COMS-W4995 Applied Machine Learning Project Deliverable

Iowa House Price Prediction

By Ananya Gandhi, YeongWoo (Janie) Kim, Austin Schaefer, Heather Song

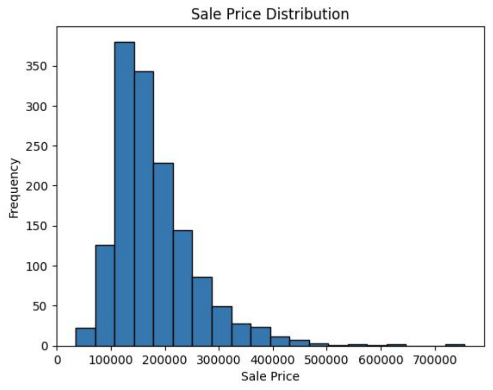
Initial Data Exploration

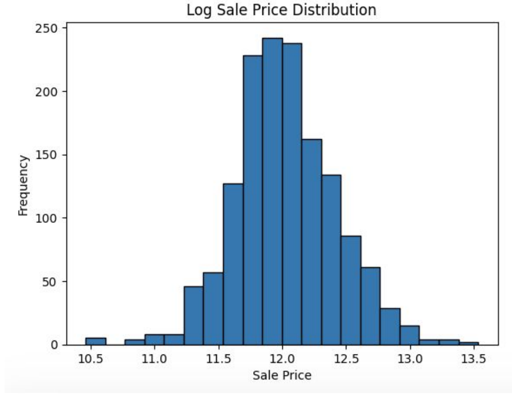
Plotted distribution of target variable (sale price). Used log of target variable to make plots more interpretable.

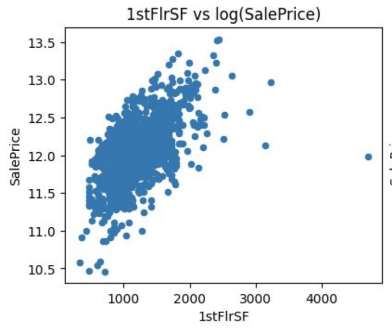
Examined feature vs target relationship with scatter plots and boxplots.

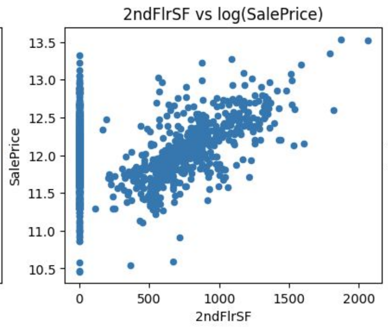
Histograms were mostly ineffective for numerical features due to the high volume of 0s.

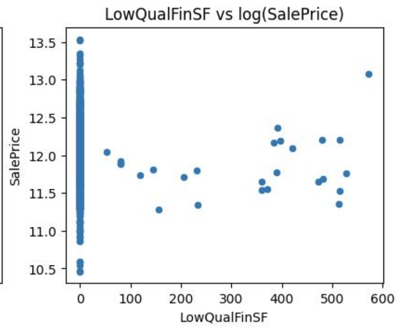
Dropped columns with no correlation or very little information (For example, 99.9% of Utilities data falls under one category)

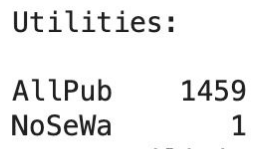




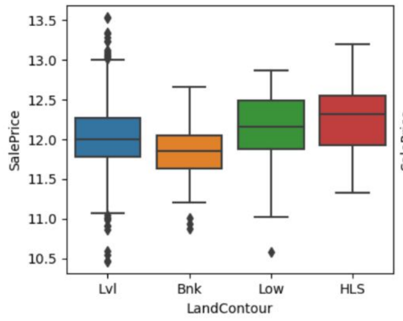


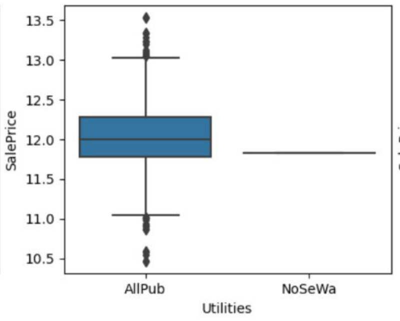


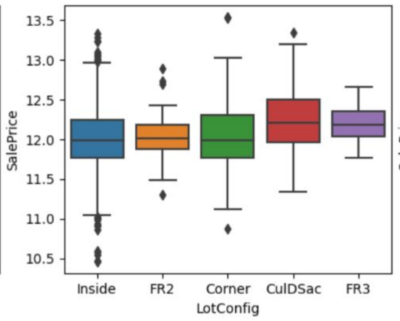


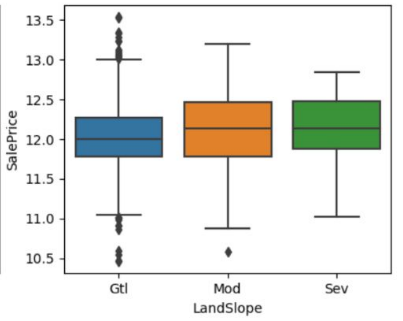

```
Utilities:

AllPub    1459
NoSeWa       1
```





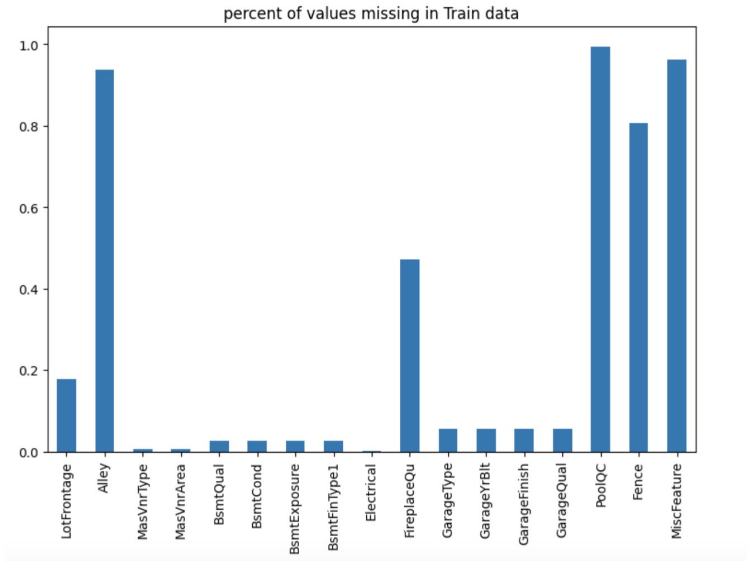


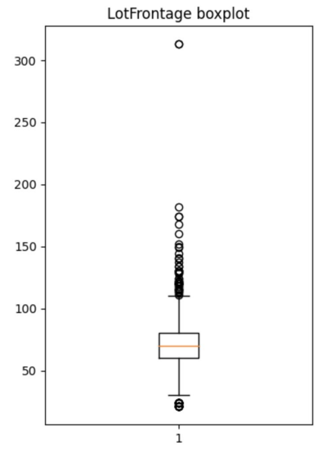


Missing Value Imputation

- For most categorical features, creating a “Missing” label, indicating that property is missing the feature, makes sense because those values are missing not at random (MNAR).
- Features with a few missing values
  - We simply dropped the rows.
- Features with a lot of missing values
  - Features such as “LotFrontage”.
  - Used median imputation after examining its distribution.
  - Mean is heavily influenced by outliers.

```
percentage of missing values:
                  train      test
feature
LotFrontage    0.177397  0.002742
MasVnrArea     0.005479  0.000000
Electrical     0.000685  0.000685
GarageYrBlt    0.055479  0.000685
```

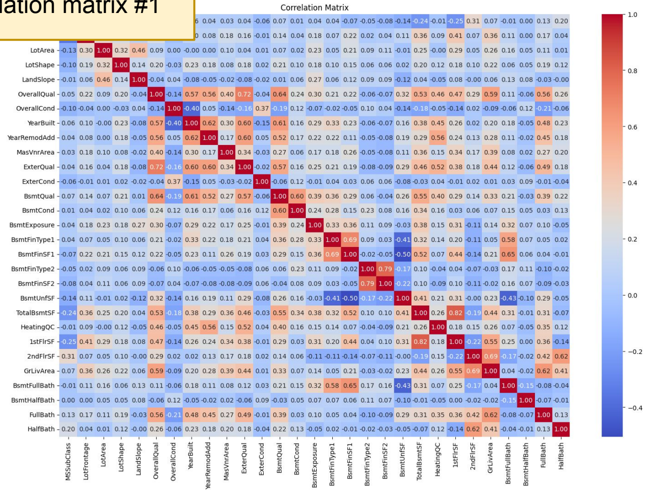




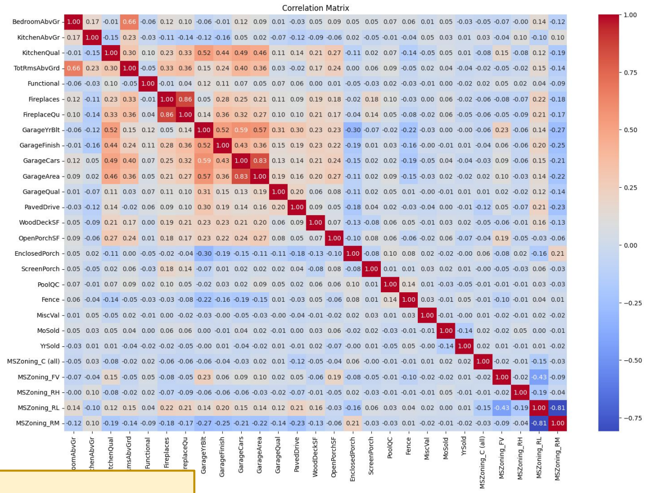
Cleaning and Feature Engineering

- Categorical Variable Encoding
  - Applied target encoding for high cardinality columns, such as “Neighborhood” and “Exterior1st”.
  - Applied ordinal encoding for various categorical columns, such as "LotShape," "ExterQual," and "BsmtQual." For missing values marked as "NA," we encoded them with the value -1.
  - Applied one-hot encoding for non-ordinal columns.
  - Applied label encoding to a specified list of ordinal columns, including "OverallQual" and "OverallCond."
- Data Splitting
  - Divided the dataset into features (stored in the 'features' variable) and the target variable (in 'target') by dropping the "SalePrice" column.

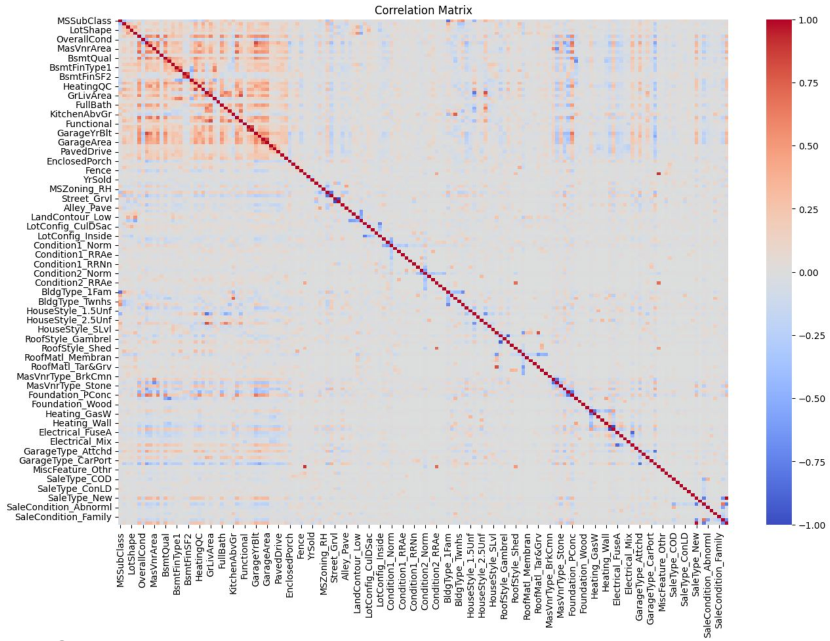
Partial correlation matrix #1



Partial correlation matrix #2



Full correlation matrix



- Correlation Matrices
  - Generated two partial correlation heatmaps for visualization purposes. The first heatmap highlights features associated with property characteristics, while the second focuses on features linked to the property's condition and location.
  - Constructed a comprehensive correlation heatmap to visualize inter-feature relationships across the entire dataset.
  - Identified highly correlated features based on a predefined threshold of 0.85.

Cleaning and Sampling

- Data Splitting for Model Training
  - Split the data into training, validation, and test sets, following a 60/20/20 distribution.

```
X_train, X_temp, y_train, y_temp = train_test_split(features, y, test_size=0.4, random_state=42) # 60/20/20
X_valid, X_test, y_valid, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)
```

- Feature Scaling
  - Applied standard scaling using the 'StandardScaler' to ensure that all features have a mean of 0 and a standard deviation of 1.
- Bias Term Inclusion
  - Added a column of ones to the feature matrices to account for the bias term in linear regression models.

Model Implementation

In order to accurately forecast the final selling price of each house, we decided to implement the following 6 ML models, and each model has its own pros and cons.

01	Linear Regression Model	• Simple baseline model • Quick to train and predict • Can overfit with many features • Assumes linear relationships
02	Ridge Regression (L2 Regularization)	• Reduces overfitting through L2 regularization • Handles multicollinearity well • Biased estimates with high regularization • Hyperparameter tuning needed
03	Lasso Regression (L1 Regularization)	• Reduces overfitting through L2 regularization • Performs feature selection • Can underperform with complex relationships • May exclude useful features with high levels of regularization
04	Elastic Net Regression	• Balances between Ridge and Lasso • Robust to various data scenarios • Computationally more intensive • Can be overkill for simple problems
05	XGBoost	• High predictive power (often performs well in various Kaggle Competitions) • Supports GPU training, sparse data & missing values • Complex, many hyperparameters • Does not support categorical variables natively
06	CatBoost	• Optimized for categorical features • Supports GPU training, sparse data & missing values • Still requires parameter tuning • May be excessive for simple datasets

Model Performance and Comparison

After implementing all 6 models, we used Root Mean Squared Error (RMSE) and R-Squared to evaluate the regression models.
- RMSE measures the average magnitude of the errors between the predicted and actual values without considering their direction. However, RMSE alone does not provide any insights into the proportion of variance explained by the model, which is where R-squared comes in. R-squared represents the proportion of variance in the dependent variable that is predictable from the independent variables. It gives a sense of the quality of the model in terms of its explained variance.

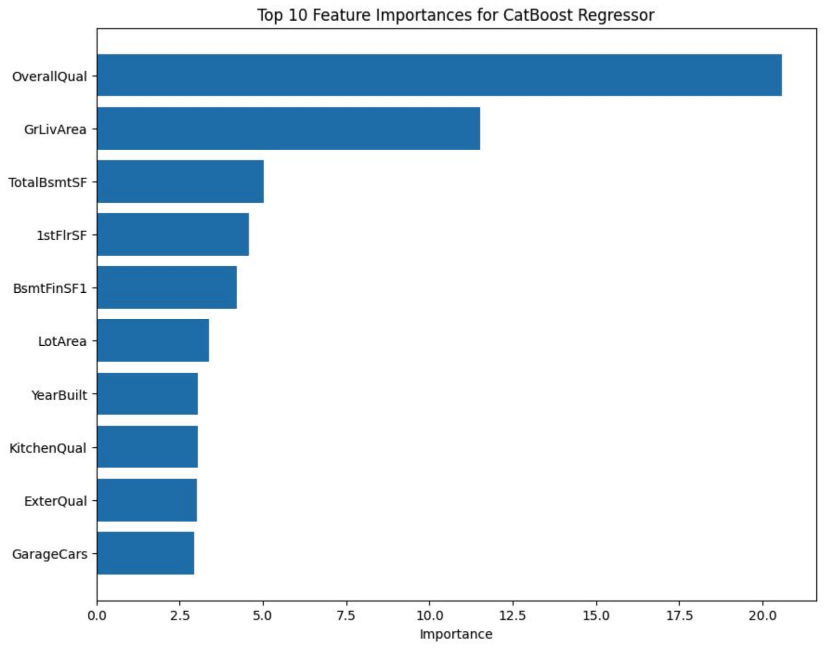
- Using both RMSE and R-squared can provide a more comprehensive understanding of the model performances. RMSE can supplement the limitations of R² by giving us the average error in the units of the measured variable, and R² can show us the proportion of the outcome's variance that is explained by the model.

Model Performances	**Root Mean Squared Error**	**R-Squared**
Linear Regression	3370845801568978.0	1.6732242479511812e+21
Ridge Regression	53029.161	0.586
Lasso Regression	53276.28	0.582
Elastic Net Regression	47666.179	0.665
XGBoost **(with Hyperparameter tuning)**	29828.033	0.869
CatBoost **(with Hyperparameter tuning)**	28609.647	0.879

As shown on the table, **CatBoost** has the best performance. The boosting algorithms (XGBoost and CatBoost) are outperforming the linear models by a considerable margin on this dataset. The poor performance of the Linear models could be multicollinearity or non-linear relationships that it cannot capture. Ridge and Lasso provide improvements through regularization, but the boosting models, which are more complex and can model non-linear relationships, offer the best performance in this case.

Feature Importance and Conclusion

Since CatBoost has the best performance, we decided to apply CatBoost to draw further insights from the dataset.



Based on the feature importance output from the CatBoost model, we can draw several insights:

Quality Over Quantity: While the size of the living areas is important, the overall quality of construction and finishes seems to have a more significant effect on sale prices.
Functional Spaces: Functional spaces like basements and garages are valued, indicating that buyers are looking for practical and usable space.
Modern and Updated: Newer homes and those with modern amenities like updated kitchens tend to fetch higher prices, which could be due to less immediate maintenance required or a preference for contemporary designs.
Investment Areas: For those looking to sell or improve their property value, focusing on improving overall quality, expanding or finishing living areas, and updating kitchens could potentially offer the best return on investment.

These findings are crucial for sellers, buyers, and real estate professionals as they provide a data-driven foundation for understanding what features are likely to drive home prices in this particular market or dataset.