1. **Data preparation, exploration, visualization**

I start by loading the train and test files into Jupyter Notebook as pandas dataframes and cleaning them. Detailed steps in appendix 2.

1. Remove unnecessary columns
2. Data type changes
3. Imputing missing values
   1. Categorical/ordinal features for replacing with None
   2. Numerical features for replacing with 0
   3. Replacing features with few missing values with the most frequent value
   4. Replace ‘LotFrontage’ NaN values with the median of other ‘LotFrontage’ values in its neighborhood
4. Outlier removal
   1. Using a quantile method removing only rows with extreme values vs SalePrice
   2. Will revise with a manual method based on feature vs target plots
5. Removing multicollinear features
   1. Will only remove ‘LowQualFinSF’ since it is not correlated with ‘SalePrice’ and also will not be used in feature engineering
6. Feature scaling
   1. Box-cox scaling on numerical features with |skew|>0.5
   2. log(1+x) scaling on ‘SalePrice’
7. Adding new features
   1. ‘TotalSF’ ‘YrBuiltRemod’ ‘TotalBath’ ‘TotalPorch’
   2. Binary features for ‘PoolArea’, ‘2ndFlrSF’, ‘GarageArea’, ‘TotalBsmtSF’, ‘Fireplaces’
8. Encoding ordinal categorical variables manually
9. Encoding nominal categorical variables through dummy encoding
10. Columns not present in either test or train sets were added to each other so that they have the same number of features.

These modifications were coded as functions and applied to both train and test datasets.

1. **Review research design and modeling methods**

I create a function to evaluate the root mean squared error through 5-fold cross validation of the models created. Each model is passed through a RobustScaler preprocessing step, and then fitted. Predicted values are generated on the test set, and have (expX – 1) applied to invert the log(1+x) on the dependent variables.

1. OLS Linear Regression
   1. LinearRegression fits a linear model with coefficients for each feature to minimize the residual sum of squares between observed targets and predicted targets
   2. No regularization
2. Linear regression with restricted features
   1. I use Recursive Feature Elimination to reduce the number of variables to a limited set that are most relevant in predicting the target variable
   2. hyperparameter tuning on n\_features\_to\_select through GridSearchCV
3. LASSO
   1. Model penalized for sum of absolute values of weights, so many features will be weighted to 0.
   2. L1 regularization will make overfitting less likely.
   3. model complexity decreases with higher alpha as more feature coefficients are set to 0.
   4. will have to do some hyperparameter tuning on alpha through GridSearchCV

I predict that each subsequent model will produce less error and a better Kaggle score when fitted to the test dataset.

1. **Review results, evaluate models**

For RFE, 89 features are found to be the best to select.

For Lasso, an alpha of 0.0007210526315789474 was found to be good.

|  |  |  |
| --- | --- | --- |
| Model | RMSE mean | RMSE std |
| OLS | 0.1182 | 0.0073 |
| RFE | 0.1210 | 0.0172 |
| Lasso | 0.1090 | 0.0046 |

The model with RFE showed worse fitting than OLS, suggesting that some features that were ranked lower still had an impact on the fit. As expected, the Lasso model showed the best fits. If I had more time or computing power, a better alpha value could be found.

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

The research question of interest was to find good prediction of house sale price when provided with its features. The models used in this week’s assignment was able to get a decent score but it seems like it will be difficult to get below 0.10. Stacking multiple regression models via a meta-regressor will probably be necessary.

Comparing my models show me that most of the features influence the target Sale Price in some way, but many of them likely have very small weights.

For management recommendations, obviously a bigger data set will result in better models. When split by dwelling class or neighborhood, there becomes very few samples in each class.

Appendix 2: Data preprocessing

1. Remove columns
   1. 'Id' - meaningless column
   2. 'Utilities' - Only 1 unit does not have all utilities.
   3. 'Street' - only 6 units have gravel access instead of pavement
2. Data type changes
   1. 'MSSubClass' - int64 but housing type is a categorical variable
   2. 'YrSold' - int64 but year is categorical
   3. 'MoSold' - month is also categorical
3. Imputing missing values
   1. Categorical/ordinal features for replacing with None: 'MiscFeature', 'Alley','FireplaceQu', 'GarageFinish', 'GarageQual', 'Fence', 'GarageType', 'GarageQual', 'GarageCond', 'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2', 'MasVnrType', 'MSSubClass'
   2. Numerical features for replacing with 0: 'GarageYrBlt', 'GarageArea', 'GarageCars', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'BsmtFullBath', 'BsmtHalfBath', 'MasVnrArea'
   3. Replacing features with few missing values with the most frequent value: 'MSZoning', 'Electrical', 'KitchenQual', 'Exterior1st', 'Exterior2nd', 'SaleType', 'Functional'
   4. Replace ‘LotFrontage’ NaN values with the median of other ‘LotFrontage’ values in its neighborhood
4. Outlier removal
   1. Using a quantile method removing only rows with extreme values vs SalePrice
   2. Will revise with a manual method based on feature vs target plots
5. Removing multicollinear features
   1. Features best correlated with ‘SalePrice’ are ‘OverallQual’, ‘GrLivArea’, ‘GarageCars’, ‘GarageArea’, ‘TotalBsmtSF’, ‘1stFlrSF’, ‘FullBath’, ‘TotRmsAbvGrd’, and ‘YearBuilt’
   2. Features that exhibit multicollinearity are ‘TotalBsmtSF’, ‘GrLivArea’, ‘LowQualFinSF’, ‘2ndFlrSF’, ‘1stFlrSF’, ‘BsmtFinSF1’, ‘BsmtFinSF2’, and ‘BsmtUnfSF’.
   3. Will only remove ‘LowQualFinSF’ since it is not correlated with ‘SalePrice’ and also will not be used in feature engineering
6. Feature scaling
   1. Box-cox scaling on numerical features with |skew|>0.5
   2. log(1+x) scaling on ‘SalePrice’
7. Adding new features
   1. ‘TotalSF’ = ‘TotalBsmtSF’ + ‘1stFlrSF’ + ‘2ndFlrSF’
   2. ‘YrBuiltRemod’ = ‘YearBuilt’ + ‘YearRemodAdd’
   3. ‘TotalBath’ = ‘FullBath’ + 0.5\*‘HalfBath’ + ‘BsmtFullBath’ + 0.5\*’BsmtHalfBath’
   4. ‘TotalPorch’ = ‘OpenPorchSF’ + ‘3SsnPorch’ + ‘EnclosedPorch’ + ‘ScreenPorch’ + ‘WoodDeckSF’
   5. Binary features for ‘PoolArea’, ‘2ndFlrSF’, ‘GarageArea’, ‘TotalBsmtSF’, ‘Fireplaces’
8. Encoding ordinal categorical variables manually
9. Encoding nominal categorical variables through dummy encoding
10. Columns not present in either test or train sets were added to each other so that they have the same number of features.