Feature selection methods

GitHub

Andrei Beliaev
beliaev2@illinois.edu

Yuzhou Fu
yuzhouf2@illinois.edu

Introduction

Variable selection is essential in modern data analysis and data science. Identifying influential and informative features not only simplifies models by reducing the number of variables but also provides valuable insights into the data. Techniques such as Principal Component Analysis are commonly used for dimensionality reduction, often eliminating variables during the preprocessing stage. Alternatively, incorporating the selection procedure directly into the model fitting process via regularization constraints can offer significant advantages.

Lasso is now one of the most widely used approaches to regularized regression. Due to the nature of the L_1-penalty, some of the estimates are shrunk exactly to zero during the process, leading to the ability of the lasso to do both continuous shrinkage and automatic variable selection simultaneously [1]. While Lasso has become a leading technique in regularized regression due to its practical effectiveness, it has some certain limitations. For example, when a group of pairwise high collinear variables is presented in a dataset, important information loss may occur, since Lasso tends to select only one variable from the group. A new method the elastic net is therefore proposed by the authors of [1] later in order to perform as well as the lasso whenever it performs the best and improve the limitations of the lasso.

Apart from common regularization methods, boosting algorithms [2] offer another approach to regression modeling by sequentially combining multiple base learners. The concept of boosting is originally rooted in the area of machine learning, and has been successfully adapted to the statistical regression models. Unlike traditional boosting methods, XGBoost (Extreme gradient boosting) introduces the explicit regularization terms in the target function, making it a hybrid between the boosting and regularization-based methods. With each subsequent iteration of the boosting algorithm, the model is updated solely using the contribution of the most informative predictor.

This project explores various variables selection methods; specifically, we focus on correlations, mutual information, SHAP (SHapley Additive exPlanations), Lasso, Elastic Net, and XGBoost to interpret and validate the importance of selected features. By applying these methods to a dataset of housing sale prices featuring 80 variables including 43 categorical features, we aim to discuss the underlying principles for these methods and their performance on the real-world data. The analysis provides insights into the advantages and limitations of each method in addressing different types of data and relationships, such as non-linearity, collinearity, and interactions.

1. Methods

1.1. Mutual information

The *entropy* $H(X)$ of a random variable X, having p as probability density function, is a common measure of uncertainty:

$$H(X) := \mathbb{E}_X[p(X)] = -\int p(x)\log p(x)\,dx.$$

Given two distributions p and q, we define the *Kullback-Leibler (KL) divergence* as:

$$D_{KL}(p \parallel q) := \mathbb{E}_X\left[\frac{p(X)}{q(X)}\right] = \int p(x)\log\frac{p(x)}{q(x)}\,dx.$$

The *mutual information (MI)* between two random variables X and Y is defined as:

$$\begin{aligned} I(X;Y) &:= H(Y) - H(Y|X) \\ &= \mathbb{E}_X[D_{KL}(p(Y|X) \parallel p(Y))] \\ &= \int\int p(x,y)\log\frac{p(x,y)}{p(x)p(y)}\,dx\,dy, \end{aligned}$$

This can be interpreted as the reduction in the uncertainty of Y due to the knowledge of X. Intuitively, it is the amount of information that one variable provides about another one. We note that the mutual information is symmetric (i.e., $I(X;Y) = I(Y;X)$), and is zero if and only if the variables are statistically independent. In contrast to correlation, MI captures non-linear relationships between variables, and thus can act as a measure of true dependence.[3]

For continuous X and Y, the joint and marginal distributions $p(x, y), p(x), p(y)$ cannot be directly computed. Instead, approximation methods like k-nearest neighbors (k-NN) or kernel density estimation (KDE)

1.2. SHAP (SHapley Additive exPlanations)

Understanding why a model makes a certain prediction can be crucial in many applications. However, for large modern datasets, the highest accuracy is achieved by complex models such as deep neural networks or ensemles that are hard to interpret. The authors of [4] propose SHAP as a unified framework for interpreting predictions of complex models, which assigns each feature an importance value for a particular prediction. This is a desirable feature that allows to not only understand the importance of features, but also the importance of particular values of the features, which can be useful in many applications.

There are three classical methods that use cooperative game theory to explain model predictions: **Shapley regression values**, **Shapley sampling values**, and **Quantitative Input Influence**.

Shapley regression values is used for linear models, especially when multicollinearity is present. By retraining the model on all feature subsets $S \subseteq F$, where F represents the complete set of features, and evaluating its prediction performance, it assigns the importance values to each feature. Specifically, the effect of including a feature is determined by training a model $f_{S\cup\{i\}}$ that includes the feature and comparing it to a model f_S where the feature is excluded. For a given input x, the difference in predictions is calculated as:

$$f_{S\cup\{i\}}(x_{S\cup\{i\}}) - f_S(x_S),$$

where x_S represents the feature values in subset S.

The shapley values are calculated as the weighted average of all possible differences across subsets $S \subseteq F \setminus \{i\}$:

$$\phi_i = \sum_{S\subseteq F\setminus\{i\}} \frac{|S|!(|F|-|S|-1)!}{|F|!} \left[f_{S\cup\{i\}}(x_{S\cup\{i\}}) - f_S(x_S)\right], \tag{1}$$

where:

- F: The set of all features.
- S: A subset of features excluding i, i.e., $S \subseteq F \setminus \{i\}$.
- $f_S(x_S)$: The model's prediction using only the features in S, with the input restricted to the feature subset S.
- $f_{S\cup\{i\}}(x_{S\cup\{i\}}) - f_S(x_S)$: The marginal contribution of feature i to the subset S, representing the change in the model's prediction when i is added to S.
- $\frac{|S|!(|F|-|S|-1)!}{|F|!}$: The Shapley weighting term, which accounts for all possible feature orders.

The SHAP value ϕ_i represents the contribution of feature i to the model's prediction for a specific input x. It is computed as the weighted average of the marginal contributions of i across all possible subsets $S \subseteq F \setminus \{i\}$.

Shapley sampling values provide an alternative by approximating the Shapley values without requiring retraining. This method uses:

1. Sampling approximations for Equation (1).
2. Estimating the effect of removing a variable by integrating over training data samples.

This approach reduces computational costs by eliminating the need to retrain the model for all subsets and requires fewer than $2^{|F|}$ calculations. Since the Shapley sampling values share the same form as Shapley regression values, they also qualify as an additive feature attribution method.

Quantitative input influence is a broader framework that can address not only feature.

By unifying simple properties for additive feature attribution methods, the authors propose SHAP values, which satisfy Local accuracy, Missingness, and Consistency properties and are the solution to the following equation:

$$\phi_i(f, x) = \sum_{z'\subseteq x'} \frac{|z'|!(M-|z'|-1)!}{M!} \left[f_x(z') - f_x(z' \setminus i)\right]$$

Since the exact computation of the exact SHAP values may be challenging, the authors also provide approximations using assumptions on expectation, independence, and model linearity.[4]

1.3. Lasso

The lasso estimates solve the optimization problem

$$\begin{aligned} &\arg\min_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 \right\} \\ &\text{subject to} \quad \sum_{j=1}^{p} |\beta_j| \le t. \end{aligned}$$

This optimization problem requires that the L_1-norm of the coefficient estimates remain below a prespecified threshold t, which ensures that the resulting solutions have small absolute values. The predictors x_j are usually standardized (i.e., $\sum_{i=1}^{N} x_{ij} = 0$ and $\sum_{i=1}^{N} x_{ij}^2/N = 1$) to ensure equal contributions of each unknown parameter to the L_1-penalty.

Alternatively, the lasso problem can be rewritten to the equivalent Lagrangian form

$$\arg\min_{\boldsymbol{\beta}\in\mathbb{R}^P} \left\{ \|\mathbf{y} - X\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1 \right\}$$

The L_1-penalty is now directly included in the minimization problem. Although inheriting the role of regularization from the explicit L_1-threshold, the new tuning parameter λ

is not identical to t, a one-to-one correspondence between t and λ is assured by Lagrangian duality. Moreover, it can be observed that as λ increases, the penalty becomes stricter, leading to a greater shrinkage and fewer non-zero coefficients.

1.4. Elastic Net

The naïve elastic net estimates solve the optimization problem

$$\arg\min_{\boldsymbol{\beta}\in\mathbb{R}^P} \left\{ \|\mathbf{y} - X\boldsymbol{\beta}\|_2^2 + \lambda_2\|\boldsymbol{\beta}\|_2^2 + \lambda_1\|\boldsymbol{\beta}\|_1 \right\}$$

or equivalent optimization problem form

$$\begin{aligned} &\arg\min_{\boldsymbol{\beta}\in\mathbb{R}^P} \|\mathbf{y} - X\boldsymbol{\beta}\|_2^2 \\ &\text{subject to} \quad (1-\alpha)\|\boldsymbol{\beta}\|_1 + \alpha\|\boldsymbol{\beta}\|_2^2 \le t, \\ &\text{where} \quad \alpha = \frac{\lambda_2}{\lambda_1 + \lambda_2}. \end{aligned}$$

Here, the elastic net penalty: $(1-\alpha)\|\boldsymbol{\beta}\|_1 + \alpha\|\boldsymbol{\beta}\|_2^2$ is a convex combination of the lasso and ridge penalty. When $\alpha = 1$, the elastic net penalty function turns into the simple ridge regression, while $\alpha = 0$, it becomes the lasso regression. Therefore, the predictors x_j are usually standardized to ensure equal contributions of each unknown parameter to the L_1-penalty.

Elastic net improves upon the naïve elastic net by applying a rescaling adjustment $(1 + \lambda_2)$ to address the issue of double shrinkage[1]. While the naïve elastic net combines the lasso and ridge penalties, the elastic net modifies the coefficients to reduce unnecessary bias and improve predictive performance. And unlike the lasso, which tends to select only one variable from a group of highly pairwise correlated predictors, the elastic net exhibits a grouping effect[1]. This means that the elastic net tends to select and shrink coefficients of correlated predictors together, assigning similar importance to them. This property is particularly useful when in the cases variables are highly correlated, since it ensures that valuable information is not excluded by selecting one variable from the group.

1.5. XGBoost (eXtreme Gradient Boosting)

In gradient boosting, the choice of a loss function, $l(y, f(X))$, depends on the properties of the response variable y. Common options include squared error loss or a negative likelihood function[2], which the algorithm seeks to minimize iteratively over multiple steps. The base-learners $h(x_j)$ are generally defined as regression functions of each predictor $h(x_j)$ in X, and these components are combined to construct the additive model $f(X) = \beta_0 + h_1(x_1) + \cdots + h_k(x_k)$, where β_0 is the intercept.

In XGBoost, the objective is defined as the overall loss with the additional regularized term

$$\mathcal{L}(\phi) = l(y, f(X)) + \sum_k \Omega(f_k),$$
$$\text{where} \quad \Omega(f) = \gamma T + \frac{1}{2}\lambda\|w\|^2$$

Each regression tree f_k is the base learner added after an iteration, it corresponds to an independent tree structure and contains continuous scores w_i on i-th leaf. Predictions can be made by summing up the score in the corresponding leaves.

At the t-th iteration, let $\hat{y}_i^{(t)}$ be the prediction of the i-th instance, the objective

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i)\right) + \Omega(f_t)$$

can be simplified by applying second-order approximation and expanding Ω [5]

$$\tilde{\mathcal{L}}^{(t)} = \sum_{j=1}^{T}\left[\left(\sum_{i\in I_j} g_i\right) w_j + \frac{1}{2}\left(\sum_{i\in I_j} h_i + \lambda\right) w_j^2\right] + \gamma T$$

where

$$g_i = \frac{\partial}{\partial \hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)}) \quad \text{and} \quad h_i = \frac{\partial^2}{\partial \hat{y}_i^{(t-1)2}} l(y_i, \hat{y}_i^{(t-1)}),$$

I_j : instance (observation) set of leaf j,

T : number of leaves in the tree.

then for a tree structure, we can compute the optimal weight w_j^* of leaf j by [5]

$$w_j^* = -\frac{\sum_{i\in I_j} g_i}{\sum_{i\in I_j} h_i + \lambda},$$

It is generally impractical to enumerate all possible tree structures. A greedy algorithm can be applied, which starts from a single leaf and incrementally adds branches to form the tree. Assume that I_L and I_R are the instance sets of left and right nodes after the split. Letting $I = I_L \cup I_R$, then the loss reduction after the split is given by [5]

$$\mathcal{L}_{\text{split}} = \frac{1}{2}\left[\frac{\left(\sum_{i\in I_L} g_i\right)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{\left(\sum_{i\in I_R} g_i\right)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{\left(\sum_{i\in I} g_i\right)^2}{\sum_{i\in I} h_i + \lambda}\right] - \gamma$$

This function calculates the gain, which measures the reduction in loss after splitting a node. It is commonly used to evaluate split candidates and find the optimal split point. There are several split finding algorithms existing for balancing the computational efficiency and optimal split selection. Some of them involve computing $\mathcal{L}_{\text{split}}$, while others use approximate methods to reduce computational cost. The details are omitted here for brevity.

2. Experiments and Results

The dataset focuses on housing sale prices and consists of 80 features, categorized into 35 numerical, 14 ordinal, and 43 categorical features in total. To ensure data quality and reliability, outliers were removed, and missing values were addressed by filling them with the median for numerical features and the mode for categorical features. This preprocessing approach helps standardize the dataset, reducing noise and ensuring that the analysis and modeling are based on complete and clean data, ultimately supporting more accurate predictive modeling and insights.

2.1. Correlations

First, we identified the most important features in the dataset by leveraging various following techniques. For numerical features, correlations were calculated to determine their relationship with the target variable, providing insights into how strongly each numerical attribute influences housing sale prices (Figure 1, Table 1).

For ordinal features, group means were analyzed, and their correlations are calculated, which helps to quantify the impact of ordinal variables. The top correlated ordinal features are presented in Table 2. These methods collectively allow for a deeper understanding of the dataset, enabling to understand the relevant features that contribute to the target variable, and what features to ordinary encode.

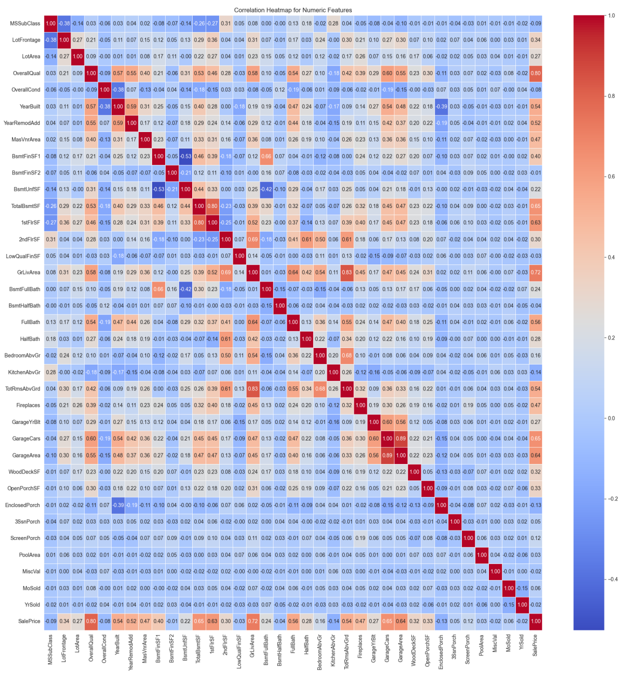

Figure 1. Correlation map, numerical

Feature	Correlation
OverallQual	0.801
GrLivArea	0.721
GarageCars	0.649
TotalBsmtSF	0.647
GarageArea	0.637
1stFlrSF	0.625
FullBath	0.559
TotRmsAbvGrd	0.537
YearBuilt	0.535
YearRemodAdd	0.521

Table 1. Correlation of top 10 numerical features with SalePrice

Feature	Correlation
ExterQual	0.695
KitchenQual	0.666
FireplaceQu	0.517
BsmtQual	0.514
HeatingQC	0.435
BsmtExposure	0.377
GarageQual	0.270
GarageCond	0.263
PavedDrive	0.237
BsmtCond	0.204

Table 2. Correlation of top 10 ordinal features with SalePrice

2.2. Boosting Regression

We chose the boosting regressor model as our baseline for feature selection because it inherently identifies important features during training through its tree-building process and often handles feature selection effectively on its own. We opted not to implement complex recursive feature elimination with boosting and nested cross-validation to find important features, as this approach is impractical in real-world scenarios. Additionally, it would unnecessarily complicate an already robust baseline provided by the boosting model.

The feature importance derived from the boosting regressor is presented in Table 3. The feature scores were derived on the one-hot-encoded (OHE) dataset consisting of 228 features, combining numerical, ordinal encoded, and one hot encoded features. To ensure robustness, we used cross-validation to optimize the model and averaged feature importance based on gain across folds.

2.3. Mutual Information Regression

The mutual information regression method was applied to a dataset with OHE categorical features. The method produced consistent estimates regardless of whether standard scaling was applied or different values of K (in K-Nearest

Feature	SHAP Value
OverallQual	0.190
GarageCars	0.122
ExterQual	0.106
BsmtQual	0.035
GrLivArea	0.034
FireplaceQu	0.027
KitchenQual	0.027
FullBath	0.025
MSZoning_RM	0.024
TotalBsmtSF	0.020

Table 3. Boosting regressor top 10 features

Neighbors) were used.

The method provided reliable and reasonable estimates of the importance of the features, focusing primarily on numerical features. However, a the notable limitation is its inability to handle collinearity among features, which could affect its robustness when dealing with highly correlated variables.

This is a good baseline that offers a valuable tool for feature selection, especially in scenarios where numerical features dominate.

Feature	Normalized Mutual Information
OverallQual	0.054
GrLivArea	0.045
TotalBsmtSF	0.036
GarageCars	0.035
YearBuilt	0.034
GarageArea	0.034
ExterQual	0.032
KitchenQual	0.031
BsmtQual	0.031
1stFlrSF	0.031

Table 4. Top 5 Normalized Mutual Information selected features.

2.4. SHAP

The top 10% of features selected by SHAP are presented in Figure 2, arranged by their importance. It is evident that numerical and ordinal features dominate this selection, with the notable exceptions of SaleCondition and Neighborhood.

It's also interesting to compare how MI and SHAP handle interactions, non-linearity, and collinearity:

- **Interactions:**
  - **SHAP:** Evaluates all subsets of features, capturing interactions explicitly.
  - **Mutual Information:** Evaluates features independently. Interactions are missed unless conditional mutual information is used.

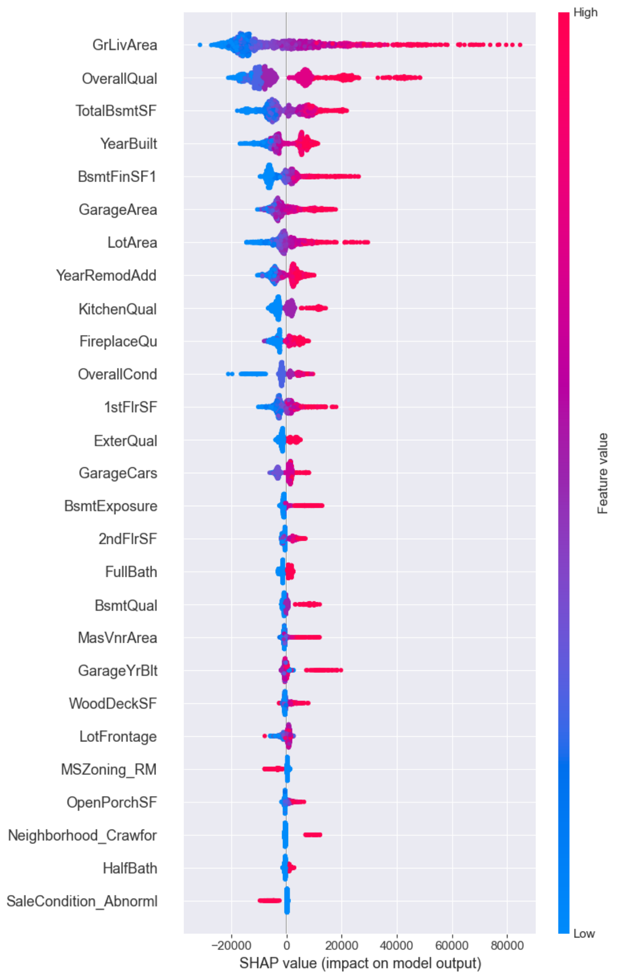

Figure 2. Top 10% SHAP values

- **Non-Linearity:**
  - **SHAP:** Captures non-linear relationships directly from the model's predictions, as learned by complex models like trees or neural networks.
  - **Mutual Information:** Detects non-linear dependencies using joint probability distributions but doesn't reveal the direction or shape.
- **Collinearity:**
  - **SHAP:** Divides contributions fairly among correlated features based on marginal contributions across subsets, ensuring fair attribution.
  - **Mutual Information:** Evaluates features independently, leading to inflated importance for collinear features. Redundancy is not accounted for.

2.5. Lasso and Elastic Net

We applied both Lasso and Elastic Net to this dataset. Notably, Elastic Net behaves almost identically to Lasso, because the chosen coefficient corresponding to the L1 penalty is 0.9. The feature importance values were calculated by averaging the coefficients across folds, ensuring a robust estimate of their contribution. Interestingly, the selected regularization parameter λ is high (1000) for both label encoding and one-hot encoding (OHE) approaches, reflecting the model's preference for stronger regularization.

The resulting top 10 features and their importance values are presented in Table 5. Interestingly, Lasso selected multiple categories corresponding to the Neighborhood feature. However, intuitively, this feature might not belong in the top 10, as house prices within the same neighborhood can still vary significantly. This suggests that the model's ability to generalize may be limited in this case.

Feature	**coefficient**
Neighborhood_StoneBr	0.072
GrLivArea	0.057
Neighborhood_NridgHt	0.056
Neighborhood_Crawfor	0.039
Neighborhood_NoRidge	0.037
SaleType_New	0.035
Functional_Typ	0.031
OverallQual	0.029
BldgType_1Fam	0.029
Exterior1st_BrkFace	0.027

Table 5. Lasso feature importance values

2.6. Model performance and comparison of selected features

We also evaluated performance of the models for different encoding of categorical features to find the best models to compare. The results are presented in the Table 6. The best models are highlighted in green.

Model	**Encoding**	**Standard Scaling**	**MAE**	**R^2**
XGBoost	Label	Yes	-	-
XGBoost	Label	No	14499	0.912
XGBoost	OHE	No	14291	0.917
Lasso	Label	No	19272	0.871
Lasso	Label	Yes	18225	0.879
Lasso	OHE	Yes	18055	0.890
Linear Regression	Label	Yes	Not Working	Not Working
Linear Regression	Label	No	18427	0.882
Linear Regression	OHE	yes	Not Working	Not Working
ElasticNet	Label	Yes	19447	0.869
ElasticNet	Label	No	18154	0.882
ElasticNet	OHE	Yes	18055	0.890

Table 6. Performance of Different Models with Encoding and Scaling

It is worth noting that Linear Regression performs on non-scaled features but fails on scaled features, which was surprising since linear models are generally expected to perform better on scaled data. This dataset appears to be too complex, and with One-Hot Encoding (OHE), the linear model could not produce reliable estimates.

As expected, the boosting model delivered the best results on OHE-encoded features. We did not scale them since the model works well with unscaled data. Lasso, on the other hand, performed best when scaling was applied to OHE features.

Based on these results, we selected the two best-performing models for feature importance analysis: Lasso on scaled OHE features and XGBoost on unscaled OHE features. We then evaluated the features selected by each model. The results are presented in Table 7.

Model	**Selected**	**Categorical**	**Numerical**	**OHE**	**Ordinal**
XGB	184	149	35	135	14
Lasso	129	96	33	83	13

Table 7. Feature selection comparison between XGB and Lasso.

XGB selected significantly more OHE features, which highlights its strength in handling categorical variables. Both methods selected similar numbers of numerical and ordinal features, demonstrating comparable handling of these feature types. The correlation of feature importance between the two methods was calculated to be 0.24. XGBoost, being tree-based, likely leverages interactions and non-linear relationships, whereas Lasso relies on additive linear effects.

3. Discussion and Conclusion

In this project, we examined various feature selection methods, including Mutual Information (MI), SHAP, Lasso, ElasticNet, and Gradient Boosting Regression, applied to a housing dataset containing a mix of categorical, numerical, and ordinal variables. Each method showcased unique strengths and limitations, especially in how they handled interactions, non-linearity, and collinearity.

Mutual Information was effective at capturing non-linear dependencies between features and the target variable, providing insights that extended beyond simple correlations. However, it struggled with handling feature interactions and collinearity, often missing redundant or complementary relationships. Despite its simplicity, MI is limited when applied to datasets with complex dependencies.

SHAP proved to be a powerful tool for feature importance analysis, especially for interpreting complex models like Gradient Boosting on trees, as it effectively combines their strengths. Beyond its technical capabilities, SHAP's interpretability was a significant advantage, providing clear insights into how each feature contributed to the model's predictions.

Lasso excelled in enforcing sparsity, making it particularly effective for datasets with many irrelevant features.

However, its reliance on linearity limited its ability to capture more complex relationships. ElasticNet, which aims to balance sparsity and feature grouping, did not perform well in this case, reducing to Lasso on this dataset.

Gradient Boosting Regression emerged as the best-performing method in terms of predictive accuracy. Its ability to handle complex interactions and non-linear relationships demonstrated its superiority as a modeling approach for both feature selection and regression tasks. As expected, its performance showed the effectiveness of tree-based methods for datasets with diverse feature types.

It is important to recognize the feature importance scores. While it might be tempting to assume that these methods can identify which features decision-makers should manipulate to influence future outcomes, this approach can be misleading in causal inference tasks. Predictive models are not always suitable for guiding policy choices, as they often fail to establish causal relationships.

4. Statement of Individual Contributions

- **Andrei Beliaev:** Project idea, experiments, theory for MI, SHAP, project report.

- **Yuzhou Fu:** Theory for Lasso, Elastic Net, and Boosting, helped debugging in data preprocessing, model building of the lasso and elastic net regression, project report.

References

[1] H. Zou and T. Hastie. *Regularization and variable selection via the elastic net*. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005. 1, 3

[2] T. Hepp, M. Schmid, O. Gefeller, E. Waldmann, and A. Mayr. *Approaches to Regularized Regression – A Comparison between Gradient Boosting and the Lasso*. Methods of Information in Medicine, vol. 55, no. 5, pp. 422–430, 2016.

[3] Mario Beraha, Alberto Maria Metelli, Matteo Papini, Andrea Tirinzoni, Marcello Restelli, "Feature Selection via Mutual Information: New Theoretical Insights," 1, 3

[4] Scott Lundberg, Su-In Lee, "A Unified Approach to Interpreting Model Predictions," 1

[5] T. Chen and C. Guestrin. *XGBoost: A Scalable Tree Boosting System*. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16), pp. 785–794, 2016. 2
3

[6] T. Hastie, R. Tibshirani, and M. Wainwright. *Statistical Learning with Sparsity: The Lasso and Generalizations*. CRC Press, 2015.

[7] T. Hothorn. *Boosting – An Unusual Yet Attractive Optimiser*. Methods of Information in Medicine, vol. 53, no. 6, pp. 417–418, 2014.

[8] P. Bühlmann and T. Hothorn. *Boosting Algorithms: Regularization, Prediction and Model Fitting*. Statist. Sci. 22 (4) 477 - 505, November 2007.