U.S. Listed Houses Model

GitHub Repo | Demo Link

1. Project Definition

1.1 Problem Statement

Buying a home in the modern housing market nowadays is a tough and confusing process. The overall market is inflated right now, and the listed prices of homes are overpriced. My project focuses on analyzing real estate data from Zillow to determine if the house listings in the United States are overpriced, underpriced, or fair. I will create visualizations to help illustrate patterns in the housing market and show how common overpriced listings are.

1.2 Connection to Course Material

My project relates to this course's material because it implements Python libraries such as pandas, numpy, matplotlib, seaborn, and SQL using sqlite 3. My project also creates visualizations based on the Zillow dataset comparing the estimated market value and listed pricings using Train Test Split, MAE, R2 score, Standard Scaler, Gradient Boosting Regressor, and KMeans. In addition, my house model project uses clustering and unsupervised learning to create visualization patterns.

2. Novelty and Importance

2.1 Importance of the Project

While Zillow shows listed prices on their service, my model adds an extra layer by classifying the listed prices as either overpriced, underpriced, or fair and letting the user know if they are getting a fair deal or not.

2.2 Related Work

Zillow offers their own similar version of my model where they give "Zestimates", but they don't go that extra step to use machine learning and tell their users whether or not a listed price is inflated or explain the criteria behind price fairness. They leave the price listings ambiguous and

with little context so that users are not discouraged from buying a house, even if it means they are secretly buying an overpriced home.

3. Progress and Contribution

3.1 Data Utilization

For this project, my model read a publicly available dataset of the 2023 real estate market in the United States on Kaggle. This dataset is under CC0 1.0 license which means that it can be publicly used and edited.

```
print("Original dataset of 2023 United States House Listings on Zillow:","\n\n", df.head())

df.dropna(subset=['ListedPrice', 'MarketEstimate', 'Area', 'Bedroom', 'Bathroom'], inplace=True)
df['Price Difference'] = df['ListedPrice']-df['MarketEstimate']
df['Overpriced'] = df['Price Difference'] > 20000
df['Price per Square Ft'] = df['ListedPrice']/df['Area']
df['Zip'] = df['Zipcode'].astype(str)
```

3.2 Models and Techniques

I used GradientBoostRegressor and built my model to predict the actual market value of houses based on features I defined such as Area, Bedroom, Bathroom, Lot Area, Price per Square Ft. I imported mass absolute error and R2 score into my regression model and recorded my evaluation results. The MAE of this dataset came out to be $58, 889.26 and the R2 score of the dataset was $0.874. My model is essentially saying that on average the price prediction was off about $59,000, which is by far overpriced. In addition, the R2 score suggests that my model is covering about 87.4% of the trend.

```
houselist_model = GradientBoostingRegressor()
houselist_model.fit(X_train, y_train)
prediction = houselist_model.predict(X_test)

mae = mean_absolute_error(y_test, prediction)
r2 = r2_score(y_test, prediction)

print("\n\n" f"Mean Absolute Error of 2023 U.S. House Listings: ${mae:,.2f}")
print(f"R^2 Score of 2023 U.S. House Listings: ${r2:,.3f}")
```

```
Mean Absolute Error of 2023 U.S. House Listings: $58,889.26
R^2 Score of 2023 U.S. House Listings: $0.874
   ListedPrice  Predicted Listing Type of Pricing
0     239900.0      226222.056302            Fair
3     335000.0      306022.225355      Overpriced
5     151000.0      152199.752522            Fair
6     239000.0      259238.291376     Underpriced
7     249900.0      226222.056302      Overpriced
```

3.3 Classification Output

I declared and initialized my model's predictions by labeling overpriced listings as anything $20,000 or above my predicted values and underpriced listings as anything $20,000 or more below my predicted pricing. Anything within $20,000 is declared as fair pricing. I added these columns to the original cleaned_df dataset and saved the output to a downloadable and viewable csv for the user to see.

3.4 SQL Integration

I implemented structured query language into my model by importing SQLite. The way I implemented this into my model was by creating a table where I stored the queries and took note of the location as indexes.

```
house_sql = sqlite3.connect("/home/jcn95/cs439/Final Project/cleaned_df.db")
df_p['Zipcode_cleaned'] = df_p['Zipcode'].astype(str).str.zfill(5)

sql_dtype = {
    'Zipcode': 'TEXT'
}

df_sqlp = df_p[['ListedPrice', 'MarketEstimate', 'Area', 'Bedroom', 'Bathroom', 'Zipcode_cleaned']].rename(columns={'Zipcode_cleaned': 'Zipcode'})

df_sqlp.to_sql("listings", house_sql, if_exists="replace", index=False, dtype=sql_dtype)

house_query = "SELECT Zipcode, AVG(ListedPrice) as AvgPrice FROM listings GROUP BY Zipcode"
avg_zip = pd.read_sql_query(house_query, house_sql)
print("\nAverage Listed Price within Zipcode:\n", avg_zip.head())
```

3.5 Clustering Analysis

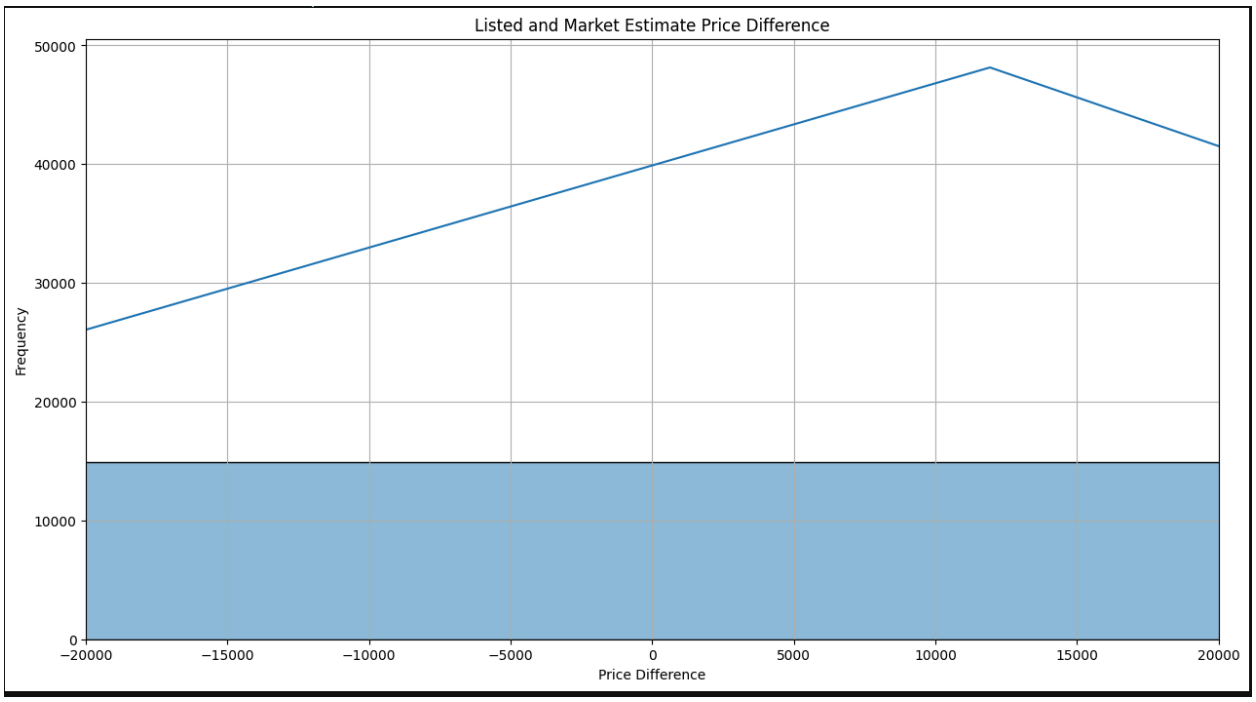
My house model imports KMeans to cluster similar houses by features such as bedroom count and price per square foot.

```
cluster_house = df_p[['Area', 'Bedroom', 'Bathroom', 'Price per Square Ft']]
scluster = StandardScaler().fit_transform(cluster_house)

kmeans = KMeans(n_clusters=4, random_state=40)
df_p['Cluster'] = kmeans.fit_predict(scluster)
sns.boxplot(data=df_p, x='Cluster', y='ListedPrice')
plt.title("Cluster of Price Listings")
plt.show()
```

3.6 Key Findings and Results

The peak price difference between the price listings and the market estimate was about $14,000 with a high frequency of about 48,000 houses. This just goes to show how common it is for United States homeowners to purchase overpriced property. This key finding and result was achieved by my model with the use of sqlite3 and saving my dataframe as an SQL table.



```
avg_zip_price = df_p.groupby('Zipcode')['ListedPrice'].mean().sort_values()
highest_zip_avg = avg_zip_price.sort_values(ascending=False).head(20).reset_index()

plt.figure(figsize=(12,6))
sns.barplot(data=highest_zip_avg, x="Zipcode", y= "ListedPrice", palette="Blues_d")
plt.title("20 Zip Codes with Highest Average Listed Price")
plt.xticks(rotation=45)
plt.xlabel("Zipcode")
plt.ylabel("Average Listed Price (100k)")
plt.tight_layout()
plt.show()
```

4. Changes After Proposal

4.1 Differences from Proposal

Instead of using XGBoost as I stated in my proposal, I used Gradient Boosting Regressor because I found that it achieved a strong variance and still found effective results for mass absolute error and R2 score. New changes that I made to the model that weren't stated in the proposal were price differences, price per square foot, and zip code encoding. I created a csv output and used KDE for my data visualization as well. I didn't implement the Zillow API into my model because there was trouble with access limits.

5. Conclusion and Future Work

5.1 Summary of Contributions

I successfully created a model that labeled the price listings of a dataset as either fair, overpriced, or underpriced. I implemented a visualization of the patterns and trends that I found in the regression model and created a csv with the encoded columns. By integrating SQL queries and clustering, I was also able to make this model convenient for users. These contributions helped my model essentially become an extension of Zillow and add an extra layer of utility with its labeling.

5.2 Future Directions

As stated before I couldn't access Zillow API so in the future I would like to enhance my model by implementing this. I would also like to create a feature where I compare neighborhood house listings and tell the user if a specific house is underpriced, overpriced, or fair compared to the other houses with the same zip code. To sharpen my model a bit I would like to add XGBoost to help improve my mass absolute error and R2 scores even more. Lastly, I plan on creating a public-facing tool where users can upload certain listings and import values into the dataset. By then I would like to create multiple datasets perhaps for 2024 and 2025, not just 2023.