Final Presentation:

Predicting Home Prices

Brandon Law, Kimsean Pen, Addy Kim

Spring 2024

Agenda

1. Quick summary
2. Feature engineering
3. Feature selection
4. Models
5. Conclusion

Quick summary

- Purpose: predict home prices using home features
- Dataset: [Kaggle House Prices]
  - ~1500 observations
  - 80 possible features
  - Y = sales price across 2006 - 2010, time series but adjust all the price to be 2010 equivalent

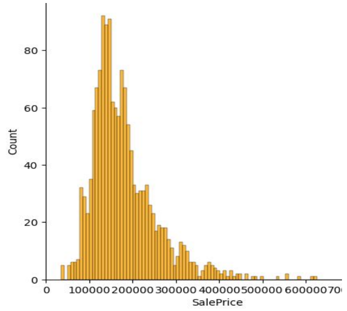
Feature engineering – data cleaning

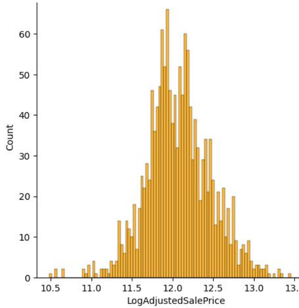
- Prevent bias and improve accuracy before analyzing the data set

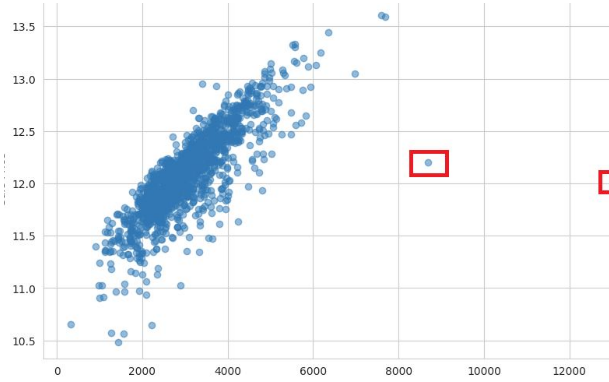
1. Fixing Skew - Normality assumption

2. Fix Outliers - Reduce variance and bias

3. Fill in Missing Values - Remove missing features with ambiguities and replace values with zero for features that used null as an indication of 0

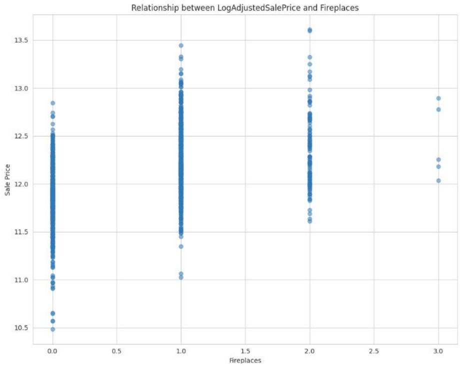


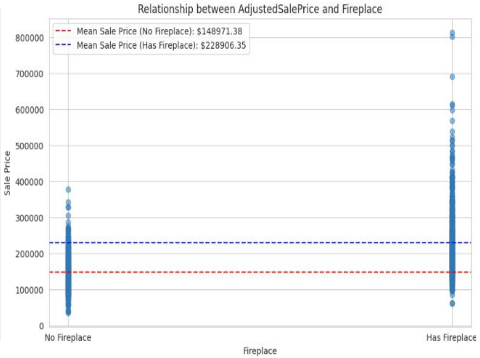




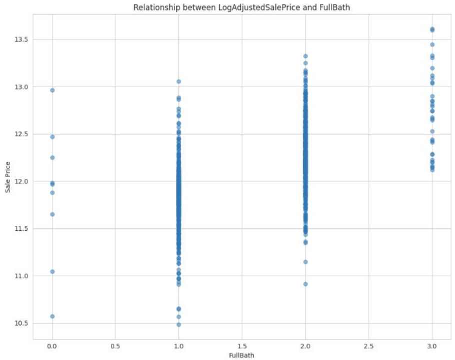
Feature engineering – numerical features

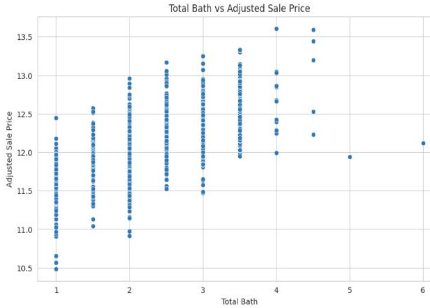
1. Binary Indicators - Measure the impact of having a feature versus missing a feature





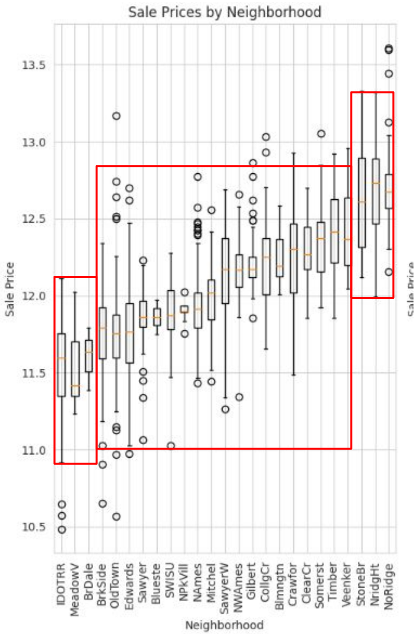
2. Combining Features - Combine segmented features to represent the feature holistically

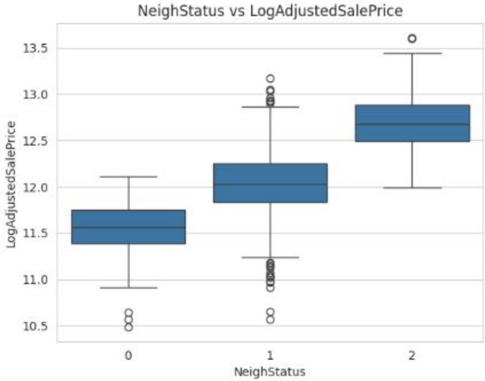


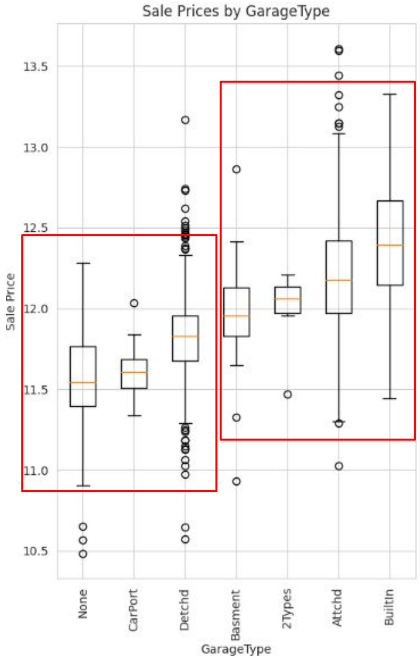


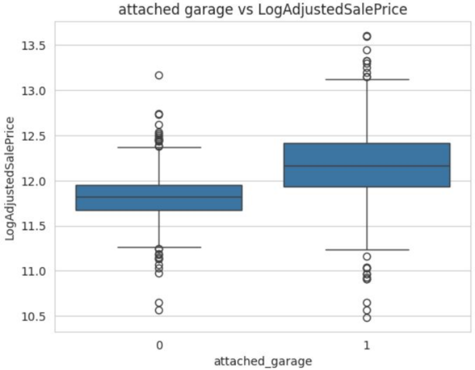
Feature engineering – categorical features

1. Binning Categories - Capture patterns in the data by grouping different entries
2. Numerical Coding







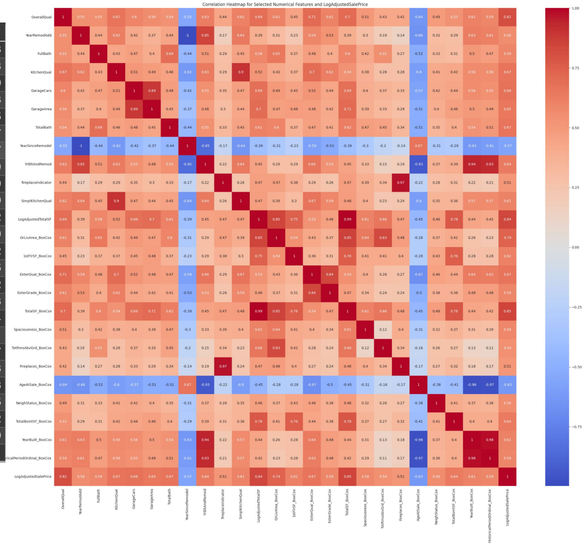


Features selection – approach

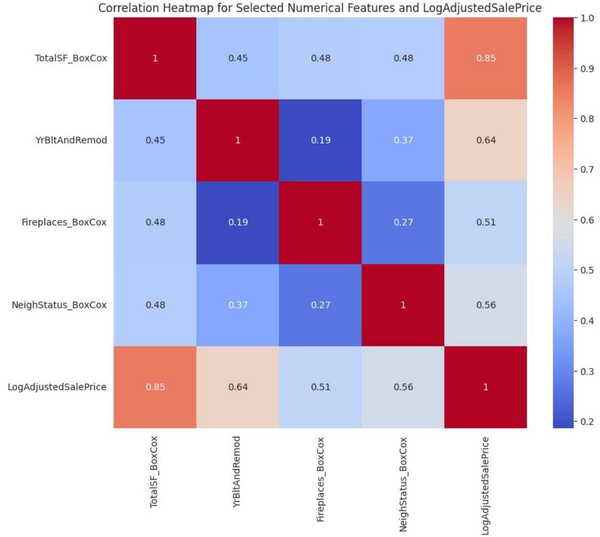
- Anova for Categorical Values
- Looked for potential categorical features with p-values that are less than 0.05

index	Feature	F-Value	P-Value
0	MSZoning	76.09874161419555	1.2348292407790178e-58
1	Street	5.093013889943343	0.02417014292143628
2	LotShape	47.11495731999076	4.6691607072613584e-29
3	LandContour	13.001694169604086	2.2103719216230244e-08
4	Utilities	0.3042266785218487	0.5813293630132635
5	LotConfig	8.530654665291456	8.380426513326464e-07
6	LandSlope	1.0852274394817265	0.33809822643782717
7	Neighborhood	78.39960490045536	1.6163720243135142e-240
8	Condition1	7.926468650266923	1.728982853279264e-10
9	Condition2	2.7994027454066646	0.006762195821970079
10	BldgType	14.956797071243894	5.523711143482286e-12
11	HouseStyle	23.39739767756993	2.9209989839682343e-30
12	RoofStyle	12.835394031808207	3.106586555604552e-12
13	RoofMatl	4.511856475733592	0.00015391124903527665
14	Exterior1st	22.364261630362655	1.980445030052059e-52
15	Exterior2nd	19.6686681922265	1.1136233278256722e-48
16	MasVnrType	109.99546943910745	3.4752003080928155e-64
17	Foundation	126.03198345669819	5.4814758909754024e-111
18	BsmtQual	298.6134840675518	1.6520634267934114e-187
19	BsmtCond	34.96903278162922	6.490743133607709e-28
20	BsmtExposure	61.82615871984261	2.7310205426382175e-48
21	BsmtFinType1	68.9326234727567	1.2552561352552739e-75
22	BsmtFinType2	11.141404858004082	3.50105576295128e-12
23	Heating	9.987102628574593	2.07129259047947e-09
24	HeatingQC	108.90611674616885	3.111488391041952e-81

- Found feature correlation relative to our target - log adjusted sale price

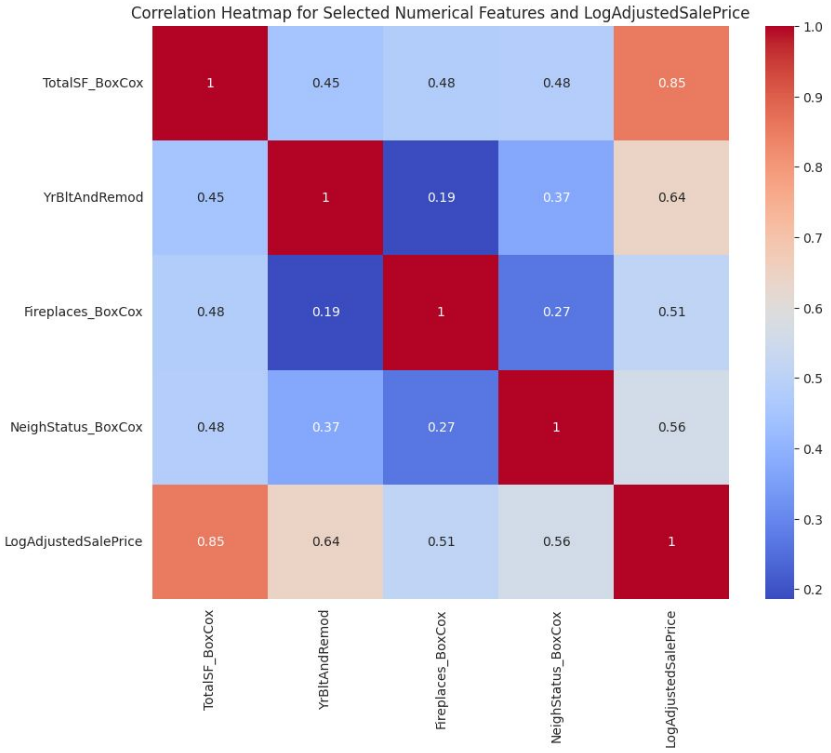


- Focused on features with high correlation to logadjustedsaleprice.
- Removed feature that was highly correlated with each other.



Features selection

- Post EDA and feature engineering, we chose features that:
  1. Have the highest correlation (>0.5) with LogAdjustedSalePrice
  2. Do not have highest correlation (>0.5) with other features
- The final list of features:
  1. TotalSF_BoxCox
  2. YrBltAndRemod
  3. Fireplaces_BoxCox
  4. NeighStatus_BoxCox



Initial Modeling – baseline & linear regression

Baseline

- We used the **median** LogAdjustedSalePrice as our baseline model
- Baseline model LogAdjustedSalePrice: 12.05
- Baseline RMSE: 0.40

Linear regression

- We used the Scikit-Learn's LinearRegression
- 730 examples in training, 728 examples in validation
- Linear Regression RMSE train: 0.17
- Linear Regression RMSE valid: 0.16

Random Forest

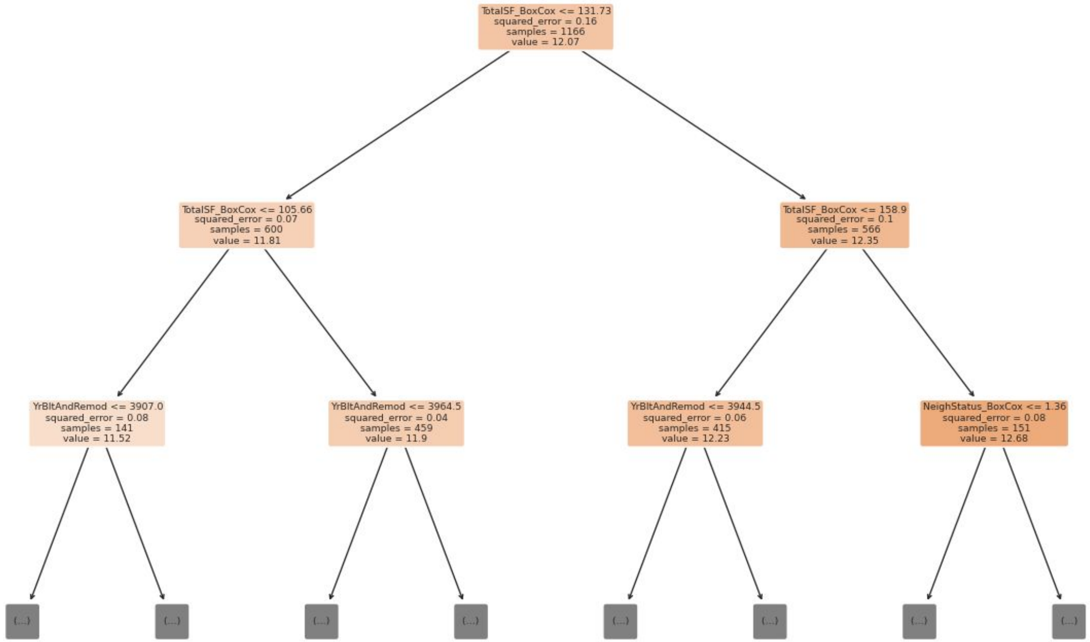
- Out of all models in tfdf.keras:
    - **tensorflow_decision_forests.keras.RandomForestModel <- chose this**
    - tensorflow_decision_forests.keras.GradientBoostedTreesModel,
    - tensorflow_decision_forests.keras.CartModel,
    - tensorflow_decision_forests.keras.DistributedGradientBoostedTreesModel
- RF RMSE train: 0.13
- RF RMSE valid: 0.17
- Slightly better than linear regression

Decision Tree

Training and Validation:
- Train test split: 80% training, 20% validation
- Set random_state = 0

Model Configuration:
- DecisionTreeRegressor
- max_dept = 6
- random_state = 0
- min_sample_split = 2
- min_sample_leaf = 3

Model Evaluation:
- RMSE on training set: 0.1488
- RMSE on validation set: 0.1682



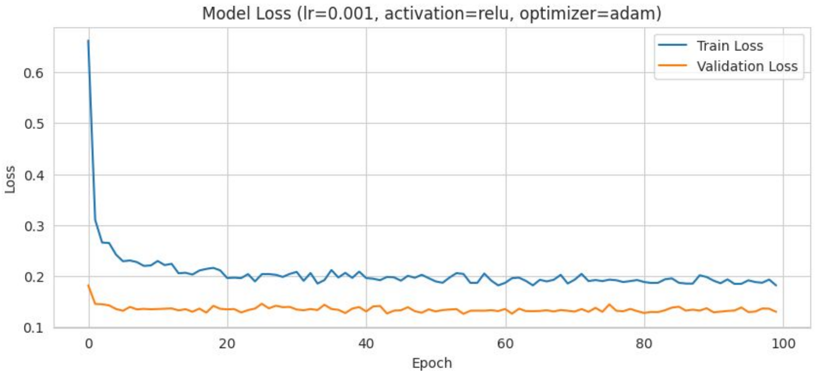
Neural Network

Neural Network Architecture:

- 2 hidden layers with 100 and 50 neurons
- Output for regression
- Included dropout layers to prevent overfitting

Ideal hyperparameter:

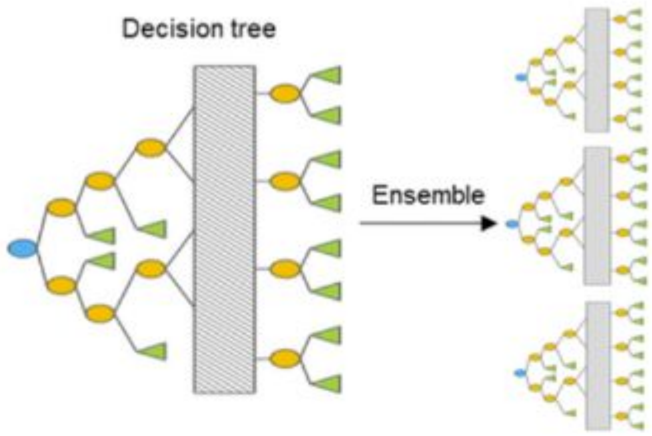
- Activation function: relu
- Optimizer: adam
- Learning Rate: 0.001



	Learning Rate	Activation Function	Optimizer	Training RMSE	Validation RMSE
8	0.0010	relu	adam	0.391484	0.357974
1	0.1000	relu	sgd	0.394308	0.362715
12	0.0001	relu	adam	0.407867	0.367368
4	0.0100	relu	adam	0.404756	0.367816
5	0.0100	relu	sgd	0.407198	0.368180
9	0.0010	relu	sgd	0.412553	0.373899
3	0.1000	tanh	sgd	0.414387	0.383733
10	0.0010	tanh	adam	0.418095	0.384898
14	0.0001	tanh	adam	0.416582	0.385429
7	0.0100	tanh	sgd	0.419325	0.389127
11	0.0010	tanh	sgd	0.422187	0.391757
15	0.0001	tanh	sgd	0.426336	0.395188
6	0.0100	tanh	adam	0.438192	0.413691
2	0.1000	tanh	adam	0.524629	0.503562
13	0.0001	relu	sgd	0.592577	0.579276
0	0.1000	relu	adam	0.739043	0.759518

XGBoost (Extreme Gradient Boosting)

- **Supervised learning**: Builds an ensemble of decision trees. Each new tree corrects the previous tree's error (MSE)
- **Regularization**: To prevent overfitting, add penalty term for the complexity of the model.
- **Tree Pruning**: Stops creating new nodes in individual decisions trees when the loss reduction falls below a threshold



XGBoost (Extreme Gradient Boosting)

Model Configuration :

- max_dept = 1
- Learning rate= 0.1
- Number of estimators = 300

Model Evaluation

- RMSE on training set: 0.17
- RMSE on validation set: 0.20

Conclusion

RMSE of each model

Models	Train	Valid
Baseline	0.40	
Linear Regression	0.17	0.16
Random Forest	0.13	0.17
Decision Tree	0.15	0.17
Neural Network	0.39	0.35
XGBoost	0.17	0.20

With more time, we'd like to also explore: (1) Ensembles, and (2) Unsupervised learning with the dataset without SalePrice

Thank You

Baseline Presentation:

Predicting Home Prices

Brandon Law, Kimsean Pen, Addy Kim

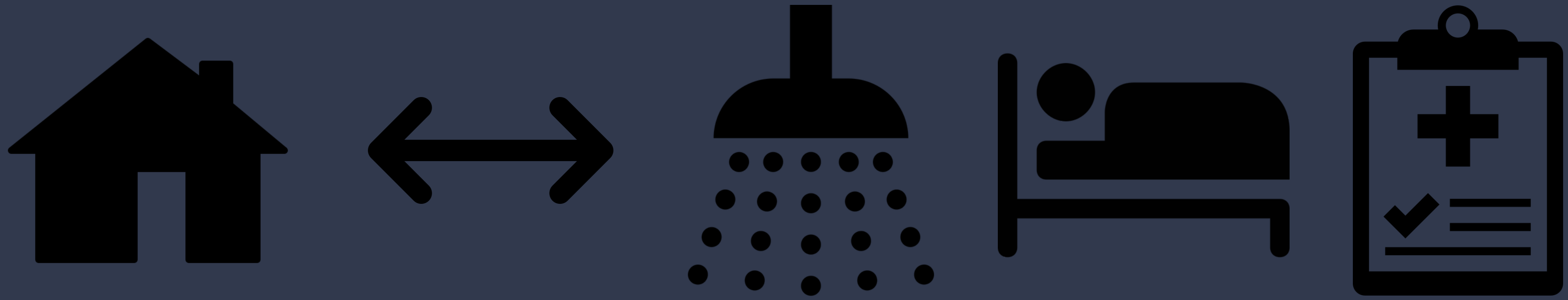
Spring 2024

Agenda

1. Why are we tackling this problem?
2. Dataset
3. Exploratory data analysis
4. Features
5. Next steps

Predicting home prices can be challenging

A home's features and its external factors all play a role in its price



Dataset

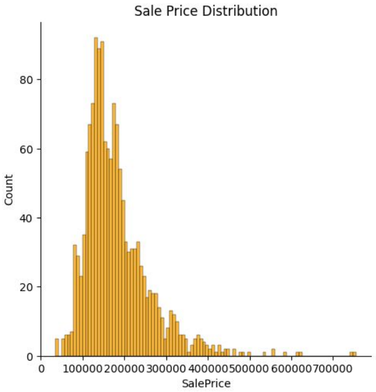
Dataset: Kaggle House Prices

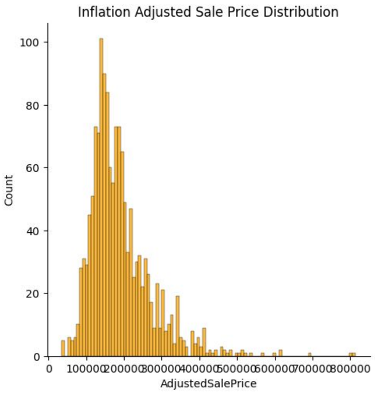
- ~3000 observations across test + train
- 80 possible features
- Y = sales price across 2006 - 2010, time series but adjust all the price to be 2010 equivalent

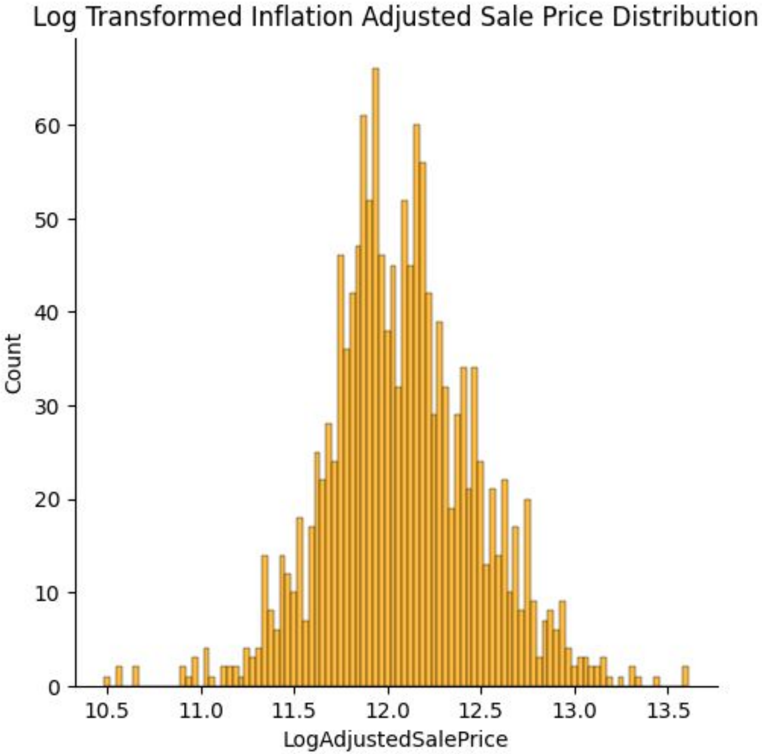
We chose this dataset because of its inherently interesting problem space and because the data was clean. We wanted to focus our project more on **data engineering** applying different methods of **machine learning**.

Exploratory data analysis

- Distribution of features vary depending on numerical vs. categorical: 37 numerical, 43 non-numerical
- Features with the most null values: PoolQC, Fence, MiscFeature, Alley
- 2009 has the highest sales count
- Because this dataset includes sales price from 2006 - 2010, we created a new column “AdjustedSalesPrice” and increased [2.5% every year].
- AdjustedSalesPrice was quite skewed, so we applied a log transformation

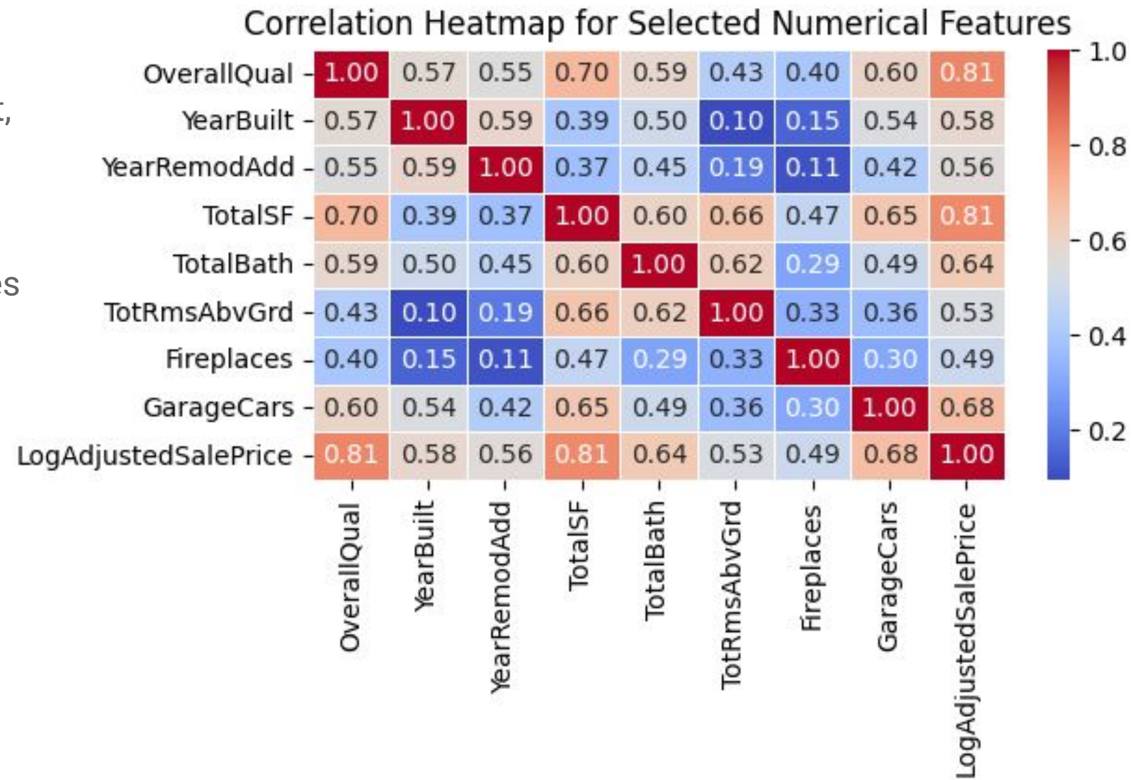






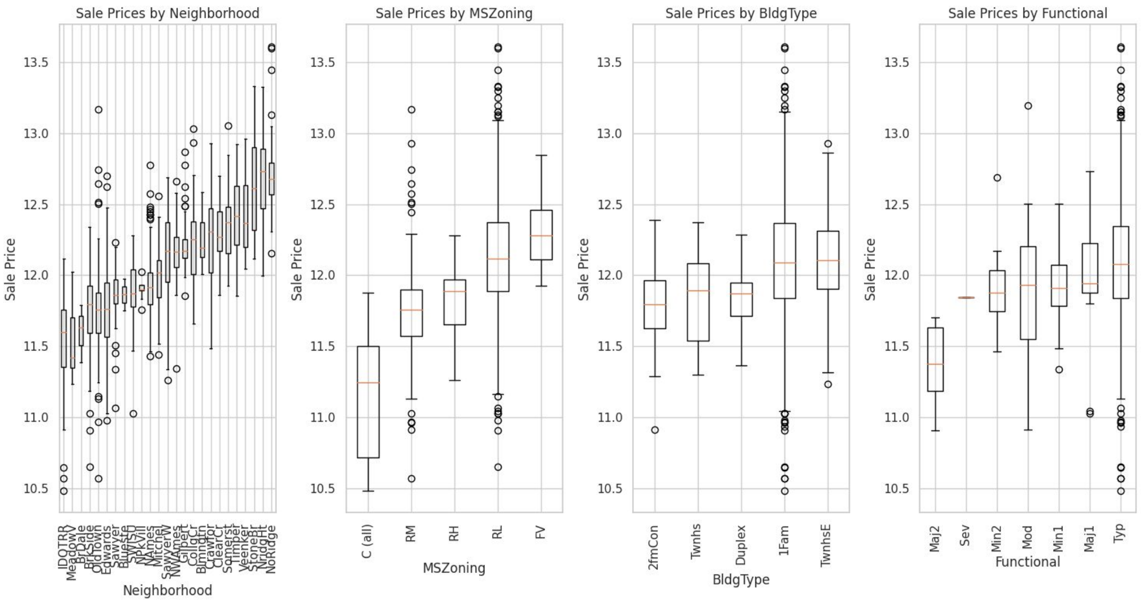
Numerical features

- We created new features combining some existing features, e.g., total sqft, total bath
- Out of the numerical features, Overall Quality, Total SqFt, and Garage Cars have the highest correlation with Sales Price
- Numerical features to include in ML:
    - 'OverallQual'
    - 'YearBuilt'
    - 'YearRemodAdd'
    - 'TotalSF'
    - 'TotalBath'
    - 'TotRmsAbvGrd'
    - 'Fireplaces'
    - 'GarageCars'



Categorical features

- We are in the process of identifying appropriate features using one-hot encoding, p-values, and domain knowledge
- Potential categorical features to include in ML:
  - 'MSZoning’
  - ‘Functional’
  - ‘BldgType’
  - [Neighborhood has too many outliers so won’t be included]



Next steps

1. Categorical feature finalization
2. Feature engineering completion
3. ML methods
    - Linear regression - simple
    - Decision trees - for non-linear relationships
    - Random forest - ensemble combining multiple decision trees
    - Neural networks - deep learning for nuanced patterns
    - K-nearest neighbor - prediction based on the majority class
    - Ensembles - combination of these
4. If time allows, the test dataset doesn’t have the outcome variable, so we may try some unsupervised learning