**Data preparation, exploration, visualization**

My data cleaning process, same as last week, is as follows:

1. Outlier removal (-30 rows)
2. Removing extra columns, data type change, imputation (-4 columns)
3. Removing high VIF columns (-1 column)
4. Box-cox transformation on high skew features
5. Feature engineering and one-hot encoding
6. RobustScaler to normalize data

**Review research design and modeling methods**

1. OLS and explicit feature elimination

I use OLS as a baseline for evaluating the other models. Explicit feature elimination also produces models with similar performance as OLS. Eliminating 0 to 30 features showed a slight performance improvement; only keeping 25 or fewer features had slightly poorer performance than OLS; and everything in between was extremely bad.

1. Lasso

Lasso updates feature weights as if the cost includes a term with the L1 norm of the weights vector (sum of absolute values of all coefficients). A hyperparameter controls how much penalty the regularization contributes to the cost function. During the optimization, the weights for some features can go all the way down to zero, effectively eliminating features that cause the most variance. I use the LassoCV function to search for the hyperparameter value that produces the best RMSE from 5-fold cross-validation. If I force the model to use positive coefficients, even with a small hyperparameter of 0.0002, more than 90 features are eliminated, but there is a slight improvement in model score.

1. Ridge

Ridge’s regularization uses the squared L2 norm of the coefficients vector to update cost, effectively penalizing features with larger weights. I get a slightly better RMSE than Lasso here.

1. ElasticNet

ElasticNet is a combination of Lasso and Ridge, and should result in smaller weight values overall while reduced the number of features eliminated. Two hyperparameters balance L1 and L2 regularization, but also mean significantly more computation for optimizing for the hyperparameters. As expected, since Ridge performed better than Lasso, the l1\_ratio is closer to 0, representing greater weight on L2 regularization.

1. BayesianRidge and Support vector regression

I also tried Bayesian Ridge and SVR models for comparison. Only minor hyperparameter tuning was done because there is no iterative parameter search like ElasticNetCV, and using GridSearchCV’s exhaustive search takes an extraordinarily long time with 4 parameters for BayesianRidge and 3 for SVR.

1. Model stacking

The final model I ran stacked all the of the above models, with Elastic Net as the meta-regressor.

**Review results, evaluate models**

I had to go back and forth to my data processing steps and realized that scaling played a critical role in how the regression turned out. I compared modeling across 3 schemes. One without any box-cox scaling, one where the lambda of the box-cox transformation of skewed features was computed from only the training data, and one where the lambda was computed from both the training and test sets. All three were normalized with RobustScaler before fitting.

The first table below are a few of the most skewed features that needed transformation along with their respective lambda. A few variables show major differences between the lambdas of the training set and the total set and greatly influence the fitting later.

|  |  |  |
| --- | --- | --- |
| feature | train | alldata |
| PoolArea | 8.47214 | -0.12555 |
| LotArea | 0.07998 | 0.16374 |
| MiscVal | 0.13480 | 0.03260 |
| 3SsnPorch | 0.01574 | 0.07876 |
| KitchenAbvGr | 8.47214 | 0.26082 |
| BsmtFinSF2 | 0.16891 | 0.16165 |
| BsmtHalfBath | 8.47214 | 0.24093 |
| ScreenPorch | 0.47284 | 0.51110 |
| EnclosedPorch | 0.37705 | 0.32784 |
| MasVnrArea | 0.44285 | 0.42993 |
| OpenPorchSF | 0.47285 | 0.45916 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | Without box cox | Box cox on all data | Box cox on train |
| Lasso | alpha | 0.0005166332665330662 | 0.00019919839679358718 | 0.0004967935871743487 |
| Ridge | alpha | 11.745745745745747 | 11.631631631631633 | 12.384769539078157 |
| ElasticNet | l1\_ratio | 0.18728643216080404 | 0.036756756756756756 | 0.10849246231155779 |
| alpha | 0.002311557788944724 | 0.00714308617234469 | 0.002628140703517588 |

The found parameters for the different models are shown in the table below.

**Kaggle results**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Without box cox | | Box cox on all data | | Box cox on train | |
| Model | RMSE | Kaggle score | RMSE | Kaggle score | RMSE | Kaggle score |
| OLS | 0.11761 | 0.15061 | 0.11553 | 0.13436 | 690832 | 0.41615 |
| Lasso | 0.10803 | 0.13395 | 0.10626 | 0.12643 | 0.13369 | 0.14545 |
| Ridge | 0.10851 | 0.14114 | 0.10620 | 0.12490 | 0.10623 | 0.14234 |
| ElasticNet | 0.10760 | 0.13490 | 0.10566 | 0.12469 | 0.10551 | 0.14075 |
| Bayesian Ridge | 0.10845 | 0.14100 | 0.10612 | 0.12491 | 40033 | 0.55548 |
| SVR | 0.15575 | 0.17083 | 0.10738 | 0.13460 | 0.33202 | 0.41714 |
| Stacked | 0.10743 | 0.13607 | 0.10532 | 0.12449 | 0.10820 | 0.14819 |

**Exposition, problem description, and management recommendations**

One reason behind the discrepancy between transforming on all data and transforming only based on the training set is the presence of outliers in the training set. My outlier removal strategy was conservative and when I reduce the outlier thresholds for more samples removed, the box-cox parameters become much closer together. However, this could also be some form of data leak, since I am using the test set to inform me of how far outliers can be.

The best Kaggle score I obtained was from stacking regression. More iterations with parameter tuning will give better scores, as well as weighting the models separately. More model types, such as gradient boosting machines and random forests, will also likely improve the model.