

MID-TERM PROJECT

Noam Atias 311394357
Chanel Michaeli 208491787

House Price Prediction – RidgeCV

Goal:

Predicting house prices based on given house features.

Model:

We chose to predict house prices using RidgeCV regression.

House Features

The data contains many features, some are numerical, and others are categorical.

The data structure:

- Number of features: 81
- Dataset is divided to train dataset and test dataset
- Shape of train dataset: 1460 houses, 81 features for each house.
- Shape of test dataset: 1459 houses, 80 features for each house.
- Target feature: 'SalePrice' (called Y_train)

General Outline

The general outline of our notebook is as follows -

- 1) Data analysis- exploring the data
- 2) Preprocessing - dealing with outliers, replacing null values, etc.
- 3) Model Building –RidgeCV and K-fold
- 4) Model Fitting and Evaluation
- 5) Submission

RidgeCV Technique

- Ridge regression is a model tuning method that is used to analyze any data that suffers from multicollinearity.
- This method performs L2 regularization.
- When the issue of multicollinearity occurs, least-squares are unbiased, and variances are large, this results in predicted values being far away from the actual values. So, ridge regression defines Lambda which is the penalty term.

RidgeCV Technique - Continue

- Lambda is denoted by an alpha parameter in the ridge function. So, by changing the values of alpha, we are controlling the penalty term. The higher the values of alpha, the bigger is the penalty and therefore the magnitude of coefficients is reduced.
- RidgeCV is ridge regression with built-in cross-validation. By default, it performs efficient Leave-One-Out Cross-Validation, which helps us deal with overfitting.
- Overfitting is a modeling error in statistics that occurs when a function is too closely aligned to a limited set of data points. As a result, the model is useful in reference only to its initial data set, and not to any other data sets.

Data Analysis

In this section, we explored the data structure and showed some data statistics :

1. Size of train set and test set:

The train data size before dropping Id feature is : (1460, 81)

The test data size before dropping Id feature is : (1459, 80)

Test set does not contain the feature 'SalePrice' , which we want to predict.

2. Features (columns) of the train set:

'Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street' , 'Alley', 'LotShape', 'LandContour', 'Utilities',
'LotConfig' , 'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType','HouseStyle', 'OverallQual', 'OverallCond',
'YearBuilt', 'YearRemodAdd','RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType','MasVnrArea', 'ExterQual',
'ExterCond', 'Foundation', 'BsmtQual','BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1','BsmtFinType2',
'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating','HeatingQC', 'CentralAir', 'Electrical', '1stFlrSF',
'2ndFlrSF','LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath','HalfBath', 'BedroomAbvGr',
'KitchenAbvGr', 'KitchenQual','TotRmsAbvGrd', 'Functional', 'Fireplaces', 'FireplaceQu', 'GarageType','GarageYrBlt',
'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual','GarageCond', 'PavedDrive', 'WoodDeckSF',
'OpenPorchSF','EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'PoolQC','Fence', 'MiscFeature', 'MiscVal', 'MoSold',
'YrSold', 'SaleType','SaleCondition', 'SalePrice'

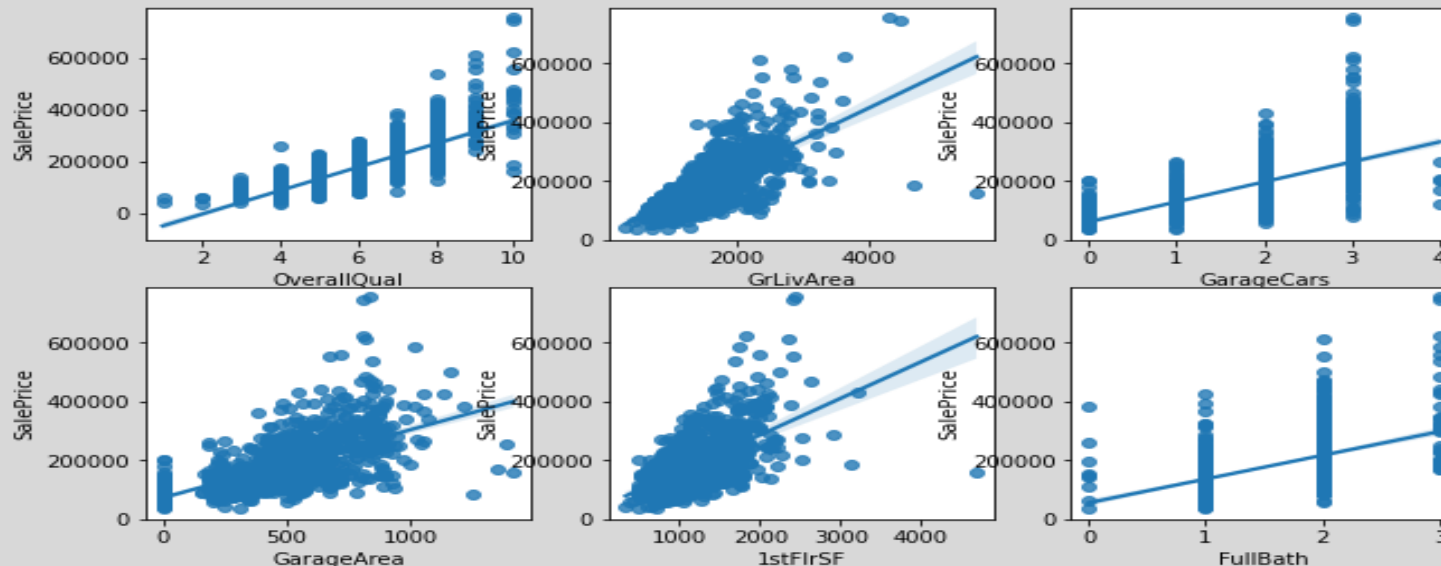
3. We can see in this section the statistic of the numerical features and the categorical features in the subsections - **Train Dataset Statistics** and **Summary statistics for categorical values**.

4. Checking the Correlation between the variables and the target variable –

If we want to use a linear regression model (ridge regression), we need to check the correlation between the variables and the 'SalePrices' Feature.

We can see the results of the correlations in the notebook.

Here, we will show the six best correlations:

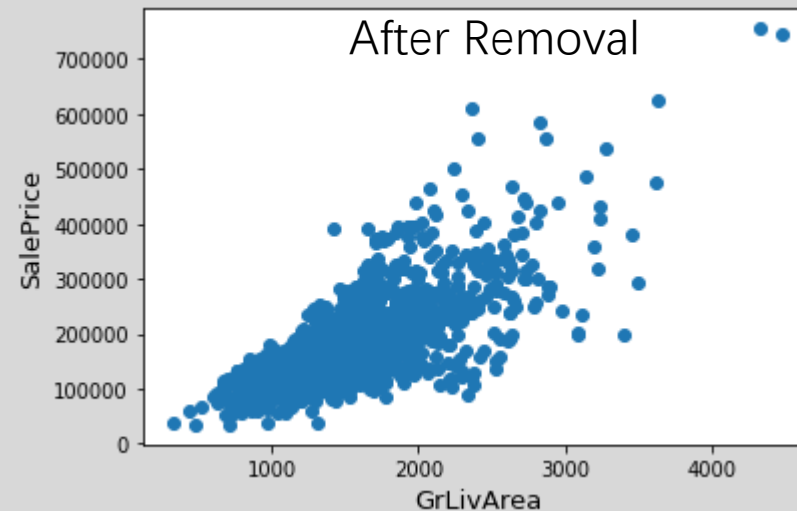
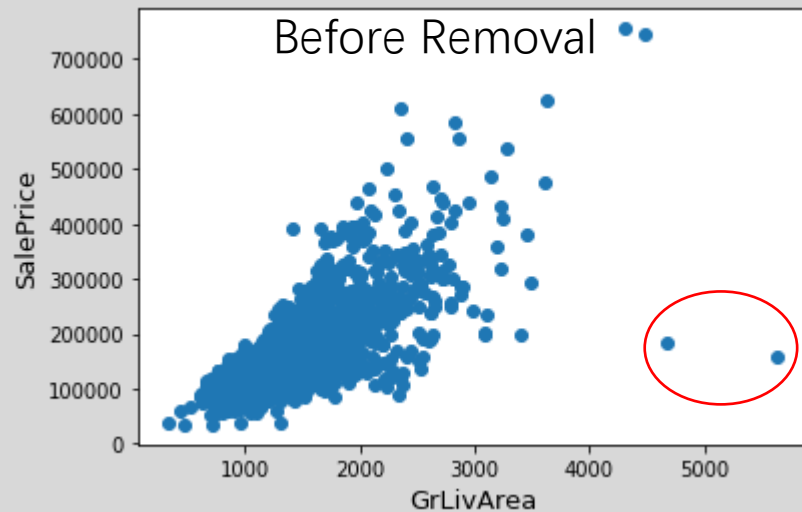


Data Preprocessing

1. Outliers

We can see in the plot that the feature 'GrLivArea' has 2 points that doesn't match the correlation between the feature and the target variable (the points in the lower right corner have extremely large GrLivArea and at low low SalePrice).

Therefore, we can remove those outliers:



2. Saving the IDs and removing from datasets

We want to remove the column ID from the test set and the train set and save them, so that in the submission section we will use the test ID.

3. Splitting the target variable

The train set contain the feature 'SalePrices' which is the target of our task.

Therefore, we will separate the target variable from the train set and save it as Y_train.

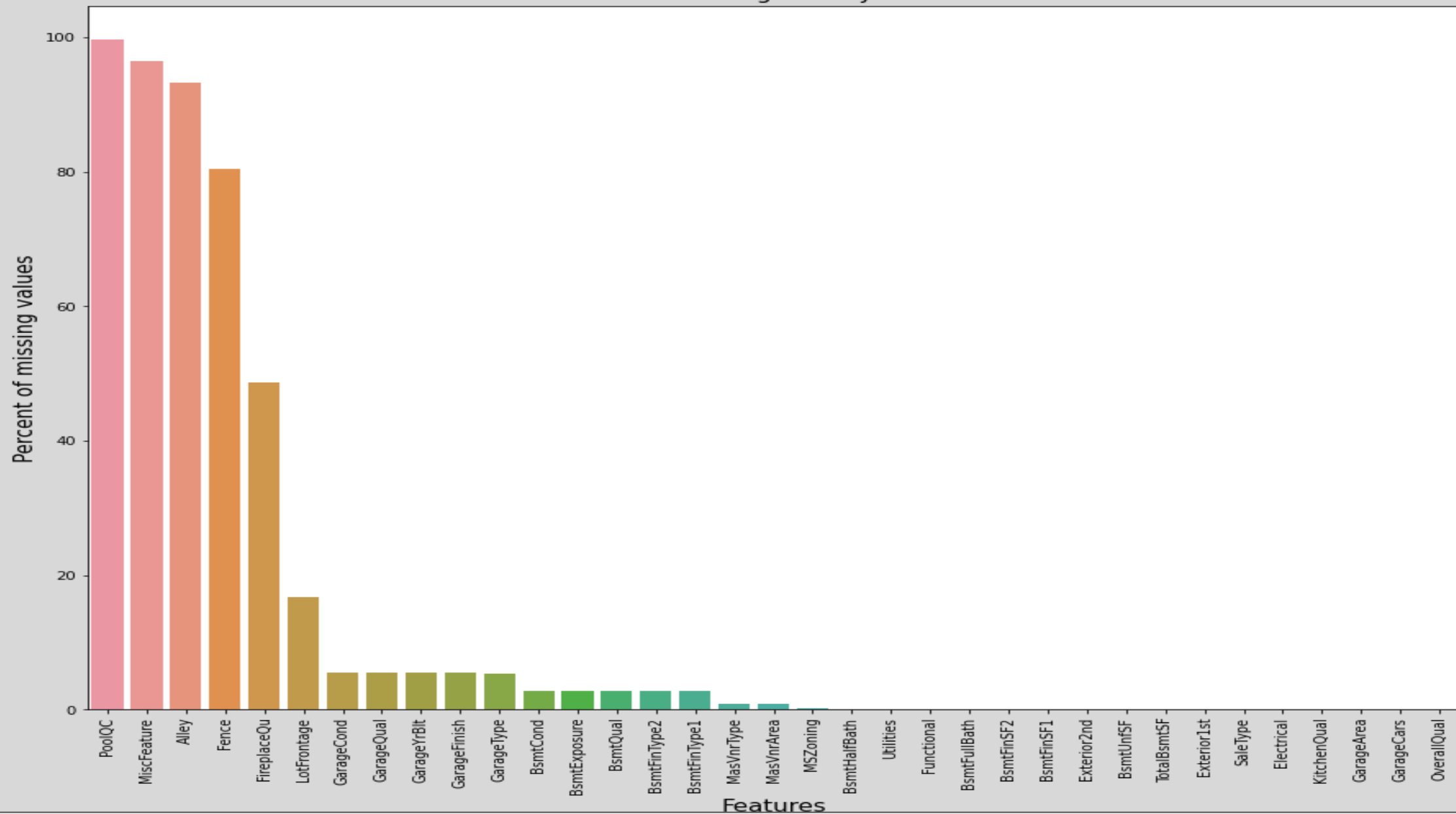
4. Dealing with missing values by each feature

To with missing values in both test set and train set, we combine them to one. Now the size of the data set is (2917, 79) – 2917 houses and 79 features.

First, we checked that there are no missing values in the target features.

Then, we checked the number and percentage of missing values in each feature.

Percent missing data by feature



- In the data description file that was given to us, we checked to see if missing values 'NA' has a meaning:

1. PoolQC - 99% of the values are missing, and it makes sense that most of the houses doesn't have pool, so we can fill in the missing values with 'None' (no pool).

2. MiscFeature - NA means no misc feature, so we can fill in the missing values with 'None' .

3. Alley - NA means no alley access, so we can fill in the missing values with 'None' .

4. Fence - NA means no fence, so we can fill in the missing values with 'None' .

5. FireplaceQu - NA means no fireplace, so we can fill in the missing values with 'None' .

6. LotFrontage - replacing missing values with the most common value.

7. GarageType, GarageFinish, GarageQual, GarageCond - Replacing missing values with 'None' .

8. GarageYrBlt, GarageArea, GarageCars - Replacing missing values with 0.

9. BsmtFinSF1, BsmtFinSF2, BsmtUnfSF, TotalBsmtSF, BsmtFullBath, BsmtHalfBath - Replacing missing values with 0 for having no basement.
 10. BsmtQual, BsmtCond, BsmtExposure, BsmtFinType1, and BsmtFinType2 - Replacing missing values with 'None' for having no basement.
 11. Functional - NA means value 'typ' .
 12. We will replace all other missing values with the most common value of each feature.
- We' ve replaced the missing values and now there are no missing values in the dataset.

5. Scaling datasets

The numerical features are scaled differently, some have low values and others have high values. We scale the numerical features so that all values will be between 0 to 1. We scale by subtracting the mean and dividing by the standard deviation of each variable.

6. Encoding the categorical features

- The number of categorical features is: 43
- We can see in the description file that some categorical variables contain information in their ordering set.

- For example, the feature 'FireplaceQu' has the categories:

Ex: Excellent - Exceptional Masonry Fireplace

Gd: Good - Masonry Fireplace in the main level

TA: Average - Prefabricated Fireplace in the main living area or Masonry Fireplace in basement

Fa: Fair - Prefabricated Fireplace in basement

Po: Poor - Ben Franklin Stove

NA: No Fireplace

- Those features are: 'FireplaceQu', 'BsmtQual', 'BsmtCond', 'GarageQual', 'GarageCond', 'ExterQual' , 'ExterCond' , 'HeatingQC', 'PoolQC', 'KitchenQual', 'BsmtFinType1', 'BsmtFinType2', 'Functional', 'Fence' , 'BsmtExposure', 'GarageFinish', 'LandSlope', 'LotShape', 'PavedDrive', 'Street', 'Alley', 'CentralAir' .
- We encode those features to maintain the ordering meaning of the categories.

- The number of remaining categorical features is: 21
- The remaining categorical features are:
'MSZoning', 'LandContour', 'Utilities', 'LotConfig', 'Neighborhood', 'Condition1', 'Condition2',
'BldgType', 'HouseStyle', 'RoofStyle' , 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType',
'Foundation' , 'Heating', 'Electrical', 'GarageType', 'MiscFeature', 'SaleType' , 'SaleCondition' .
- Those features have no meaning to order. For all remaining categorical features, we will use get dummies, so each category is converted to an indicator variable.
- Now, the size of the data set is (2917, 221) – 2917 houses and 221 variables.

7. Splitting the data to train, test and validation:

Now, we can split the data set back to the test set and train set, so the test set contains 1459 houses and 221 variables, and the train set contains 1458 houses and 221 variables.

In addition, we will split the train set into validation set and train set.

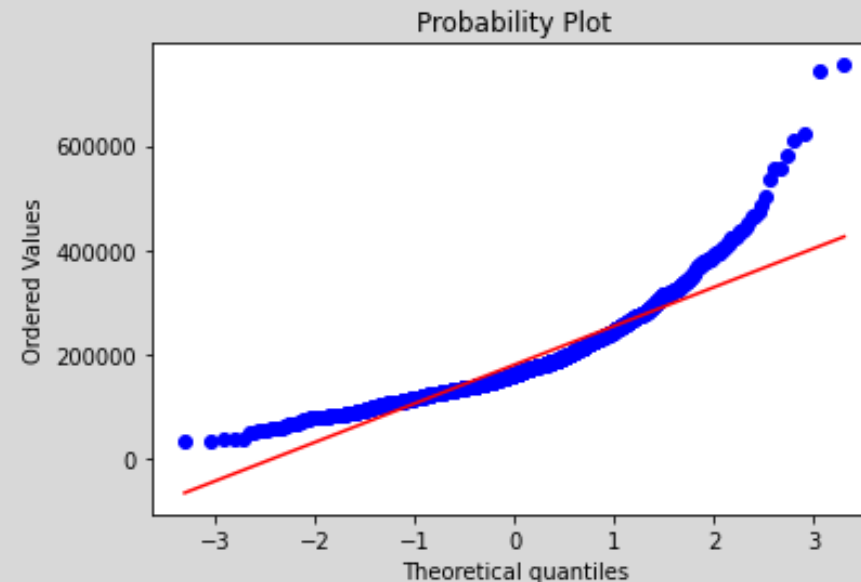
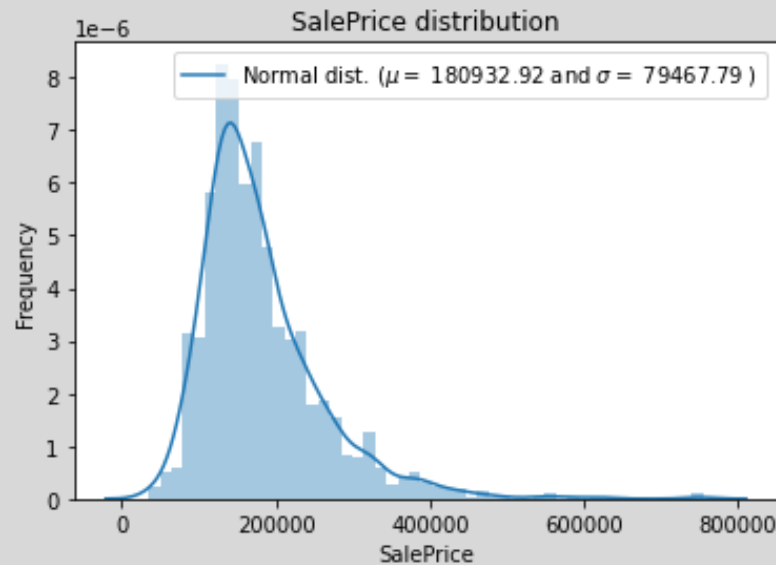
The validation set will help us estimate the accuracy of the model by evaluating data it hasn't trained on.

The shape of all data sets:

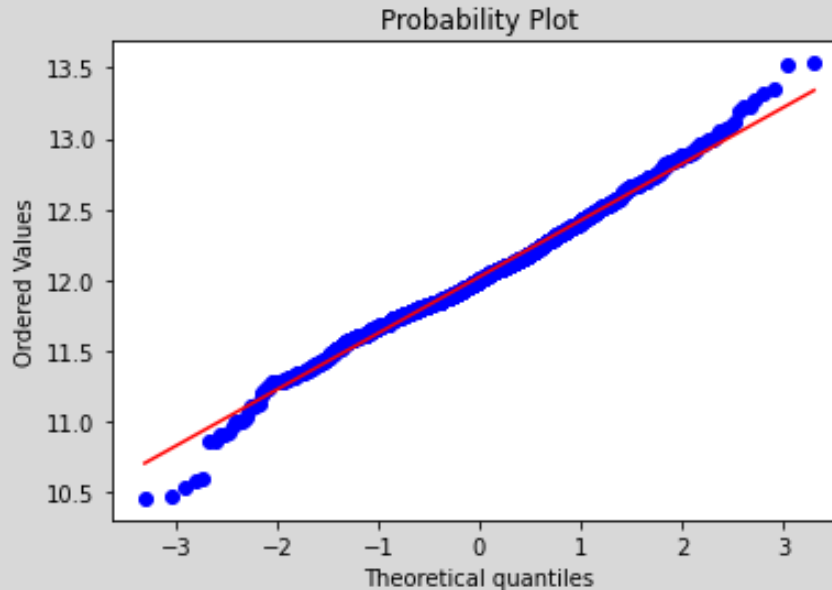
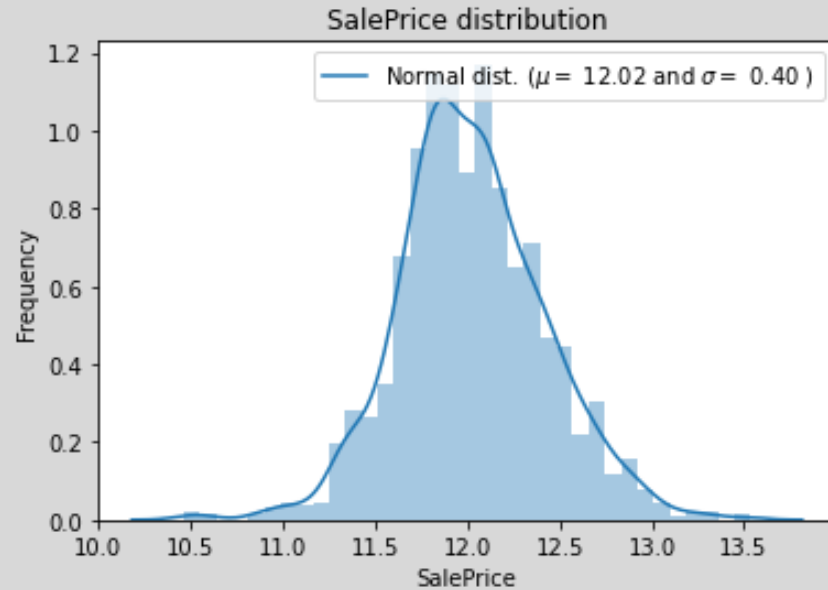
- Shape of test dataset: (1459, 221)
- Shape of train dataset: (1210, 221)
- Shape of validation dataset: (248, 221)
- Shape of train target variable: (1210,)
- Shape of validation target variable: (248,)

8. Log-transformation of the target variable

- We checked the distribution of the target variable 'SalePrices' .
In addition, we checked the probability plot for assessing whether the data set follows a specified theoretical distribution (the normal distribution).
- Also, we computed the skewness of the variable, which is 1.881296 .
- We can conclude that the variable is right-skewed, meaning that the mean is biased towards a higher price than the median. It is also deviating from the normal distribution.



- Using $\log(y+1)$ provides a nice distribution:



- We can see that now the mean and the standard deviation are relatively very small. The variable is normally distributed, and the data set follows the normal distribution.
- The target variable has a more symmetric distribution in log-space, and therefore we apply the $\log(y+1)$ on the validation target variable and the train target variable.

Model Building

- We build the model RidgeCV and defined the K-fold technique for cross-validation, using $K=5$ and shuffle.
- What makes this regression model more effective is its ability to regularize. The term "regularizing" stands for models' ability to structurally prevent overfitting by imposing a penalty on the coefficients. The main tuning parameter for the regularization model is alpha - a regularization parameter that measures how flexible our model is. When alpha is too large, the regularization is too strong, and the model cannot capture all the complexities in the data. However, if we let the model be too flexible (small alpha), the model begins to overfit.

Model Fitting and Evaluation

1. Make Predictions

- We searched for the optimal alpha value that gives us the best model fitting and the highest accuracy.
- We made predictions for each dataset (train, validation, test)

2. Evaluate The Model

- The optimal alpha value we got is 18.801
- We will use the RMSE and R^2 to evaluate our models' prediction on train set and validation set:
- R^2 score for training is : 0.9373253893260599
- R^2 score for validation is : 0.9099295234081932
- RMSE for training is : 0.10086817857730129
- RMSE for validation is : 0.11489958962635259

- R-squared is a statistical measure of how close the data are to the fitted regression line. R-squared equal to 1 indicates that the model explains all the variability of the response data around its mean.
- The root-mean-square error (RMSE) is a frequently used measure of the differences between values predicted by a model and the values observed. We want the RMSE close to zero.
- In our case, the R-squared of the training data is very close to 1 and the R-squared of the validation data is a little smaller but also very close to 1.
- In addition, the RMSE of the training data is very close to zero, and the RMSE of the validation data is, as expected, a little bigger but also very close to zero.
- In conclusion, the model we build is well fitted, we did not get overfitting, and the prediction of the validation data is relatively good.
- Therefore, we can use the prediction we made on the test set and upload it to Kaggle.

3. Inverse-transformation of the target variable

We returned the target variable and the predicted variable to their original values to present the results.

4. Results Presentation

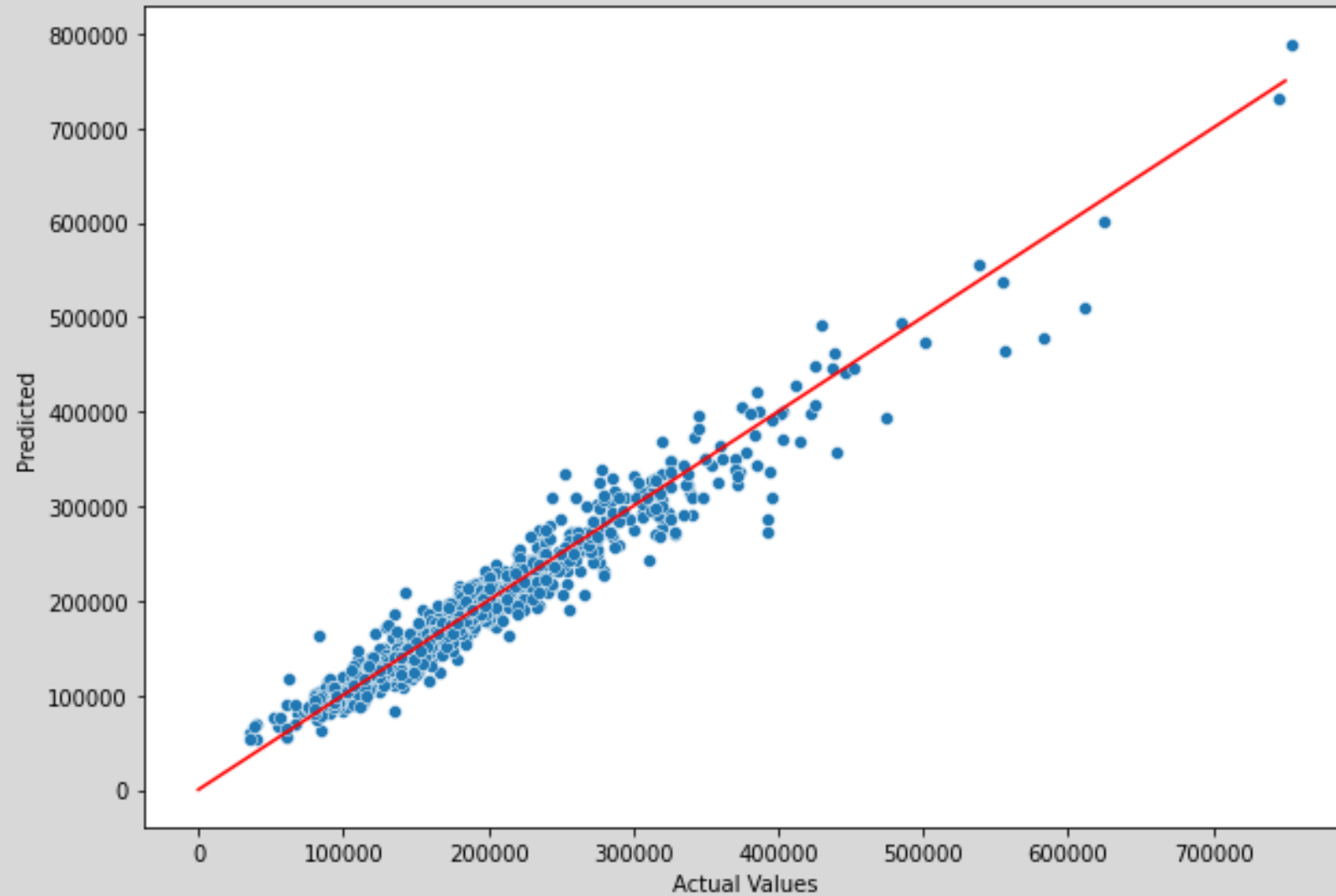
We plotted the predicted values and the real values of the target variable for the train set and the validation set, to see if the predicted values match the real values.

The plots are shown in the next slides.

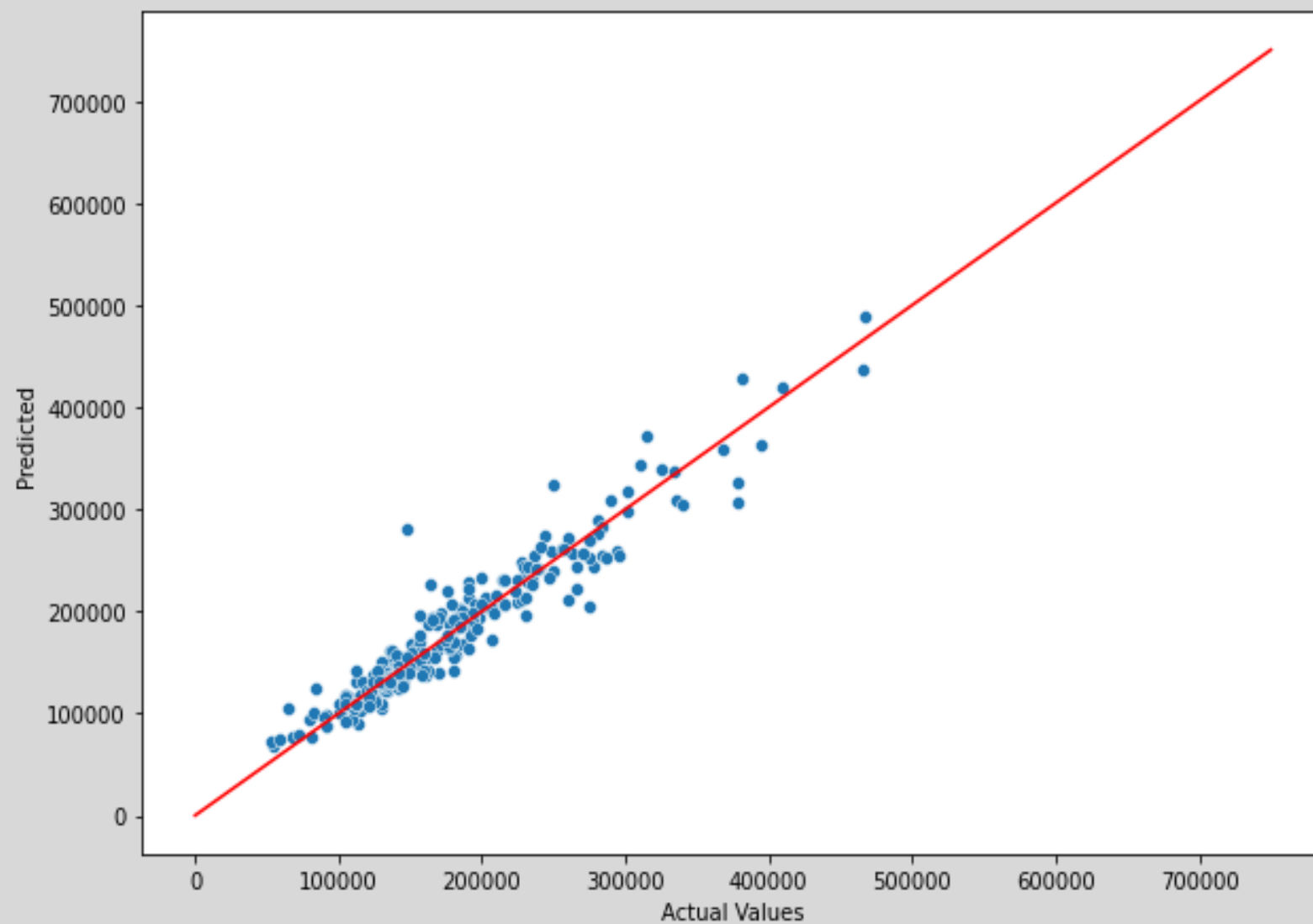
We can conclude that our predictions are very accurate.

The model has trained well and made quite accurate predictions for data points it has not trained on.

RidgeCV Regression Model - Training Results

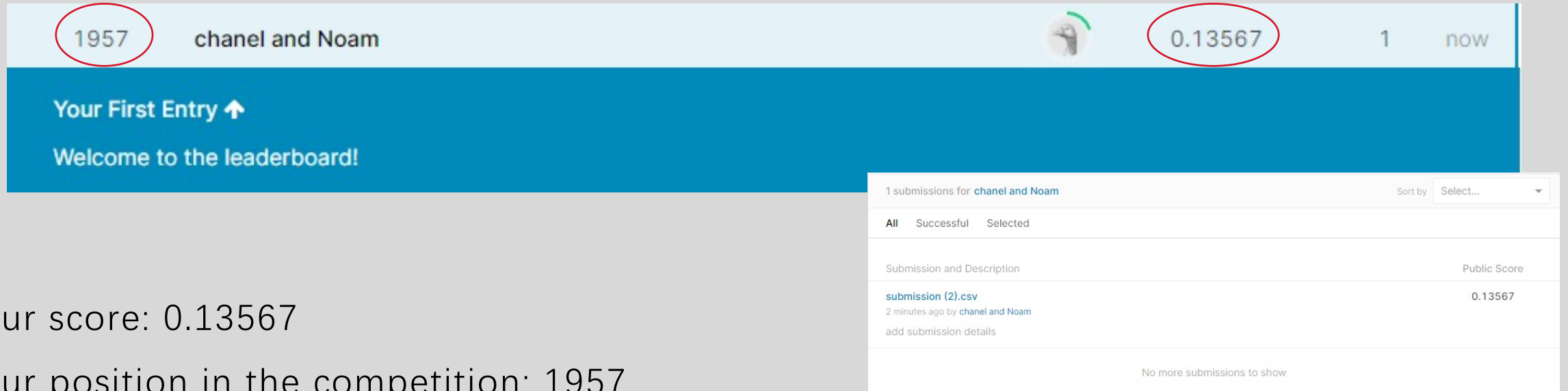


RidgeCV Regression Model - Validation Results



5. Submission

Our submission to Kaggle competition:



The image shows a Kaggle submission leaderboard. At the top, a light blue header bar displays the submission rank '1957' (circled in red), the team name 'chanel and Noam', a profile icon, the public score '0.13567' (circled in red), the number of submissions '1', and the time 'now'. Below this is a blue banner with the text 'Your First Entry ↑' and 'Welcome to the leaderboard!'. To the right, a white box shows a list of submissions for 'chanel and Noam'. It includes a 'Sort by' dropdown menu and tabs for 'All', 'Successful', and 'Selected'. The submission list has two columns: 'Submission and Description' and 'Public Score'. One submission is listed: 'submission (2).csv' with a public score of '0.13567'. It was submitted '2 minutes ago by chanel and Noam' and has a link to 'add submission details'. At the bottom of the list, it says 'No more submissions to show'.

Submission and Description	Public Score
submission (2).csv 2 minutes ago by chanel and Noam add submission details	0.13567

Our score: 0.13567

Our position in the competition: 1957