Predicting House Sale Price in Pierce County

Question: How can the features of a home help accurately predict sale price?
Dataset: Pierce County home sales, 2020.

Avalon, Rebecca, and Thomas

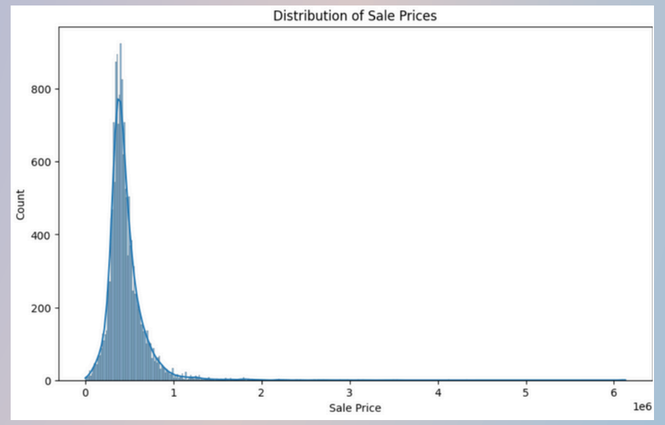
```
<class 'pandas.core.frame.DataFrame'>
Index: 16797 entries, 0 to 16813
Data columns (total 19 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   sale_date                     16797 non-null  datetime64
 1   sale_price                    16797 non-null  int64
 2   house_square_feet             16797 non-null  int64
 3   attic_finished_square_feet    16797 non-null  int64
 4   basement_square_feet          16797 non-null  int64
 5   attached_garage_square_feet   16797 non-null  int64
 6   detached_garage_square_feet   16797 non-null  int64
 7   fireplaces                    16797 non-null  int64
 8   hvac_description              16797 non-null  category
 9   exterior                      16797 non-null  category
 10  interior                      16797 non-null  category
 11  stories                       16797 non-null  int64
 12  roof_cover                    16797 non-null  category
 13  year_built                    16797 non-null  int64
 14  bedrooms                      16797 non-null  int64
 15  bathrooms                     16797 non-null  int64
 16  waterfront_type               16797 non-null  category
 17  utility_sewer                 16797 non-null  category
 18  sale_month                    16797 non-null  int32
dtypes: category(6), datetime64[ns](1), int32(1), int64(11)
memory usage: 1.8 MB
```

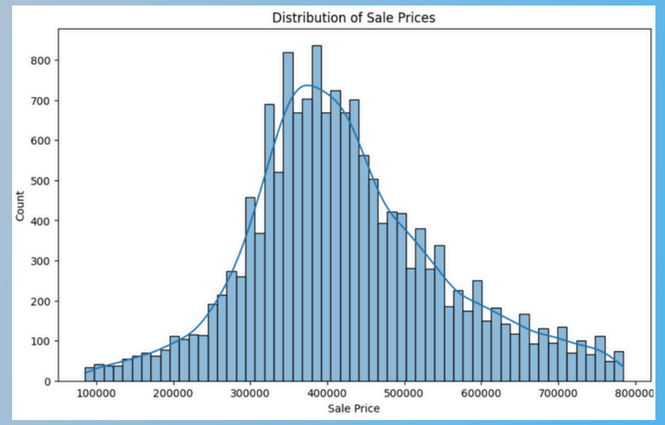
Our Method: Decision Tree Regression

Why we chose it

- Chose Regression Decision Tree for flexibility (continuous target, mixed features).
- Cleaned and simplified data (e.g., dropped "view quality", categorized features).
- Tuned hyperparameters with GridSearchCV to improve performance.

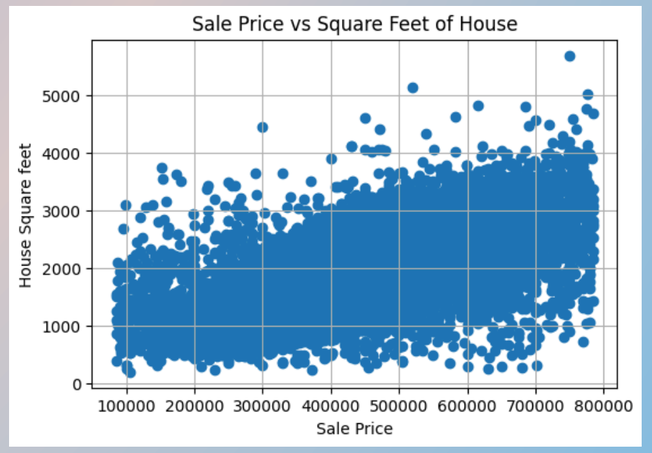
Predictive Features

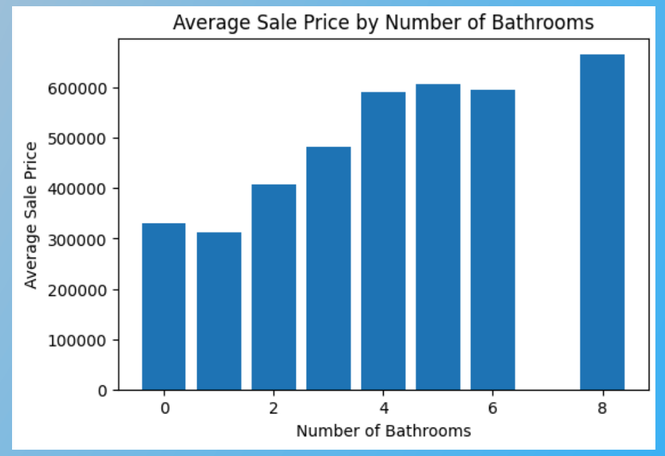




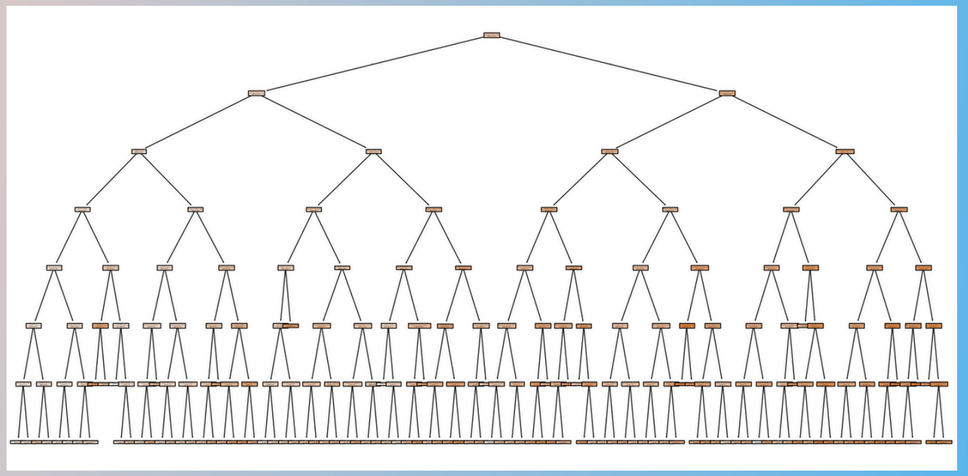
Key Insights from EDA

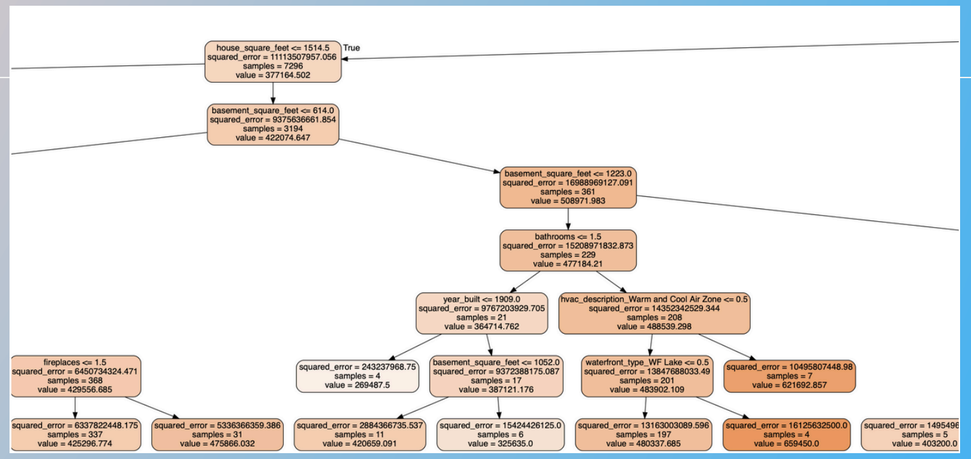
- Sale price distribution was right-skewed; removed outliers using IQR.
- Larger homes (house square footage) strongly correlated with higher prices.
- Bathrooms and year built also important predictors.
- Simplified rare categories for better model fitting.





How Well Did Our Model Perform?





Decision Tree Regressor Results

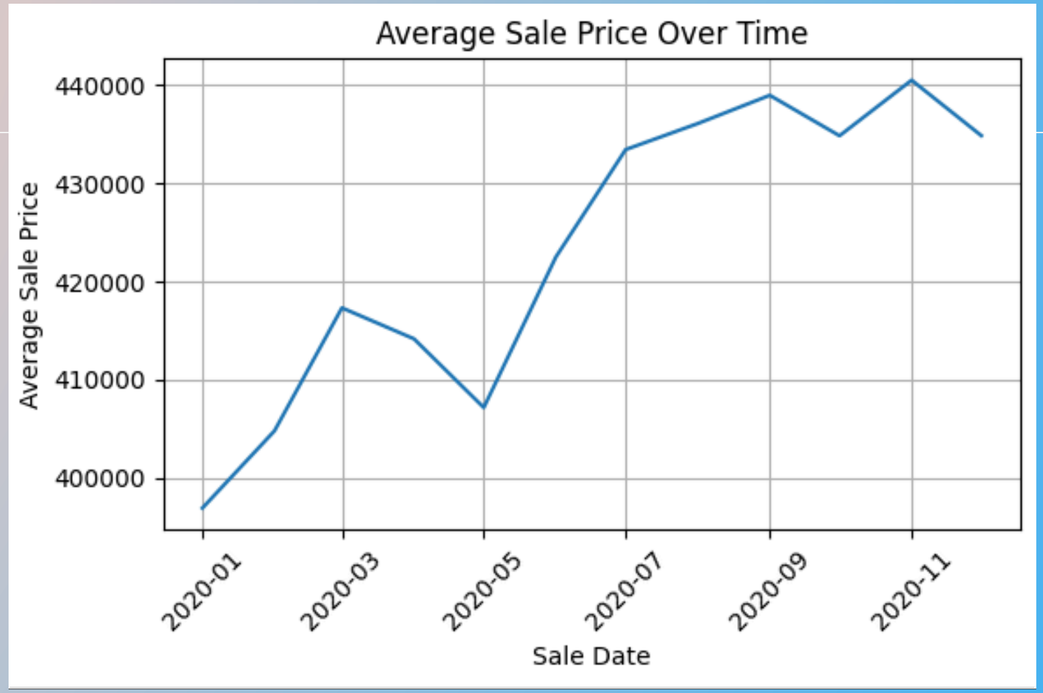
- Model explains about 52% of variance in sale prices.
- RMSE around $89k shows reasonable prediction error size.

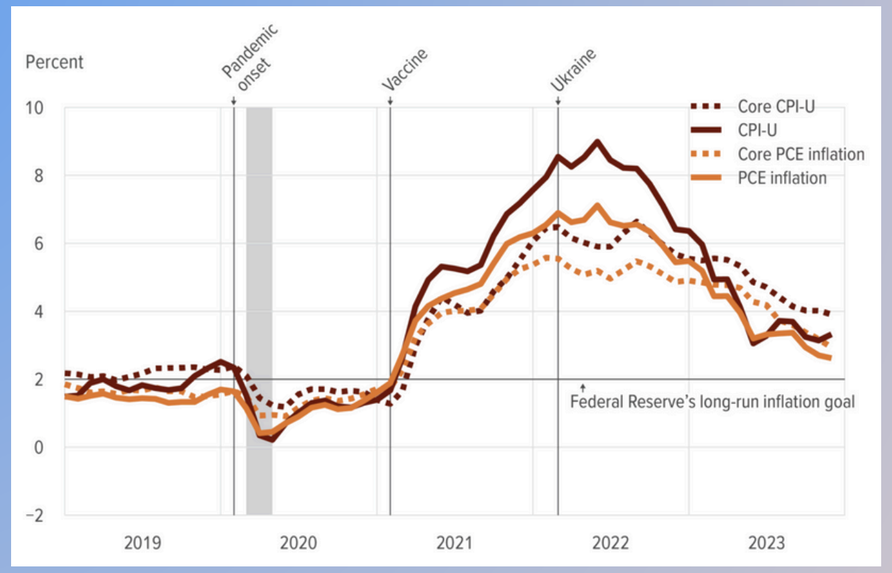
	Train (Model 1)	Tune (Model 2)	Test (Best Model)
RMSE	83683.582031	90340.054829	88779.232797
R^2	0.580644	0.511104	0.523080

Challenges and Constraints

Our Limitations

- Data from 2020 may reflect COVID-driven buyer behavior (e.g., higher demand for space).
- Model does not adjust for inflation or broader economic changes.
- Model focused on house features, not external economic forces.





Next Steps for Improvement

Future Work

- Explore later years (2021–2022) to test generalization.
- Compare model performance during major market events (e.g., 2008 crash).
- Adjust for inflation to normalize sale prices across time.
- Try binning sale price (classification) to simplify prediction.

Appendix

```python
## Dropping unnecessary column (hard to work with)
house_sales = house_sales.drop(columns=['view_quality'])

# Convert 'sale_date' to datetime
house_sales['sale_date'] = pd.to_datetime(house_sales['sale_date'])

#adding a month category
house_sales['sale_month'] = house_sales['sale_date'].dt.month

# Categorical conversions
house_sales['hvac_description'] = house_sales['hvac_description'].astype('category')


# Simplify 'exterior' categories
house_sales['exterior'] = house_sales['exterior'].apply(
    lambda x: x if x in ["Frame Siding", "Frame Vinyl", "Masonry Common Brick", "Frame Stucco"] else "Other"
)

#changing into categorical
house_sales['exterior'] = house_sales['exterior'].astype('category')
house_sales['utility_sewer'] = house_sales['utility_sewer'].astype('category')
house_sales['interior'] = house_sales['interior'].astype('category')
house_sales['roof_cover'] = house_sales['roof_cover'].astype('category')

# Handle 'waterfront_type'
house_sales['waterfront_type'] = house_sales['waterfront_type'].astype(object)
house_sales['waterfront_type'] = house_sales['waterfront_type'].fillna('Not on waterfront')
house_sales['waterfront_type'] = house_sales['waterfront_type'].astype('category')

#dropping na
house_sales = house_sales.dropna()

# Preview data
house_sales.info()
```

Cleaning Function

```python
#Define Features and Target
# Drop only existing columns: sale_price (target) and sale_date (timestamp and model cant use)
X = house_sales.drop(columns=['sale_price', 'sale_date'])

# One-hot encode categorical variables
X = pd.get_dummies(X, drop_first=True)

# Define Target
y = house_sales['sale_price']

# STEP 6: Train/Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.70, random_state=21)
X_tune, X_test, y_tune, y_test = train_test_split(X_test,y_test,  train_size = 0.50,random_state=42)

kf = RepeatedKFold(n_splits=10,n_repeats =5, random_state=42)

param={
    "max_depth" : [3,5,7,9,11],
    "splitter":["best","random"],
    "min_samples_leaf":[1,2,4],
    'ccp_alpha':[.001,.01,.1]
      }


scoring= ['neg_mean_squared_error','r2','neg_mean_absolute_error']
reg=DecisionTreeRegressor(random_state=42)
search = GridSearchCV(reg, param, scoring=scoring, n_jobs=-1, cv=kf,refit='r2')

model = search.fit(X_train, y_train)
best = model.best_estimator_
print(best)
```

Training Model

```
print("\nSummary Statistics:\n", house_sales.describe())
```

```
Summary Statistics:
                           sale_date    sale_price  house_square_feet  \
count                          16797  1.679700e+04       16797.000000
mean   2020-07-16 14:02:33.027326464  4.609776e+05        1880.764363
min              2020-01-01 00:00:00  2.000000e+03           1.000000
25%              2020-04-27 00:00:00  3.480000e+05        1320.000000
50%              2020-07-27 00:00:00  4.163430e+05        1774.000000
75%              2020-10-09 00:00:00  5.235450e+05        2352.000000
max              2020-12-31 00:00:00  6.130000e+06        9510.000000
std                              NaN  2.342215e+05         759.762716

       attic_finished_square_feet  basement_square_feet  \
count                16797.000000          16797.000000
mean                    24.902364            167.925641
min                      0.000000              0.000000
25%                      0.000000              0.000000
50%                      0.000000              0.000000
75%                      0.000000              0.000000
max                   1212.000000           4000.000000
std                    101.885451            429.103156

       attached_garage_square_feet  detached_garage_square_feet    fireplaces  \
count                 16797.000000                 16797.000000  16797.000000
mean                    364.635292                    38.320117      0.889445
min                       0.000000                     0.000000      0.000000
25%                       0.000000                     0.000000      1.000000
50%                     420.000000                     0.000000      1.000000
75%                     528.000000                     0.000000      1.000000
max                    2816.000000                  3664.000000      5.000000
std                     286.374648                   164.759289      0.596574

            stories    year_built      bedrooms     bathrooms    sale_month
count  16797.000000  16797.000000  16797.000000  16797.000000  16797.000000
mean       1.559028   1980.469012      3.280169      2.318390      7.004703
min        0.000000   1880.000000      0.000000      0.000000      1.000000
25%        1.000000   1959.000000      3.000000      2.000000      4.000000
50%        2.000000   1990.000000      3.000000      2.000000      7.000000
75%        2.000000   2006.000000      4.000000      3.000000     10.000000
max        3.000000   2021.000000     25.000000      8.000000     12.000000
std        0.511287     33.337722      0.888038      0.826711      3.296518
```

Summary Statistics

```python
y_train_pred = best.predict(X_train)

r2_train = r2_score(y_train, y_train_pred)
rmse_train = np.sqrt(mean_squared_error(y_train, y_train_pred))

print(f"RMean Squared Error: {rmse_train:.2f}")
print(f"R2 Score: {r2_train:.2f}")
```

```
RMean Squared Error: 83683.58
R2 Score: 0.58
```

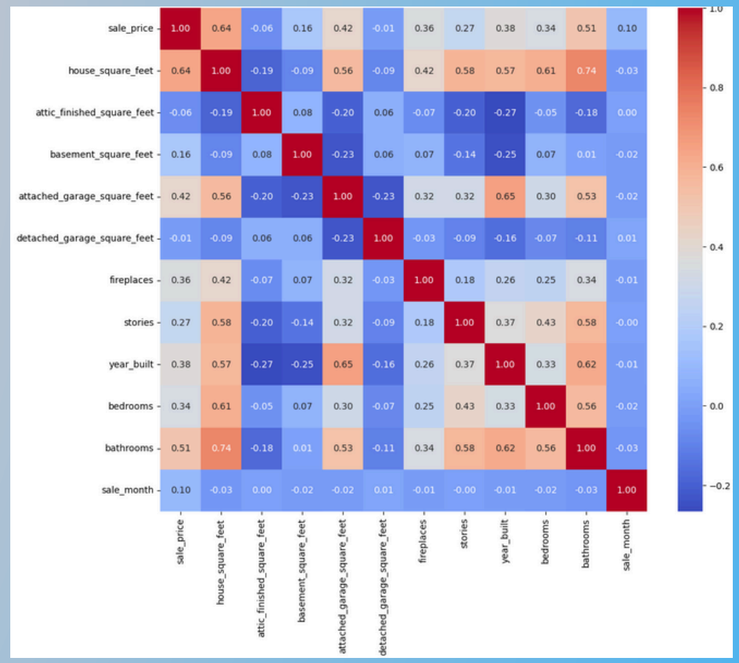
```python
y_tune_pred = top4_model.predict(X_tune_top4)
r2 = r2_score(y_tune, y_tune_pred)
rmse = np.sqrt(mean_squared_error(y_tune, y_tune_pred))

print("Top 4 Features Used:", top4_features)
print(f"Tune R² Score (Top 7 Model): {r2:.4f}")
print(f"Tune RMSE (Top 7 Model): {rmse:.2f}")
```

```
Top 4 Features Used: ['house_square_feet', 'basement_square_feet', 'attached_garage_square_feet', 'bathrooms']
Tune R² Score (Top 7 Model): 0.5111
Tune RMSE (Top 7 Model): 90340.05
```

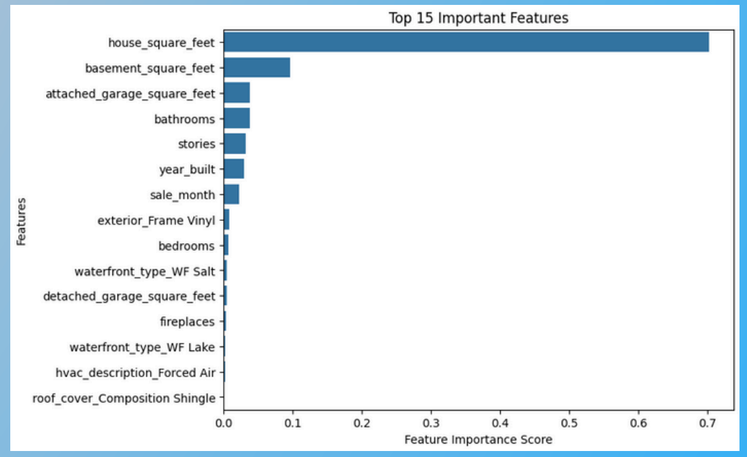
```python
model = search.fit(X_test, y_test)
best= model.best_estimator_
print(best)

y_test_pred = best.predict(X_test)
r2_test = r2_score(y_test, y_test_pred)
rmse_test = np.sqrt(mean_squared_error(y_test, y_test_pred))
print(f"RMean Squared Error: {rmse_test:.2f}")
print(f"R2 Score: {r2_test:.2f}")
```

```
DecisionTreeRegressor(ccp_alpha=0.001, max_depth=5, min_samples_leaf=4,
                      random_state=42)
RMean Squared Error: 88779.23
R2 Score: 0.52
```

Evaluation Metrics



Correlation Matrix for Numerical Features



Feature Importance