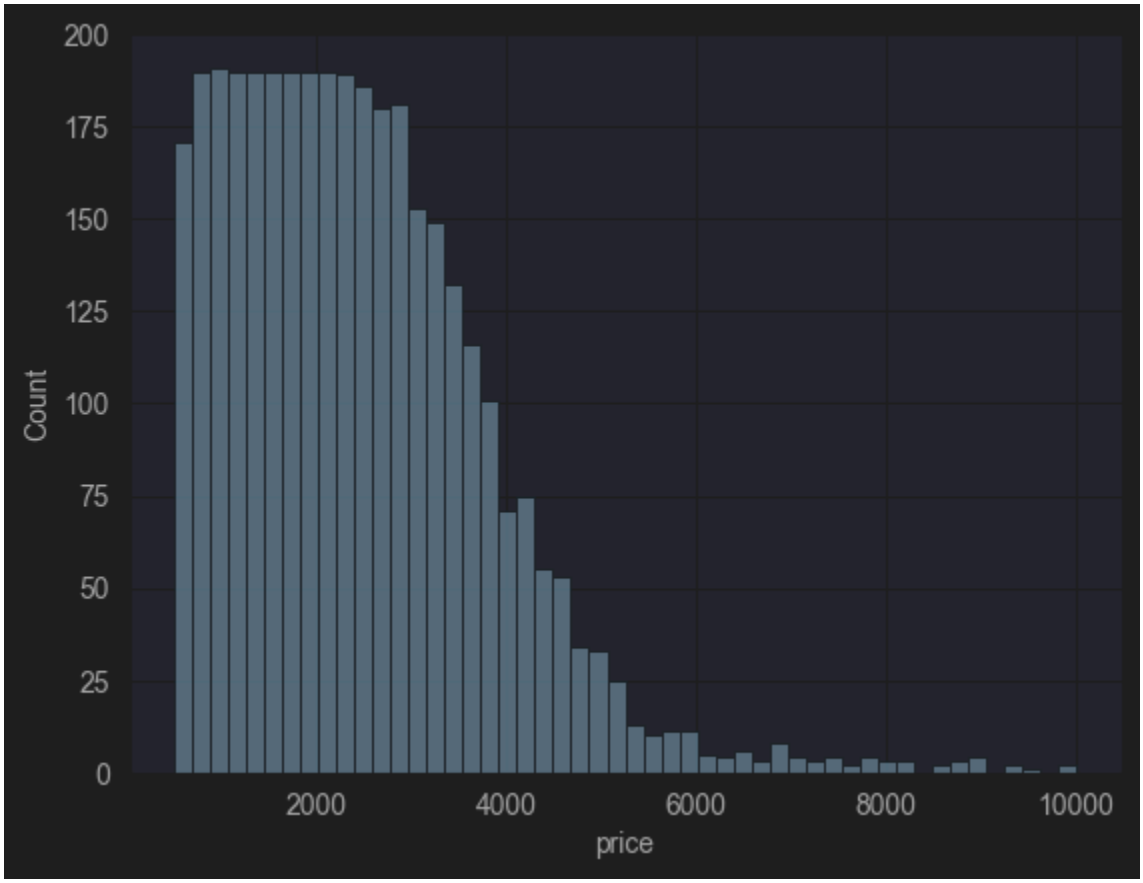
- We defined the range of monthly price of an apartment between $500 - $10,000
- We decided this was reasonable because anything less than $500 is extremely low for a low-income apartment and anything above $10,000 was extremely high for a luxury/skyrise apartment.
- Any listing that did not fall within this price range was filtered out.

Distribution of Price Before Filtering



- Heavily skewed right, long thin tail
- Unimodal
- Centered between $1,500 - $2,500
- Outliers priced between $10,000 - $30,000
- Range was between ~$500-$30,000

Distribution of Price After Filtering

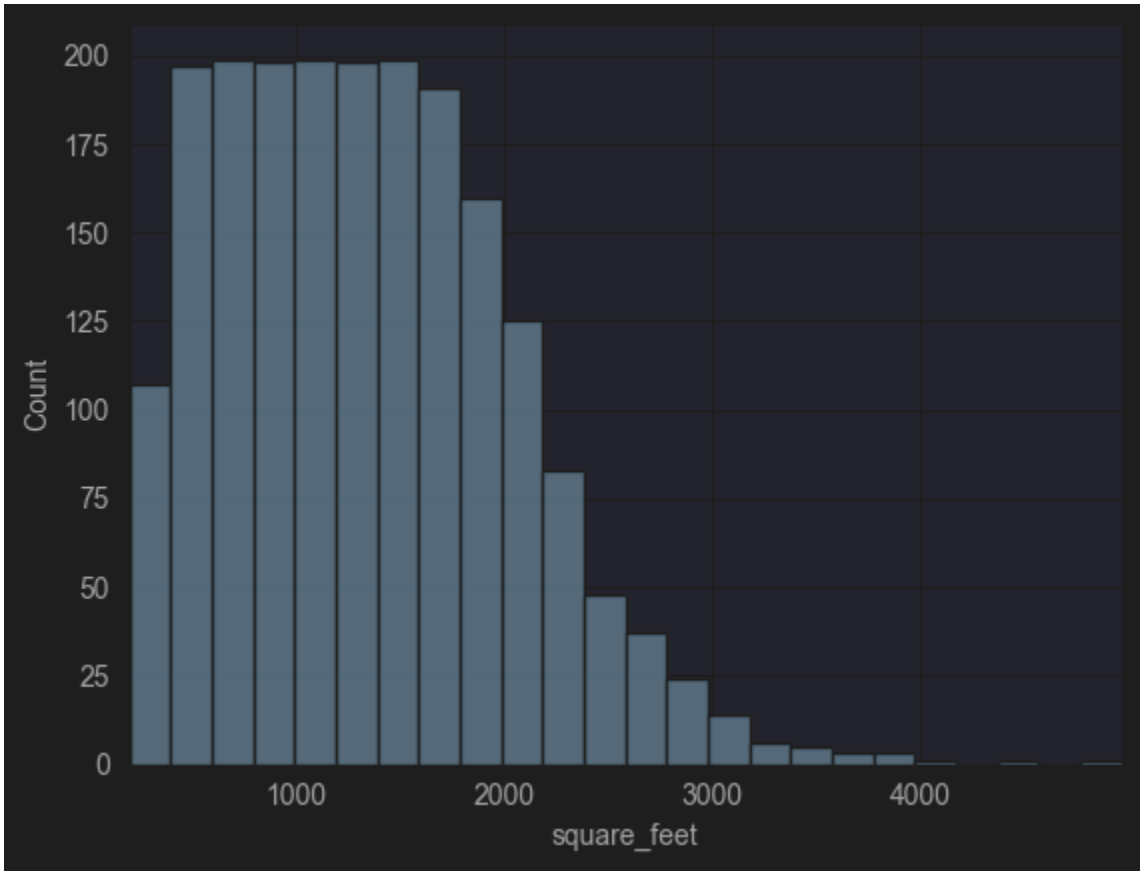


- Unimodal
- Strongly skewed right, long thin tail
- Centered between $1,000 - $3,000
- Outliers are priced between $6,000 - $10,000
- Range was reduced to ~$500-$10,000

Handling Skewness of Square Footage

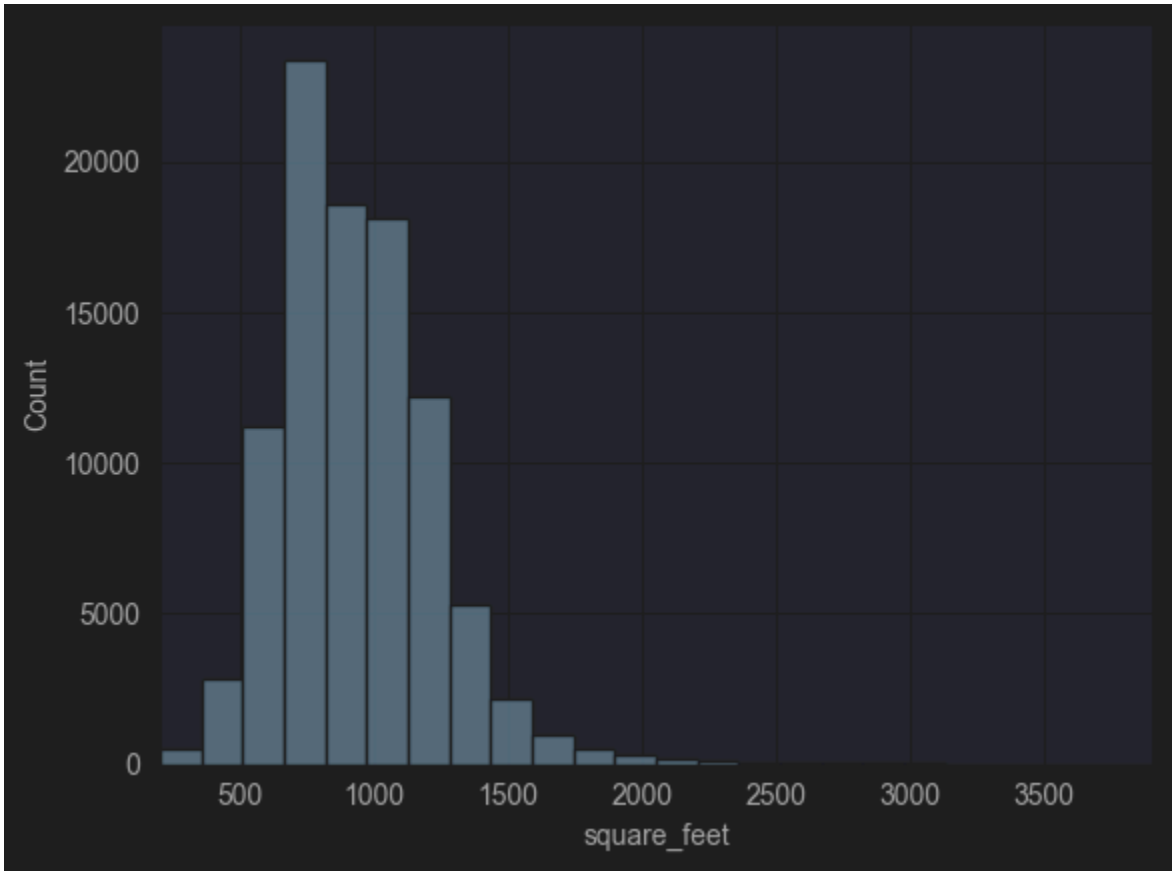
- We defined the range of square footage of an apartment up to 4,000 square feet.
- We decided this was reasonable because anything above 4,000 square feet was extremely high for a luxury/skyrise apartment or in a sparse region.
- We did not bound the lower end because in denser population regions like New York, it is common to see very small square footage apartments.
- Any listing that exceeded 4,000 square feet was filtered out.

Distribution of Square Feet Before Filtering



- Strongly skewed right
- Unimodal
- Centered between ~250 square feet and ~1,750 square feet
- Outliers were between ~4,000 and ~5,000 square feet
- Range was between ~250 square feet - 5,000 square feet

Distribution of Square Feet After Filtering



- Strongly skewed right
- Unimodal
- Mean is about ~800 square feet
- Centered between ~750 and ~1,250 square feet
- Range ~250-4,000 square feet

Skewness Analysis and Log Transformation of Square Feet and Price

- Even after defining criteria of what is representative of a common apartment and manually filtering, there still existed skewness from a small number of listings.
- We still wanted low-income, small apartments and expensive, luxury apartments to be represented in our dataset so we didn't filter out all the data.
- We needed a further means to reduce the influence of these listings in our model to predict rental price and find a linear relationship.
- We applied log transformation of these features because:
    - It revealed a linear relationship between square feet and price which is necessary for linear regression

- Stabilized variance and,
- Reduced effects on outliers on our model's performance and accuracy.

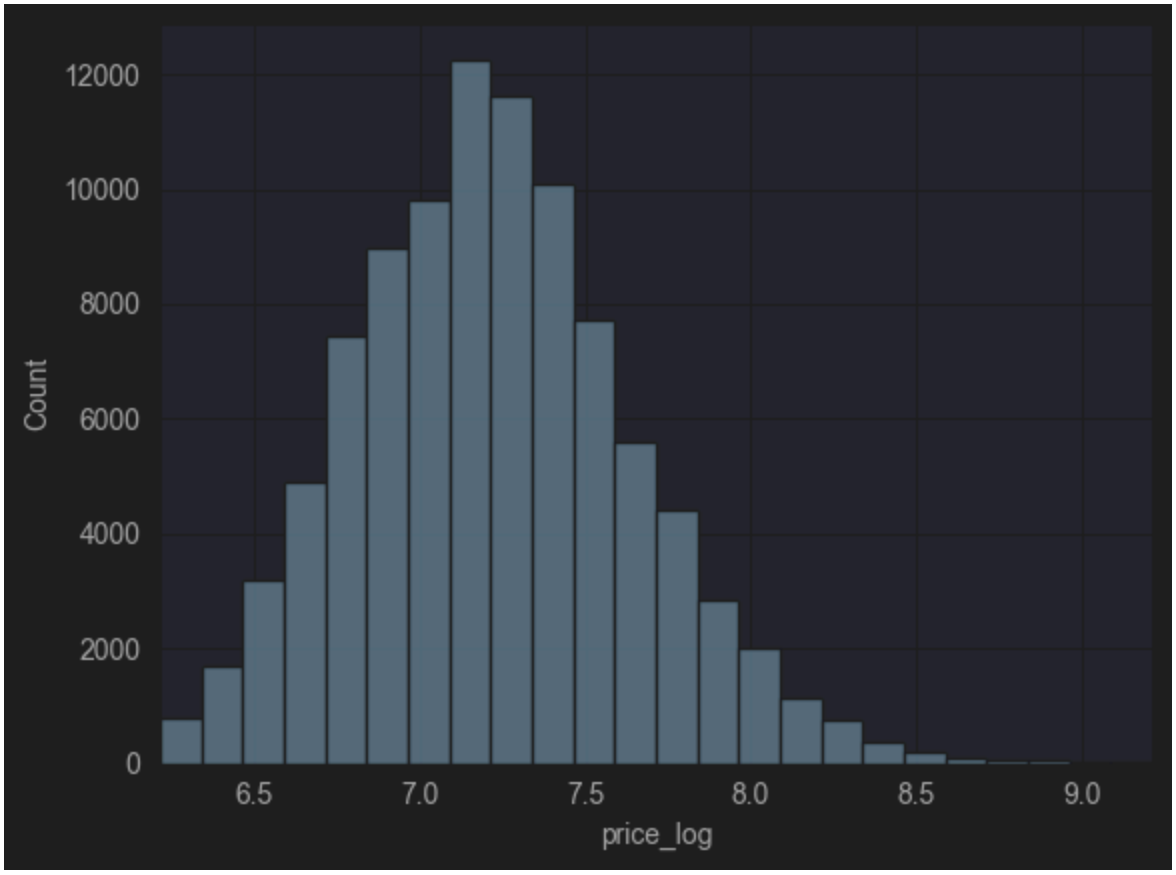
Skew Analysis After Filtering and Before Log Transformation

Feature	Skew	Interpretation	Action
price	2.28	Strongly right-skewed	Should transform
square_feet	1.17	Strongly right-skewed	Should transform

Skew Analysis After Filtering and Log Transformation

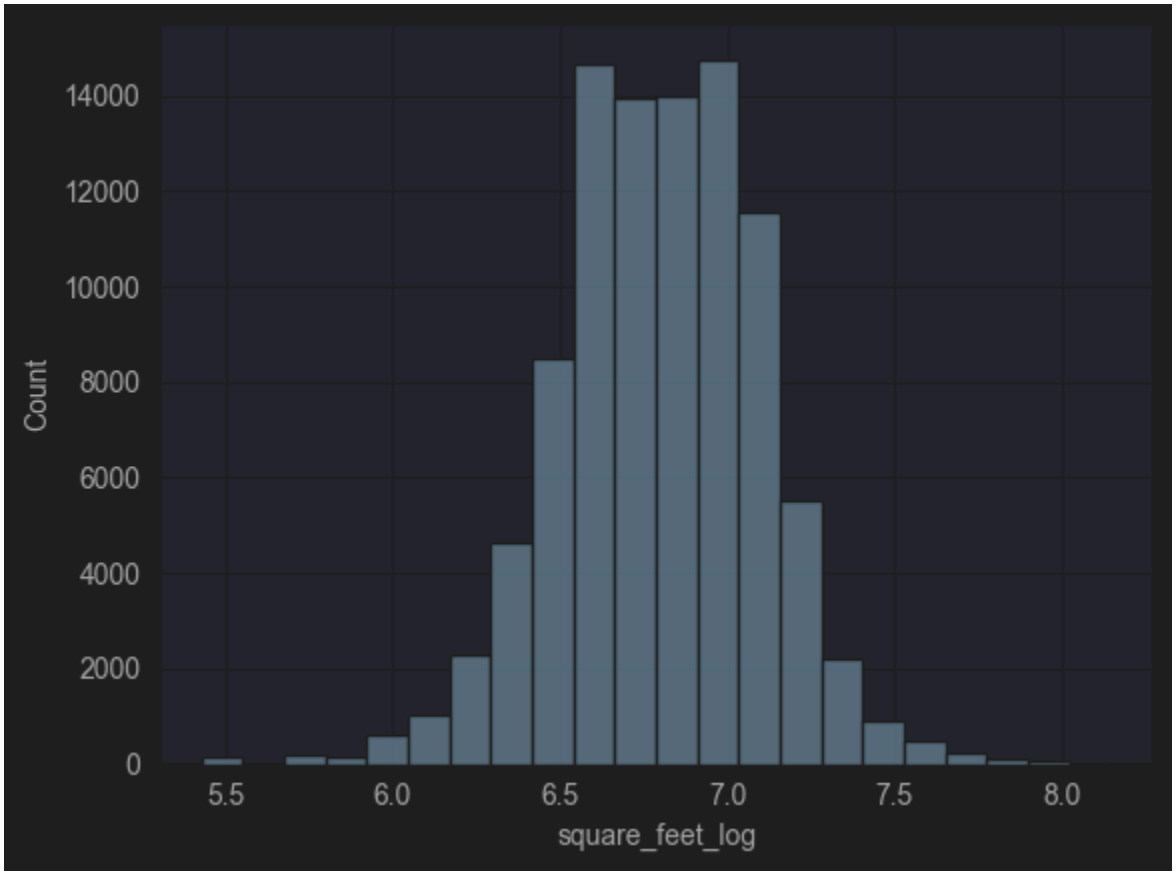
Feature	Skew	Interpretation
price	2.28 -> 0.36	Weakly right-skewed
square_feet	1.17 -> -0.21	Weakly left-skewed

Distribution of Log(Price)



- Weakly skewed right
- Unimodal, somewhat normal
- Mean is about 7.2
- Centered at about 7.2
- Range was between 0 to 9

Distribution Plot of Log(Square Feet)



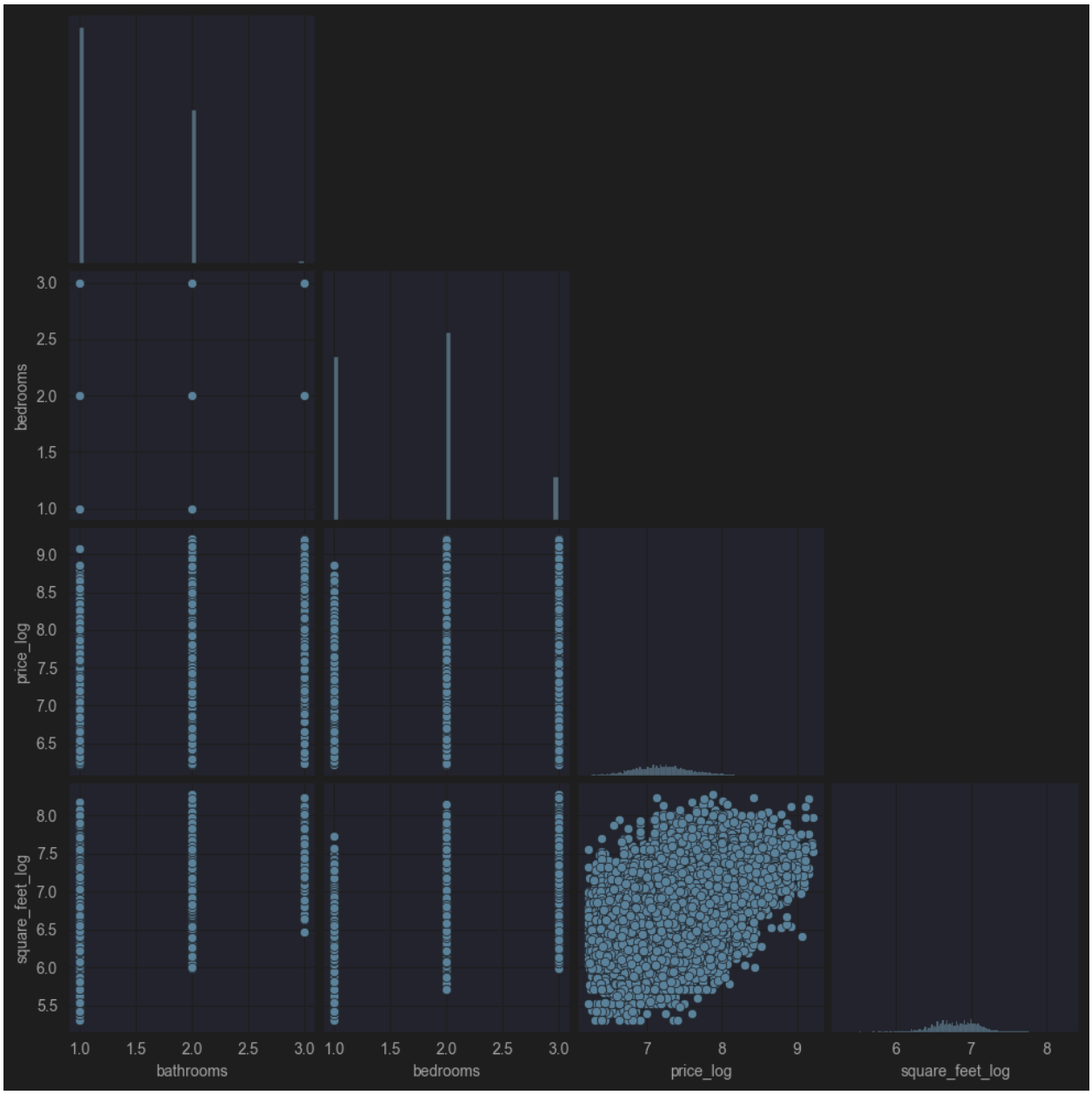
- Weakly skewed left
- Unimodal, somewhat normal
- Mean is about 7.2
- Centered between 6.5 and 7.0
- Range was between 5.5 to 8.0

Pair Plot of Structural Features and Price

We leveraged pair plots to visualize relationships and distribution between:

- Bedrooms
- Bathrooms
- Log(square feet)
- Log(price)

It revealed a linear relationship between log(square feet) and log(price). There was a positive trend -> as price(log) increased so did log(square feet).



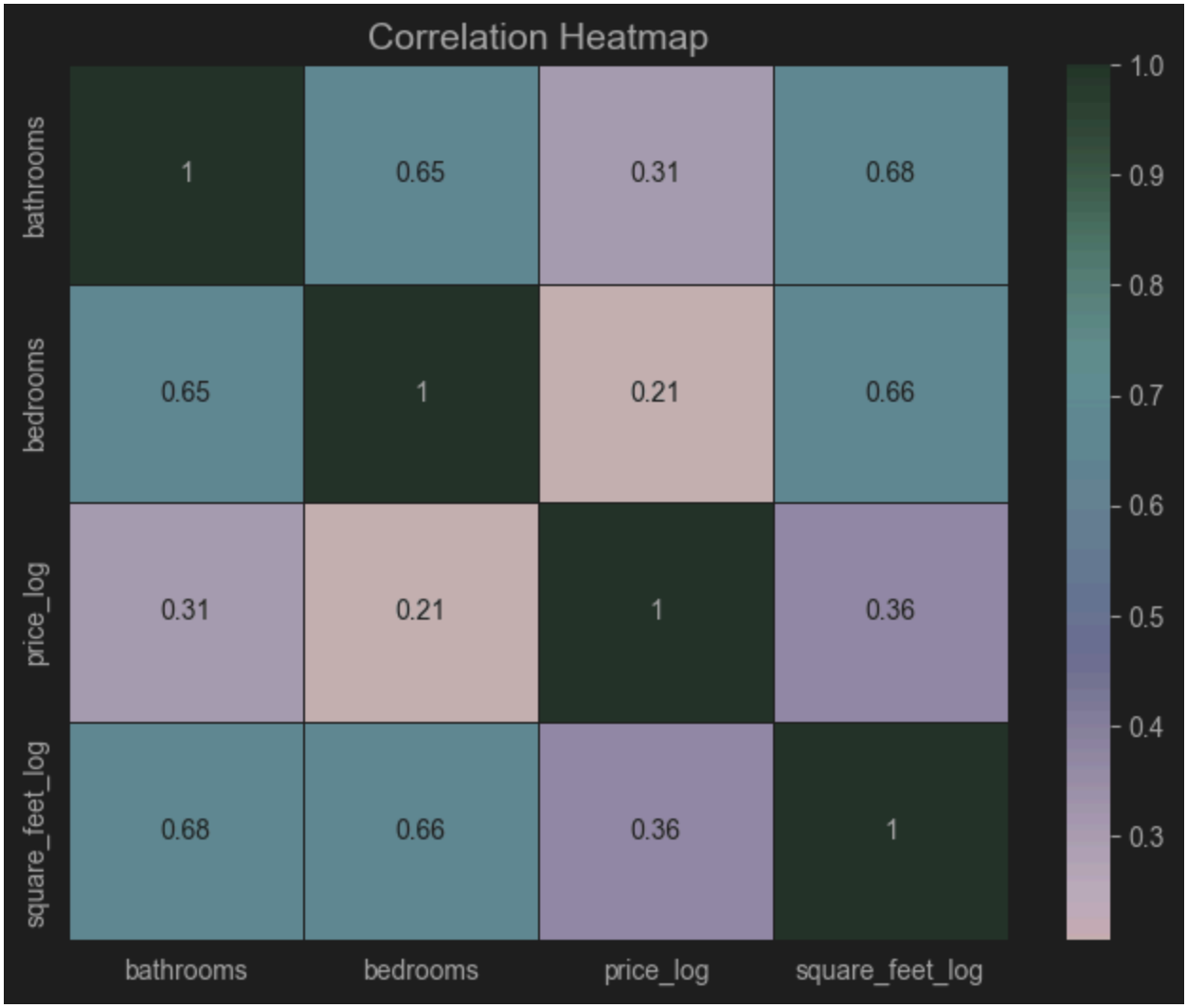
Pearson Correlation

- Pearson's correlation coefficient was used to quantify the strength of linear association between all feature pairs.
- Although the counts of bedrooms and bathrooms are discrete and ordinal, their range [0–9] allowed the resulting correlation coefficient to act as a measure of the estimated monotonic relationship.
- This approach helped us compare the strength and direction of association across both the discrete-ordinal variables and the continuous variables (log(price) and log(square feet).

Pair	Correlation	Interpretation
bathrooms ↔ bedrooms	0.65	Strong positive relationship — homes with more bedrooms usually have more bathrooms.
bathrooms ↔ square_feet_log	0.68	Strong positive correlation — larger houses naturally have more bathrooms.
bedrooms ↔ square_feet_log	0.66	Same strong pattern — larger homes have more bedrooms.
price_log ↔ square_feet_log	0.36	Moderate positive relationship — price generally rises with size, but not perfectly (other factors matter).
price_log ↔ bathrooms	0.31	Mild correlation — price increases somewhat with bathroom count, but not linearly.
price_log ↔ bedrooms	0.21	Weak correlation — price doesn't increase as predictably with bedroom count, possibly because extra bedrooms add less marginal value than square footage.

Correlation Matrix of Listings in the United States

- Log(square feet) showed moderate linear association with log(price) (0.36) and was also moderately-strongly correlated with bedrooms (0.68) and bathrooms (0.66).
- Due to the fact that log(square feet) was the only continuous predictor and that it was not linear, including quadratic terms might be reasonable.
- Bedrooms, bathrooms, and square footage were all moderately-strongly correlated (0.65–0.68), which shows they were associated in a consistent way.
- However, these correlations weren't high enough to cause multicollinearity problems.
- So it was reasonable to include all three features in the model at the same time.
- Bedrooms (0.21) and bathrooms (0.31) had relatively weak correlations with log(price), and the scatterplots did not show clear nonlinear patterns.
- Because there was no strong evidence of curvature or a more complex relationship, the simplest reasonable choice was to model these features with linear terms.

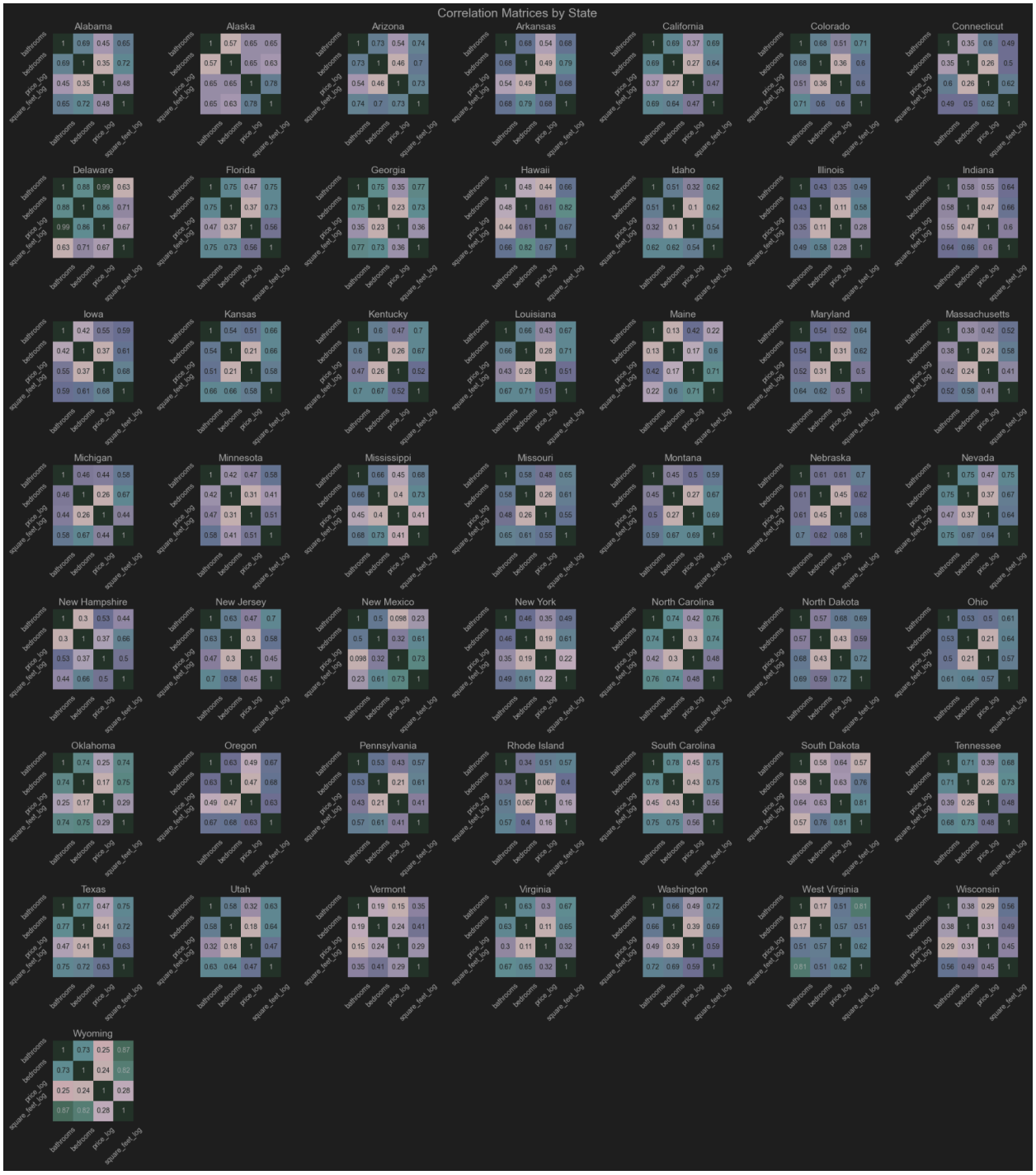


Correlation Matrices by State

We decided to group by state and examine the correlation and found various strengths of linear relationships between each structural feature: bedroom, bathroom, log(square feet) - against log(price) depending on the state.

- Notable States with Strong Correlation
    - Alaska
    - Hawaii
- Notable States with Moderate Correlation
    - California
    - Minnesota
    - Texas
- Notable States with Weak Correlation

- Delaware
- Indiana



Key Insights

Structural Features

- Square_foot_log, bedrooms, and bathrooms weren't enough to reach our goal.
- These structural features explained some variation in price.
- But they couldn't capture location-driven effects.
- For example, homes with identical size and room counts could differ in price depending on the neighborhood, city, or state.
- We needed spatial features.

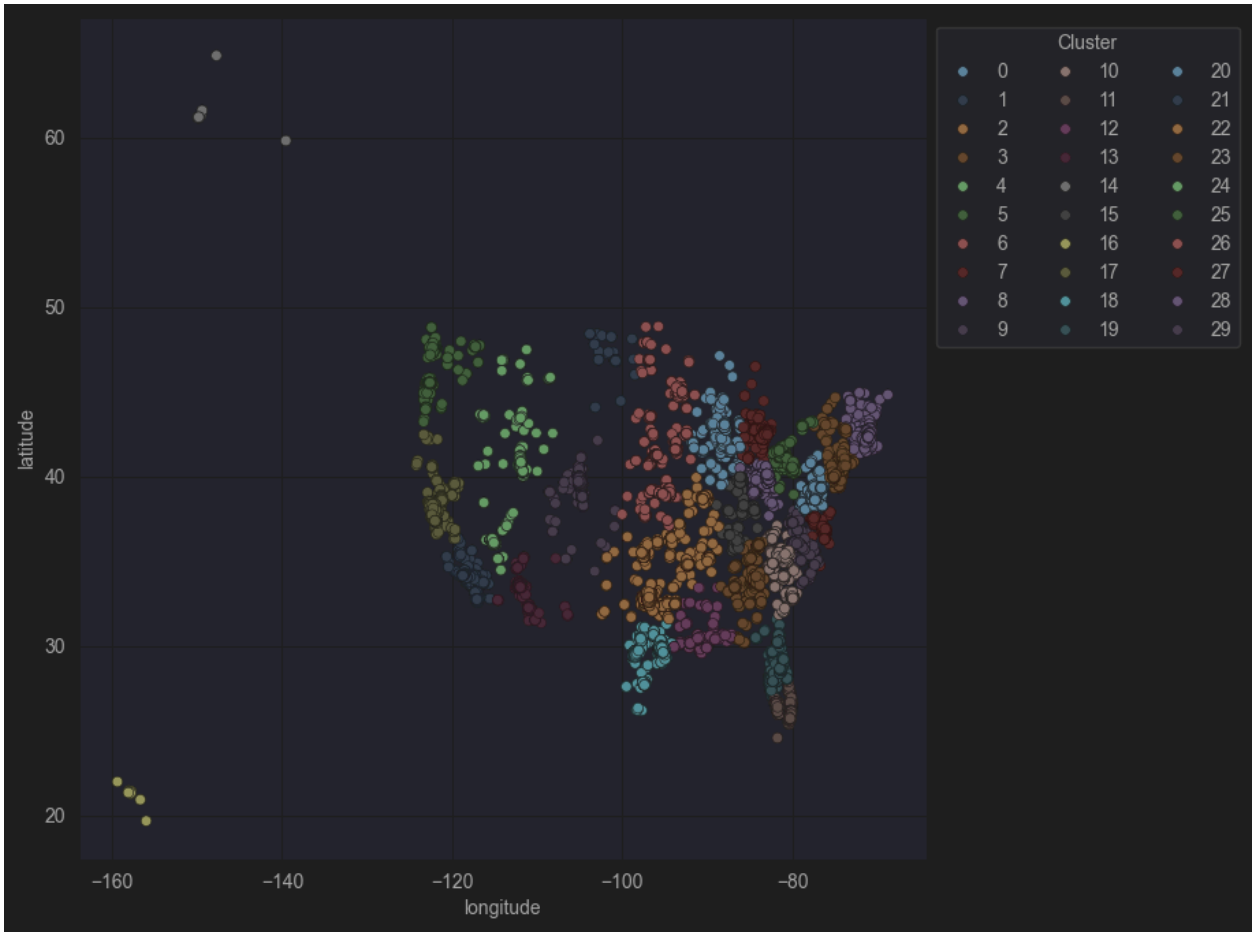
Spatial Features

- We decided to use aspects of spatial features, at macro and micro levels (state, city, and neighborhood).
- Macro Level
    - City and State provided geopolitical boundaries
    - Captured broad market differences such as labor markets, tax structures, regulations, and overall cost-of-living.
- Micro Level
    - We wanted to create neighborhood-like groups that capture these location-specific price differences that macro boundaries could not capture.
- In essence, we decided that clustering features could introduce spatial structure enabling our model to adjust baseline rental price prediction levels across regions and more accurately represent real-world housing markets.

K Means Clustering

- We generated clusters by latitude and longitude and obtained 30 clusters.
- Clusters spanned across a minimum of 1 city and state and maximum of 397 cities and/or 7 states.
- These clusters further encoded spatial information across the United States, from Hawaii to Alaska to the intercontinental states.

```
clusterer = KMeans(n_clusters=30, random_state=42)
cleaned_subset_df['geo_cluster'] = clusterer.fit_predict(cleaned_subset_df[['latitude', 'longitude']])
✓ [136] 58ms
```



Linear Regression Model

In summary, we represented our model by:

$$\widehat{log(price)} = \beta_0 + \beta_1 log(square\ feet) + \beta_2 (square\ feet)^2 + \beta_3 (bedrooms) + \beta_4 (bathrooms) + C(city) + C(kmcluster(latitude.longitude)$$

where $C(x)$ = one hot encoding of categorical variables

Loss Function

Decision context

- Our aim was to estimate rental log-price based on apartment characteristics (structural and spatial), we selected Ordinary Least Squares (OLS) because it is designed to capture central trends rather than extreme cases.

- We valued a model that would be generally good at predicting apartment rental log-prices of the market.

Consequences of Prediction Errors

- Due to the fact that squared residuals heavily penalizes large mistakes and that OLS naturally emphasizes reducing large pricing errors, it aligns with the practical costs of mispricing homes which could result in financial losses of renters and owners.

Performance Measurement

- OLS connects directly to standard regression metrics (RMSE, R^2) and provides interpretable coefficients, making performance assessment straightforward.

Heteroskedasticity & Outlier Mitigation

- We performed log transformation to price and square foot

Pipeline

Our pipeline applied column transforms to the input features by using a column transformer before input into the linear regression model. Column transforms included the following:

- generating a new feature matrix consisting of all polynomial combinations of the log of square feet feature with degree less than or equal to 2, without including bias; and
- encoding the categorical features, cluster and city, as one hot numeric array.

We utilized an 80%-20% training-test split to fit the model and evaluated its performance across various regression loss functions including but not limited to R^2 score, mean square error (MSE) and mean absolute error (MAE). Our goal was to assess its performance with respect to variability, accuracy, and sensitivity to outliers.

```
X = cleaned_subset_df[['square_feet_log', 'bedrooms', 'bathrooms', 'state', 'geo_cluster', 'cityname']]
y = cleaned_subset_df['price_log']

X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size=0.2,
    random_state=42,
)

preprocessor = ColumnTransformer([
    ("square_feet_log_poly", PolynomialFeatures(degree=2, include_bias=False), ['square_feet_log']),
    ('structura_linear', 'passthrough', ['bedrooms', 'bathrooms']),
    #("state_ohe", OneHotEncoder(handle_unknown='ignore'), ['state']),
    ('city_encoded', OneHotEncoder(handle_unknown='ignore'), ['cityname']),
    ('neighborhood_cluster_encoded', OneHotEncoder(handle_unknown='ignore'), ['geo_cluster'])
])

model = Pipeline([
    ('preprocess', preprocessor),
    ('linreg', LinearRegression()),
])


model.fit(X_train, y_train)

y_train_pred = model.predict(X_train)
y_test_pred = model.predict(X_test)
```

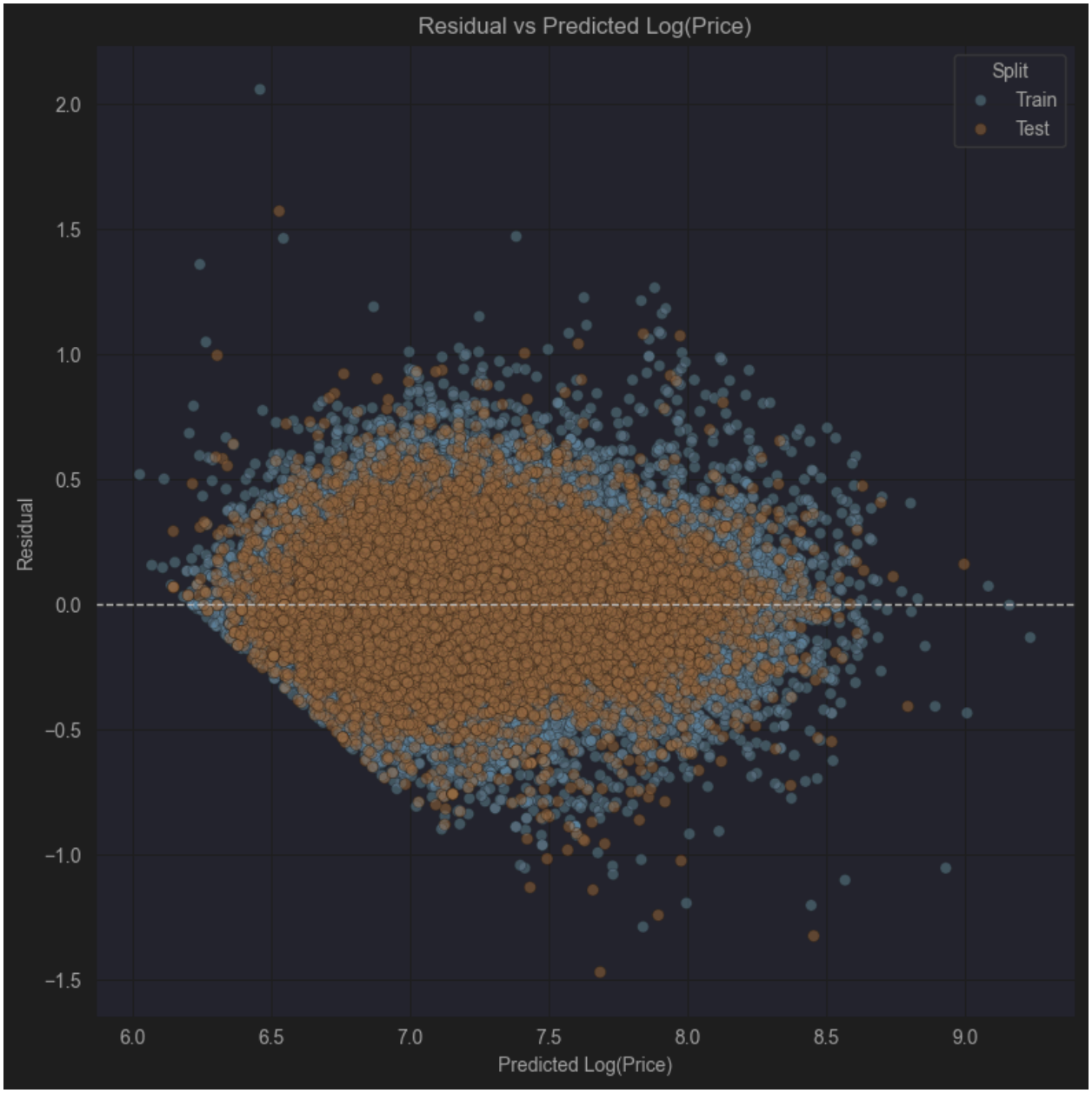
Training Metrics Table			
Metric	Value	$e^{(value)}$	Interpretation
R^2	0.78	-	% of variance explained. The model captures ~78% of training variation
Mean Squared Error	0.04	1.04	Average squared error. The model is off by ~4% on average in true price
Root Mean Square Error	0.20	1.22	predictions are typically off by ~22% in true price
Mean Absolute Error	0.15	1.16	On average, predictions are off by ~16% in true price

Mean Absolute Percent Error	0.02	1.02	Mean Absolute Percentage Error. ~2% if the target is scaled, or ~2% error in log space. Very low. Consistent accuracy.
Median Absolute Error	0.11	1.12	For half the homes, predictions are off within ~12% in true price

Test Metrics Table			
Metric	Value	$e^{(value)}$	Interpretation
R^2	0.78	-	% of variance explained. The model captures ~78% of unseen variation. strong fit, low overfitting.
Mean Squared Error	0.04	-1.04	Average squared error. The model is off by ~4% on average in true price.
Root Mean Square Error	0.20	1.22	predictions are typically off by ~22% in true price
Mean Absolute Error	0.15	1.16	On average, predictions are off by ~15% in true price
Mean Absolute Percent Error	0.02	1.02	Mean Absolute Percentage Error. ~2% if the target is scaled, or ~2% error in log space. Very low. Consistent accuracy.
Median Absolute Error	0.11	1.12	For half the homes, predictions are off

			within ~12% in true price

In addition, we assessed the model using a residual plot of the residuals vs the predicted log(price) of the training and test data.



- Residuals stayed centered around zero and ranged between 2 and -2
- Spread is fairly uniform, heteroskedasticity largely reduced by log transform.
- Train and test points overlap, good generalization, no obvious overfitting.
- No visible curvature model form is reasonably appropriate.

- A few large residuals appear, but not enough to distort the overall pattern.
- Slight right skewed

Classification Model

Objective

- Classify apartment rental listings as mispriced (in either direction) or “fair”

Theory of Operations

Note: our classifier is a continuation of the predicted log(price) derived from the linear regression model, $\widehat{log(price)}$

We have log of actual price:

- $log(price)$

Our linear regression model predicted log “price/fair market value”

- $\widehat{log(price)}$

We wanted to classify rental apartments that are mispriced in either direction

- overpriced or underpriced
- using our linear regression model as the “market baseline”

So, we used absolute percentage error vs the model in these steps:

1. Converted price from log-space to price ratio:
    a. $price\ ratio\ =\ \frac{price}{predicted\ price}\ =\ e^{(log(\ price) - \widehat{log(\ price)})}$
2. Turn that into an absolute percentage difference vs model:
    a. $mispricing\ percent\ =\ |price\ ratio\ -\ 1|$

For example:

- When mispricing percent = 15% the actual price is ±15% off the model’s fair price

We labeled the output of the binary classifier as

- 1
    - Overpriced or
    - Underpriced
    - Mispriced
- 0
    - Not overpriced

- Not underpriced
- "Fair"
- Within acceptable tolerance (e.g., ±15%
  - If mispricing_pct ≤ threshold

In summary, we represented our model by:

- $\widehat{log(price)} = \beta_0 + \beta_1 log(square\ feet) + \beta_2 (square\ feet)^2 + \beta_3 (bedrooms) + \beta_4 (bathrooms) + C(city) + C(kmcluster(latitude.longitude)$

  where $C(x) =$ one hot encoding of categorical variables
- $price\ ratio = \frac{log(price)}{\widehat{log(price)}} = e^{(log(price) - \widehat{log(price)})}$
- $mispricing\ percent = |price\ ratio - 1|$
- $mispriced = \{1, if\ |price\ ratio| > 0.15\}$
  $\{0, if\ |price\ ratio|\} \leq 0.15\}$

Loss Function

Decision context

- Log loss seemed like a reasonable choice because we cared about probabilities of mispricing, not just hard class labels, and log loss tends to discourage overconfident wrong predictions
- We treated our linear regression model as an approximate fair market value baseline in log-price space.
- The classification model could then answering:
  "Given the features, how likely is this home to be more than ±15% away from that baseline?"
- The model is not claiming to know the true "correct" price. It is flagging listings that may disagree with our baseline model beyond a chosen tolerance that we considered practically meaningful.

Consequences of Prediction Errors

- False Positive (flagged as mispriced but actually within ±15%)
  - Leads to unnecessary alerts and extra review. This adds friction but may be acceptable if the cost of missing true mispricings is higher.
- False Negative (classified as fair but truly > ±15% off)
  - Potentially more costly: we might miss overpricing (renter overpays) or underpricing (owner loses gains).

We use the model as a decision support signal, not a definitive decision maker.

Performance Measurement

We evaluated the performance of our classification model with the following metrics:

- Accuracy for an overall sanity check.
- Precision, recall, and F1 for the mispriced class to understand:
    - how trustworthy an alert is (precision),
    - how many mispriced homes we actually caught (recall),
    - and their balance (F1).
- The confusion matrix to map these numbers back to real-world outcomes (TP, FP, FN, TN).

This gave us a way to discuss trade-offs rather than a single good/bad score.

Heteroskedasticity & Outlier Mitigation

Several decisions were motivated by heteroskedasticity and outliers:

- Working in log-price space reduced variance growth with price level and right skewness.
- Using percentage deviation (mispricing_pct) normalized error relative to price, so a $50k error on a $300k home is treated differently than on a $3M home.

These steps may not fully solve issues of heteroskedasticity, but enable us to approach the problem more strategically, and be interpretable for our classification task.

Pipeline

- We first converted the continuous log-price predictions from our baseline regression model into a binary “mispriced” label. Using the difference between the actual and model-predicted log prices, we computed the implied ratio of actual price to model fair value and defined a symmetric percentage deviation from the model. Properties whose actual prices were more than 15% above or below the model’s fair value were labeled as mispriced, while all others were labeled as fair.
- Using these derived labels, we constructed a classification pipeline that applied column-wise transformations before fitting a gradient boosted tree classifier. The preprocessing stage generated polynomial features up to degree two for the log of square footage (followed by standardization), standardized the structural numeric features (bedrooms and bathrooms), and encoded the categorical features (city and neighborhood cluster) using one-hot encoding. These transformed features were then passed to a histogram based gradient boosting classifier with class balancing to better handle the relative rarity of mispriced homes.
- We utilized an 80/20 stratified train, test split to preserve the proportion of mispriced versus fair homes in both sets, fit the model on the training data, and evaluated its performance on the held-out test set. Classification performance was summarized using

accuracy, precision, recall, F1 score for both classes (fair and mispriced), as well as the confusion matrix. These metrics allowed us to assess not only overall correctness but also the trade-off between correctly identifying mispriced homes and avoiding unnecessary false alerts on fair homes, with particular attention to the real-world implications of false positives and false negatives.

```
# ----------------------------------------------------------------
# 1. Define "mispriced" using log prices (either direction)
# ----------------------------------------------------------------

X = X_train  # your feature matrix (DataFrame)
log_price_true = y_train          # actual log price
log_price_pred = y_train_pred     # model-predicted log fair value

# actual / predicted (in price space)
price_ratio = np.exp(log_price_true - log_price_pred)

# symmetric % difference vs model: |(actual - pred) / pred|
mispricing_pct = np.abs(price_ratio - 1.0)

# ----------------------------------------------------------------
# 2. Choose what "mispriced" means > 15% off model in either direction
# ----------------------------------------------------------------

MISPRICED_THRESHOLD = 0.15
y_mispriced = (mispricing_pct > MISPRICED_THRESHOLD).astype(int)
```

```
# ----------------------------------------------------------------------
# 3. Train/test split (only once, stratified)
# ----------------------------------------------------------------------

X_train_split, X_test_split, y_train_split, y_test_split = train_test_split(
    X,
    y_mispriced,
    test_size=0.2,
    random_state=42,
    stratify=y_mispriced,
)

numeric_poly = Pipeline([
    ("poly", PolynomialFeatures(degree=2, include_bias=False)),
    ("scaler", StandardScaler())
])

numeric_struct = Pipeline([
    ("scaler", StandardScaler())
])

preprocessor = ColumnTransformer([
    ("square_feet_log_poly", numeric_poly, ['square_feet_log']),
    ("structural_linear", numeric_struct, ['bedrooms', 'bathrooms']),
    ("city_encoded", OneHotEncoder(handle_unknown="ignore", sparse_output=False), ['cityname']),
    ("neighborhood_cluster_encoded", OneHotEncoder(handle_unknown="ignore", sparse_output=False), ['geo_cluster']),
])

classifier = HistGradientBoostingClassifier(
    random_state=42,
    class_weight="balanced",
    max_depth=6,
    learning_rate=0.05,
    max_iter=300
)
```

```
pipeline_cls = Pipeline([
    ("preprocess", preprocessor),
    ("logreg", classifier),
])

pipeline_cls.fit(X_train_split, y_train_split)
```

```
# ------------------------------------------------------------------
# 4. Evaluate at probability threshold
# ------------------------------------------------------------------
y_pred = pipeline_cls.predict(X_test_split)
```

Results

Accuracy	0.67

	Metric	Precision	Recall	F1
Class	0	0.72, When the model predicts a home is FAIR, it is correct 72% of the time.	0.78, It correctly identifies 78% of all fair homes.	0.75
Class	1	0.57, When the model predicts MISPRICED, it is correct 57% of the time.	0.50, It successfully catches 50% of all mispriced homes.	0.53

Confusion Matrix		
	N	P
F	2886, Model missed mispriced homes (risky).	2167, Model flagged fair homes as mispriced (unnecessary alerts).
T	7475, Model correctly recognized fair homes.	2834, Model caught mispriced homes (good).

Conclusion

- This project set out to understand and predict U.S. apartment rental prices using both structural and spatial features, and to identify potentially mispriced listings. After extensive data cleaning, outlier handling, and log transformations to reduce skewness, we built a linear regression model incorporating square footage (with polynomial terms), bedroom and bathroom counts, and location-based categorical encodings (city and K-means geographic clusters).
- The resulting model explained over 75% of the variance in the test set, demonstrating strong predictive capability for estimating log-price. We then used the model's predicted log-price as an approximate "fair market value" baseline to construct a binary classifier for detecting mispriced listings. This classification system was able to flag many mispriced homes but also produced a significant number of false positives and false negatives possibly due to dataset imbalance, regional representation issues, and modeling assumptions.
- While both models performed reasonably well from a statistical perspective, the limitations of the dataset, including uneven state representation, ambiguous or synthetic listings, and assumptions made throughout preprocessing, introduce enough concern that we would be cautious about deploying these models to real world end users without further refinement, validation, and improved data quality.