ISYE-6740

Final Project Report
Submitted By: Mark Pearl

Problem Statement:

For this project, I will be delving into a supervised learning problem by performing advanced regression techniques on Kaggle's "House Prices: Advanced Regression Techniques" to explore the full machine learning life cycle from feature engineering / extraction (i.e. including Exploratory Data Analysis), model development, model validation, and finally exploring future work to improve the model performance and recommendations for additional data sources to include.

We will develop a system that will utilize the full predictive power of both our categorical and numerical features, and transform them in a way before fitting them to our regression techniques to provide a prediction on the "Sales Price" of the property. This is a widely studied area that involves thousands of data sources that have both micro and macroeconomic implications for the housing market. Our focus will be strictly on the features that are tied to the property and it's surrounding area.

This system can provide an effective tool for real estate developers to understand best how to price properties in their current portfolio, while accounting for the WTP (Willingness to Pay) and ensuring that their margins are maximized in the process. This could also be used for mortgage insurance providers, who in some cases have to seize a property when a customer has defaulted on their mortgage, and they need to resurface the property back into the housing market, with minimal upkeep as possible. This model will provide an effective baseline model that these parties could use with additional feature engineering (i.e. adding features relevant to their application) and providing estimates on property 'Sales Prices'

Datasets
Contained in the project we have a data folder organized as follows in accordance with the data science cookie cutter project structure.

```
├── data                <- Source code for use in this project.
|   ├── external        ←  Scripts to train models and then use trained models to make
|   |   ├── train.csv     ←  Training dataset we will conduct our CV and model fitting
|   |   └── test.csv       ← Test dataset to assess the prediction accuracy of our model
|   |   └── sample_submission.csv  ← Sample submission for the kaggle competition
|   |   └── data_description.txt ← Data dictionary containing description of the features
```

Our training dataset is broken down into the following features:

Ordinal Features:
LotShape: General shape of property (Needs ordinal encoding)
LandSlope: Slope of property (Needs ordinal encoding)
OverallQual: Rates the overall material and finish of the house
OverallCond: Rates the overall condition of the house
ExterQual: Evaluates the quality of the material on the exterior (Needs ordinal encoding)
ExterCond: Evaluates the present condition of the material on the exterior (Needs ordinal encoding)
BsmtQual: Evaluates the height of the basement (Needs ordinal encoding)
BsmtCond: Evaluates the general condition of the basement (Needs ordinal encoding)
BsmtExposure: Refers to walkout or garden level walls (Needs ordinal encoding)
HeatingQC: Heating quality and condition (Needs ordinal encoding)
KitchenQual: Kitchen quality (Needs ordinal encoding)
FireplaceQu: Fireplace quality (Needs ordinal encoding)
GarageQual: Garage quality (Needs ordinal encoding)
GarageCond: Garage condition (Needs ordinal encoding)
PoolQC: Pool quality (Needs ordinal encoding)
Fence: Fence quality (Needs ordinal encoding)

Categorical Features
Alley: Type of alley access to property
Utilities: Type of utilities available
MSSubClass: Identifies the type of dwelling involved in the sale.
Exterior1st: Exterior covering on house
Exterior2nd: Exterior covering on house (if more than one material)
Street: Type of road access to property
Neighborhood: Physical locations within Ames city limits
MSZoning: Identifies the general zoning classification of the sale.
LandContour: Flatness of the property
LotConfig: Lot configuration
Condition1: Proximity to various conditions
Condition2: Proximity to various conditions (if more than one is present)
BldgType: Type of dwelling
HouseStyle: Style of dwelling
RoofStyle: Type of roof
RoofMatl: Roof material
Foundation: Type of foundation
Heating: Type of heating
Electrical: Electrical system
Functional: Home functionality (Assume typical unless deductions are warranted)

GarageType: Garage location
GarageFinish: Interior finish of the garage
PavedDrive: Paved driveway
MiscFeature: Miscellaneous feature not covered in other categories
SaleType: Type of sale
SaleCondition: Condition of sale
BsmtFinType1: Rating of basement finished area
BsmtFinType2: Rating of basement finished area (if multiple types)

Numerical Features:
LotFrontage: Linear feet of street connected to property
LotArea: Lot size in square feet
YearBuilt: Original construction date
YearRemodAdd: Remodel date (same as construction date if no remodeling or additions)
MasVnrArea: Masonry veneer area in square feet
BsmtFinSF1: Type 1 finished square feet
BsmtFinSF2: Type 2 finished square feet
BsmtUnfSF: Unfinished square feet of basement area
TotalBsmtSF: Total square feet of basement area
1stFlrSF: First Floor square feet
2ndFlrSF: Second floor square feet
LowQualFinSF: Low quality finished square feet (all floors)
GrLivArea: Above grade (ground) living area square feet
BsmtFullBath: Basement full bathrooms
BsmtHalfBath: Basement half bathrooms
FullBath: Full bathrooms above grade
HalfBath: Half baths above grade
Bedroom: Bedrooms above grade (does NOT include basement bedrooms)
Kitchen: Kitchens above grade
TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)
Fireplaces: Number of fireplaces
GarageYrBlt: Year garage was built
GarageCars: Size of garage in car capacity
GarageArea: Size of garage in square feet
WoodDeckSF: Wood deck area in square feet
OpenPorchSF: Open porch area in square feet
EnclosedPorch: Enclosed porch area in square feet
3SsnPorch: Three season porch area in square feet
ScreenPorch: Screen porch area in square feet
PoolArea: Pool area in square feet
MiscVal: $Value of miscellaneous feature

MoSold: Month Sold (MM)
YrSold: Year Sold (YYYY)

Binary Features
CentralAir: Central air conditioning

Exploratory Data Analysis
In statistics, Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns,to spot anomalies,to test hypothesis and to check assumptions with the help of summary statistics and graphical representations. [1]

Firstly, you can tell with the naming conventions of some of our features that we may have an issue with multicollinearity. For instance we can see several variables like 1stFloor Square Footage, 2nd Floor Bsmt Square Footage, Basement Square Footage, etc. These variables will obviously be correlated because they have a child relationship to the overall square footage of the house. Having this can impact the convergence of our model, and provides justification to look at the Pearson

Correlation Matrix:

[1] What is Exploratory Data Analysis https://towardsdatascience.com/exploratory-data-analysis-8fc1cb20fd15

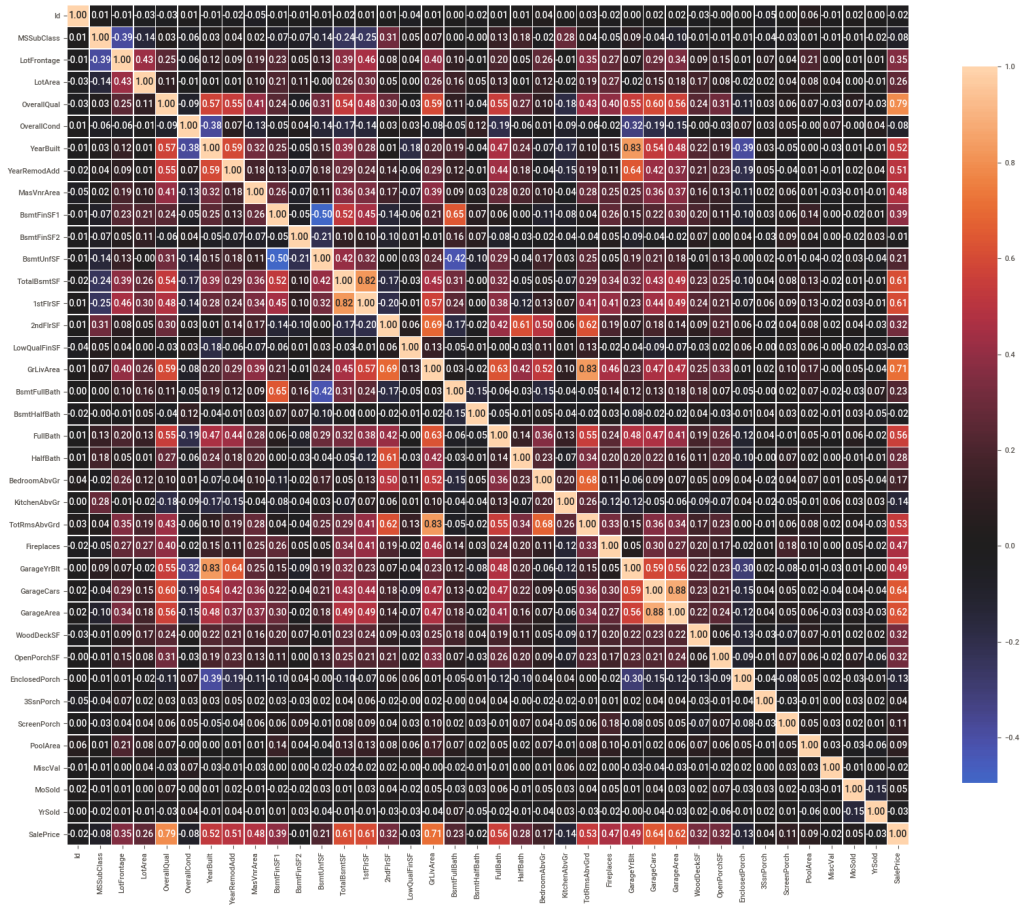


Figure 1: Pearson Correlation Matrix

As a result, our conclusion is correct in relation to these variables. We can see that the correlation is 83% for 1stFloorSquareFootage and TotalBasementSquareFootage. We can also see important insights in relation to the indepedent variables that are correlated with our target variable "SalesPrice". When using a cutoff value of 45%, we only have 13 variables out of numerical features that have a possible correlation with "SalesPrice". This can provide benefit from feature selection standpoint, as we can filter out the variables that have no significance to the target variable like the 'row id', 'bsmt square ft 2', etc.

We can also start to look at the plots of some of our numerical features that provide significance to our 'SalesPrice' dependent variable and determine if any outliers, skewness exists in the distribution of the data. Some features like GarageArea have a beautiful normal distribution as seen here:

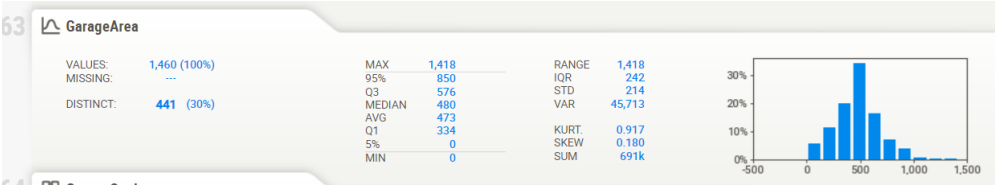


Figure 2: GarageArea EDA Output

However, other numerical features in relation to square feet seem to have some irregularities that need further investigation such as BsmtUnfSf, TotalBsmtSF:

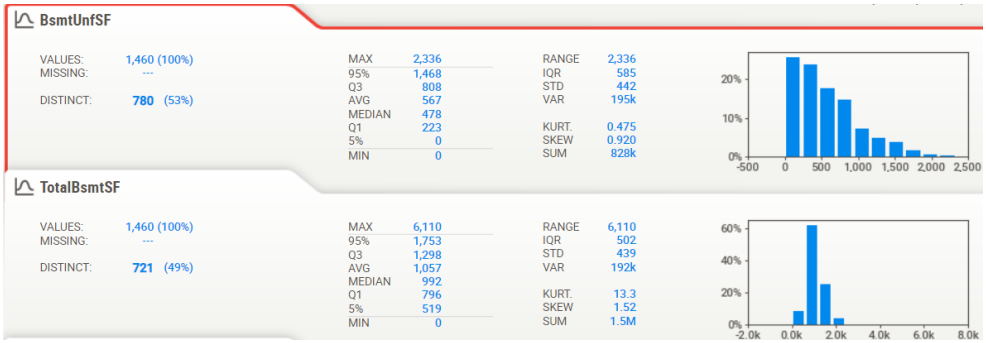


Figure 3: BsmtUnfSf, TotalBsmtSF EDA Output

We can see when looking at the summary statistics that the difference between the max observation and 95% value for the interquartile range for TotalBsmtSF is quite significant, which suggest to me we may have an outlier problem. Same seems to hold true for our dependent variable SalesPrice.

When looking at the boxplot representation of these variables we can see that both variables are exhibiting the outliers:

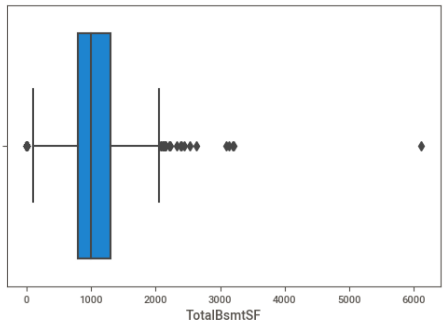


Figure 4: Boxplot Total Basement Square Footage

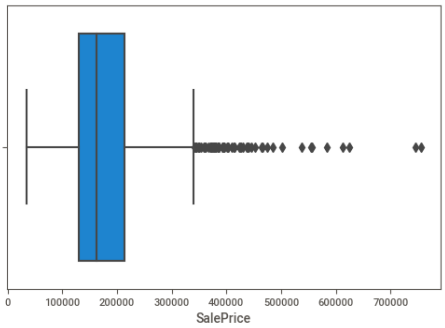


Figure 5: Boxplot SalePrice

We can see that both of these variables have significant outliers that fall well above the 95% mark of the interquartile range. Therefore, our data may need the use of scaling techniques provided by sklearn to reduce the impact these outliers will have on the target prediction.

Feature Engineering & Extraction

From our analysis, we can see we have a significant number of variables. Therefore, we will likely require feature engineering in order to make certain transformations to our variables and address problems with our numerical features such as multicollinearity, outliers, distribution (normal distribution, skewed or bimodal distributions). Applying these methods ultimately result in better predictive performance for our target variable, and allow our models to provide better convergence while reducing the impact of noise that can occur across some of our features.

For our categorical data, there exists several transformations to provide a numerical value for each distinct value of the feature such as one-hot encoding, dummy variables. etc. Will we will be applying these techniques to our data so that it can be useful when used in a supervised manner in this case.

Numerical Features

Generic Feature Extraction / Selection: Some example features will be calculating are generic calculations such as calculating the difference between the current year and the each of the year columns such as YearBuilt, YearGarageBuilt, etc. This will give us an age indication of the house, and help with our interpretation of the effect of these variables.

We will also start to select variables that have a positive correlation with the target variable above a certain threshold based on the pearson correlation matrix. This will help reduce our overall training time of the model and improve our results. However, we will provide join back our prediction results with the property 'id' has this is used in the submission to get the

overall score of our model.

Scaling: Due to our issue of numerical features containing outliers, in addition to not being on the same scale, we will need to use a pre-processing method on these features before we fit our model.
A popular method would be to use the RobustScaler package as a part of sklearn.preprocessing. It provides more advantages when compared to the Standard & Min Max scaling libraries, as it uses a minimum and maximum cutoff of the interquartile range instead of discrete values. In addition it is able to handle variables that don't have normal distributions, which we seem to be exhibiting with some of our independent variables. StandardScaler in this case would not be suitable as it operates based on the assumption that the data is normally distributed, which in some cases for us it is not.

The formula for RobustScaler is as follows[2]:

$$\frac{x_i - Q_1(x)}{Q_3(x) - Q_1(x)}$$

Figure 6: Robust Scaler Calculation

The output of scaling will allow are distributions to be brought to the same scale while minimizing the impact of the outliers.

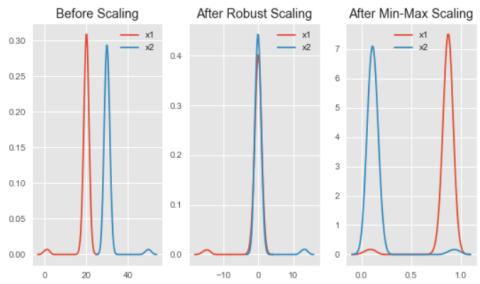


Figure 7: RobustScaler Before & After Comparison

Ordinal Data: We can see that there are a significant number of ordinal attributes that are not scaled to a 1-X value based on their hierarchy. We will use the OrdinalEncoder Library to rectify this by iterating through each feature and provide a value for it's rank. This is better then one hot encoding in this case because our output will be one column, instead of a column for each level like one hot encoding or dummy variables would produce.

Categorical Data: For our categorical data, we will use the common technique of creating dummy variables so that each of our distinct values of our variables get placed into their own column and provided with a numerical value of either 0 or 1, if that value exists in the

[2] Feature Scaling with scikit-learn https://benalexkeen.com/feature-scaling-with-scikit-learn/

column.[3] In our case we will use the pandas library pd.dummies().

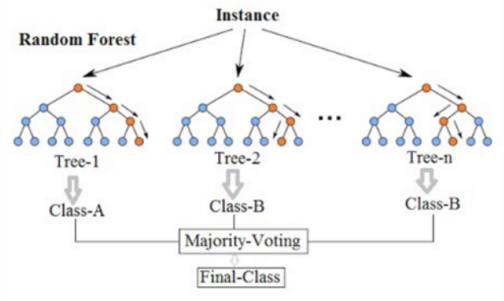
Imputation: In order to sklearn to be able to converge when fitting models, there cant be any Nan or Inf values contained in the dataset. Therefore, a section in the feature.py section is used to impute these missing values for both categorical (i.e. replace with None) and numerical (i.e. Replace with 0). A value of 0 was provided because that feature simply wasn't available on the house, so it makes sense to be provided with a value of 0. However, the missing values for the GarageYrBlt feature was imputed with the YearBuilt feature, and the remaining features were replaced with the mean for it's respective column.

Now that are our features are ready we'll be able to move onto the modelling step of our process.

Model Development & Validation

Since we're trying to solve a supervised problem on the predicted variable 'SalesPrice', we should be focusing on regression techniques that have capabilities for feature selection, and ensuring we don't overfit our data.

Random Forest:
To start we will explore using tree based method called RandomForest regressor. As learned in class, RandomForest trees will be developed using smaller subsets of our features to construct smaller trees. An average result for each of these trees will then be used to inform a prediction on our target variable. The tree will then conduct pruning to only include values ranges with statistical significance when predicting against our target variable. [4]



Random Forest Simplified (Source)

Figure 8: RandomForest Depiction

[3] Categorical Data https://towardsdatascience.com/understanding-feature-engineering-part-2-categorical-data-f54324193e63
[4] Understanding Random Forest https://towardsdatascience.com/understanding-random-forest-58381e0602d2

ElasticNet, Ridge and Lasso

These models found in the linear modelling library form a true multiple regression model against our data and use the regularization benefits provided from L1 and L2 normalization. Firstly, a critical concept to tie into this problem is the Bias-Variance Tradeoff for simple linear regression. When trying to determine our coefficient values B, we need to make an estimate from a sample size and not the true population. For ordinary least squares, we need to ensure that sum of square residuals is as low as possible. These parameter estimates are based on the following equation[5]:

$$L_{OLS}(\hat{\beta}) = \sum_{i-1}^{n}(y_i - x_i'\hat{\beta})^2 = ||y - X\hat{\beta}||^2$$

To achieve this, we need to try and capture the bias (true population parameter and the expected estimator) and the variance (measures the spread or uncertainty in our data). These two metrics are interconnected, and we need to find the minima of our total error that provides the optimal variance and bias value:

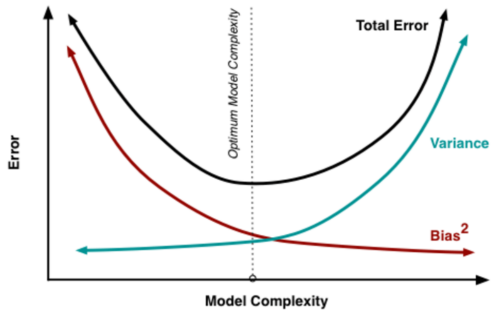


Figure 9: Bias vs Variance Tradeoff

These 3 models take into consideration this concept, but each with differing L1 and L2 regularlization techniques.

Model	Penalty	Feature Selection

[5] Regularization: Ridge, Lasso and Elastic Net
https://www.datacamp.com/community/tutorials/tutorial-ridge-lasso-elastic-net

Ridge	L2 Penalty	Places the coefficient values for the less useful features as close to 0 as possible
Lasso	L1 Penalty	Lasso Places the coefficient values for the less useful features to 0
ElasticNet	Combination of Both	Takes the advantages of both models by setting the lambda between 0 and 1 to determine how much of either model is used.

Biggest advantages of these models is that they inherently solve multicollinearity which is going to be present in this dataset, when compared to the tree based methods which can sometimes not fully account for these correlations, or may miss import correlations entirely.

Cross-Validation

We will conduct cross validation on our training data in order to split our model into the appropriate training and validation datasets when fitting our model. K-fold cross validation will ensure that we don't use our test dataset prematurely. Initially, the entire training data set is broken up in k equal parts. The first part is kept as the hold out (testing) set and the remaining k-1 parts are used to train the model. Then the trained model is then tested on the holdout set. This process is repeated k number of times until we have our result. For the purposes of our model, we will be using k = 5. The concept can be visualized as follows[6]:

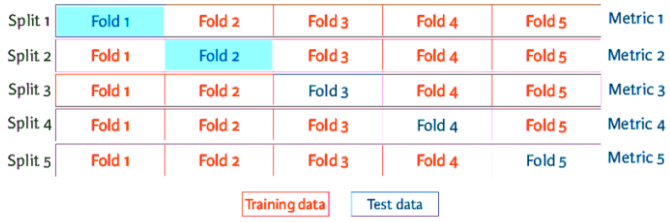


Figure 10: k-fold cross validation

GridSearchCV

An extremely popular technique when conducting hyperparameter tuning is to use GridSearch. This method will ensure that we're using the most optimal values for a range of ~~string or integer parameter r~~anges when we run the training process on each model.

[6] Cross Validation https://towardsdatascience.com/cross-validation-430d9a5fee22#:~:text=Refer%20the%20docs.-,Cross%20Validation,complementary%20subset%20of%20the%20data.

For example,

Provided is the parameter set used to tune the Elastic Net model:

```
param_grid = {'alpha': np.arange(1e-4,1e-3,1e-4),
'l1_ratio': np.arange(0.1,1.0,0.1),
'max_iter':[100000],
'tol':[10]}
```

Gridsearch will iterate through the range of values for the alpha and l1_ratio and ensure that the optimal hyperparameter value is chosen when producing the final result.

Model Results:

For our model validation criteria we will run our prediction against the test dataset and calculate the RMSE (Root Mean Square Error). This is calculated with the following formula:

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

Figure 11: RMSE Formula

Model Name	Model Score	RMSE Value
Ridge	0.901361	24941.873190
Lasso	0.772974	37839.171679
ElasticNet	0.760011	38904.535546
RandomForestRegressor	0.542331	52245.232431

The Ridge Regression model seems to provide the best performance with a score of 90% accuracy an RMSE of 24941. Due to the significant number of correlated features to the target variable, it would make sense that Ridge performs the best as it is able to handle many large parameters, while Lasso tends to perform better when only a fewer number of features provide statistical significance to the target variable.

Conclusion & Improvements

In conclusion the model will provide useful predictions for sales prices to any party interested

in understand predictions or valuations on an existing property. To improve the model's performance, we would benefit from more data as the model currently only has 1460 observations, and is likely causing our model to overfit. In addition, more macroeconomic data related to the housing market would gain a sense of the stance of the market (i.e. buyers vs sellers market), in addition to demographic data and income information related to the people within the area. By adding this data we would be able to improve predictions against our target variable.

To deploy the model to a production environment, we could utilize technologies such as MLFlow, in order to provide a pickled version of the model that can be accessed directly using a predict() call. This will grab the most recent version of the pickle file, and provide results to any downstream business users.