AnneLouise de Boer

Assignment 2

documents:

- NewModelling.py
- Modelling.py

The dependent variable is the “Sale Price” of the houses in Ames.

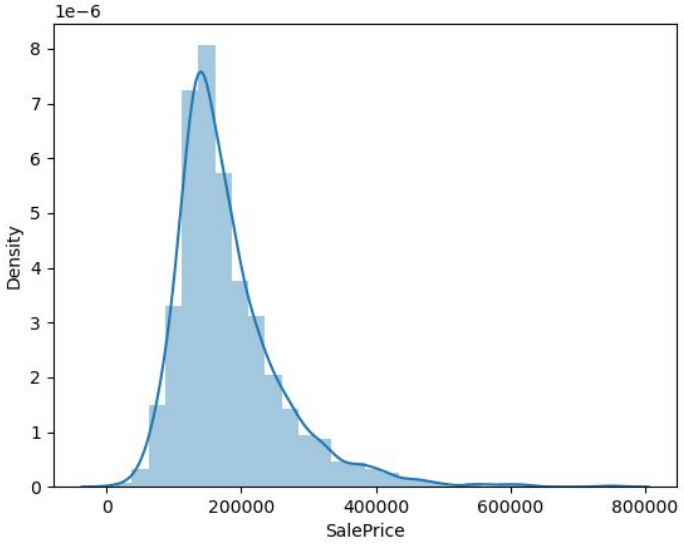
Introduction

My biggest mistake in this assignment was that I initially did not spend enough time on data preprocessing. I did not focus enough on data exploration and jumped directly to the regression methods.

First, I created a file “modelling” where I only used 5 features (columns_to_use). After spending a lot of time testing/trying different things I created another file called (newmodelling) where I used more features. On both train data (modelling) and train_2 data (newmodelling) I applied my models and I will compare these results.

Housing Prices

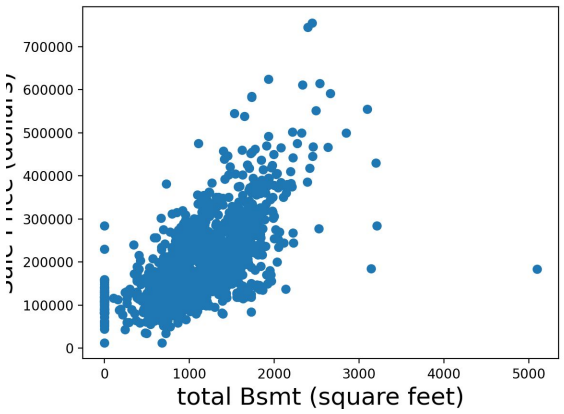
The Plot shows that the Sale Prices are positively skewed. There appear to be a lot of outliers with expensive houses. Next time I should consider removing outliers. The distplot of the training data shows the following distribution for sale prices.

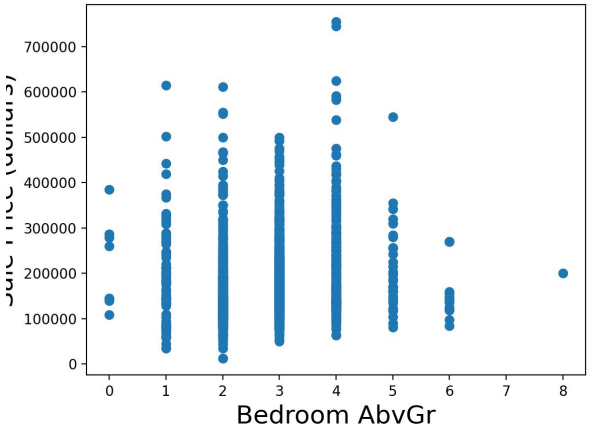


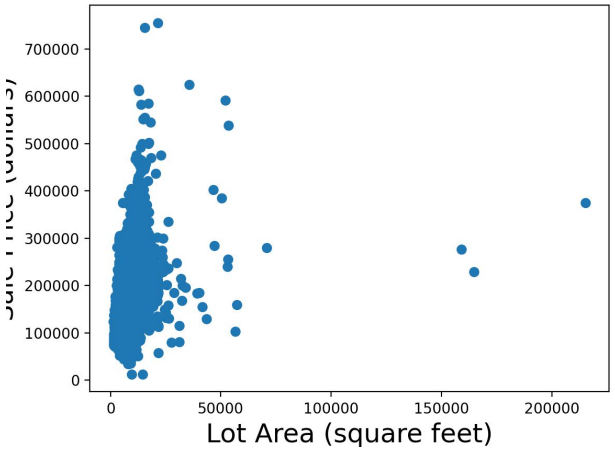
It is not normally distributed and this could reduce the performance of the ML because it assumes a normal distribution. Next time I could make a log transformation to make the distribution more normal.

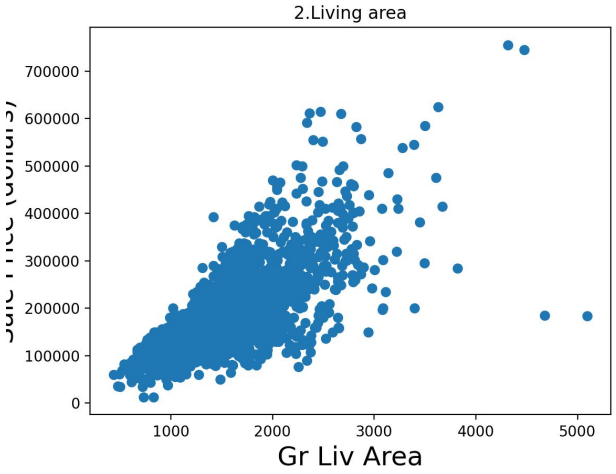
Looking at features

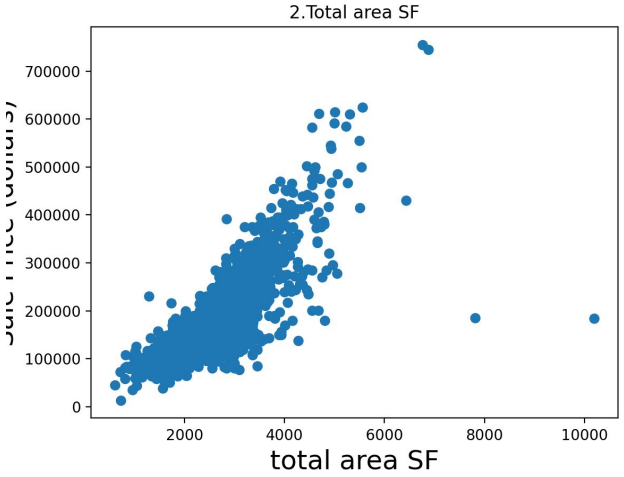
In the document Graphs I plotted some of the features against the SalePrice


700000
600000
500000
400000
300000
200000
100000
0
0
1000
2000
3000
4000
5000
total Bsmt (square feet)

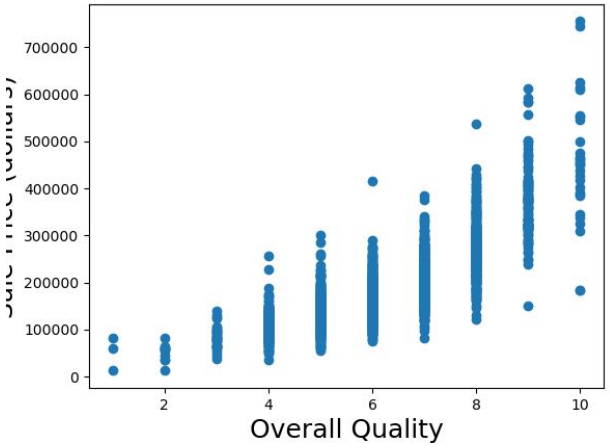

700000
600000
500000
400000
300000
200000
100000
0
0
1
2
3
4
5
6
7
8
Bedroom AbvGr


700000
600000
500000
400000
300000
200000
100000
0
0
50000
100000
150000
200000
Lot Area (square feet)



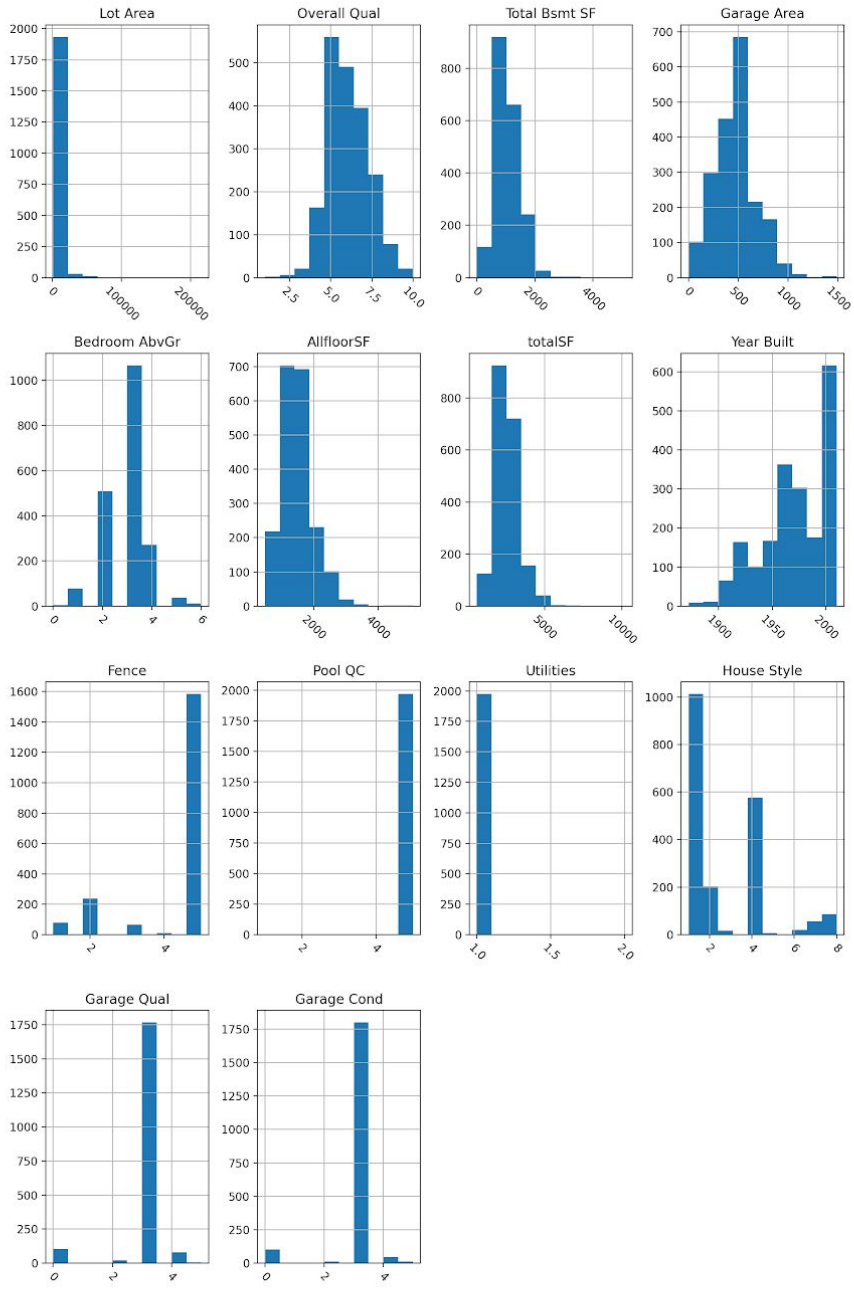


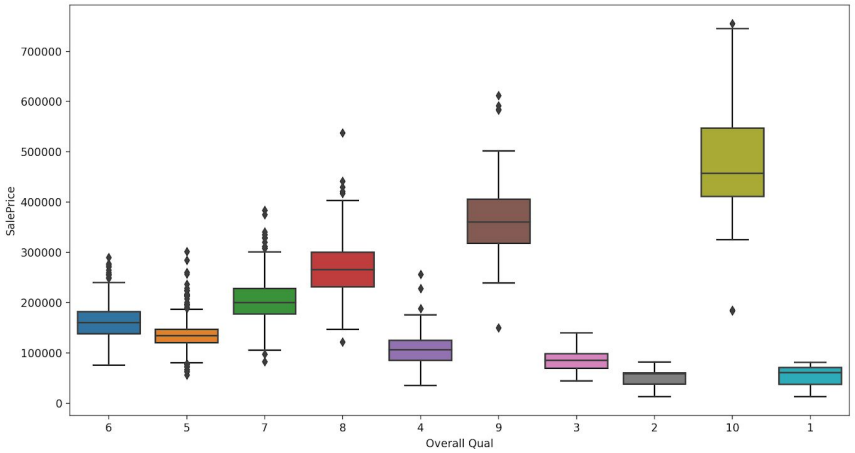
When plotting some of the features against housing prices we can see that values such as the total area could indicate a higher sale price. This newly created column already looks better than one of its constituents Gr Liv Area, for example (see new columns)
I did not remove outliers, but I believe that that could be a good idea next time.



Other features such as Overall Quality are numerical but the numbers seem to indicate categories. I tried changing numerical values like this into categorical values.

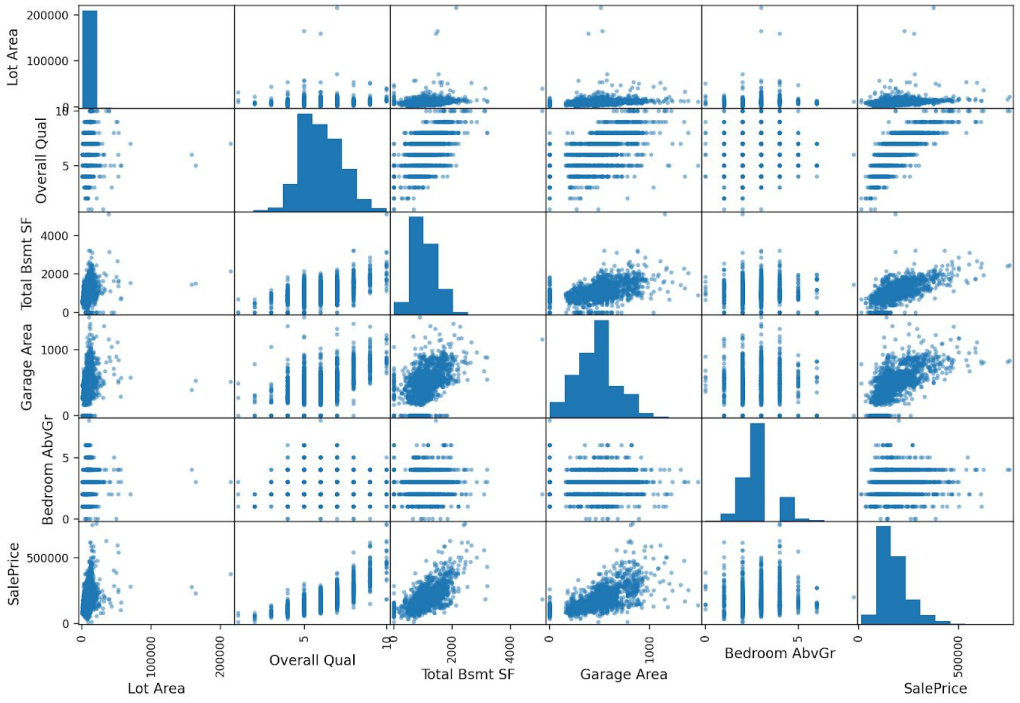
I plotted a histogram for the features in (more_columns)





There seems to be a stronger correlation between the overall quality and the sale price.

Scatter Matrix



Categorical features

Looking at the columns, I found that the following columns are categorical features and numerical features.

categorical =

['MS Zoning', 'Street', 'Alley', 'Lot Shape', 'Land Contour', 'Utilities', 'Lot Config', 'Land Slope', 'Neighborhood', 'Condition 1', 'Condition 2', 'Bldg Type', 'House Style', 'Roof Style', 'Roof Matl', 'Exterior 1st', 'Exterior 2nd', 'Mas Vnr Type', 'Exter Qual', 'Exter Cond', 'Foundation', 'Bsmt Qual', 'Bsmt Cond', 'Bsmt Exposure', 'BsmtFin Type 1', 'BsmtFin Type 2',

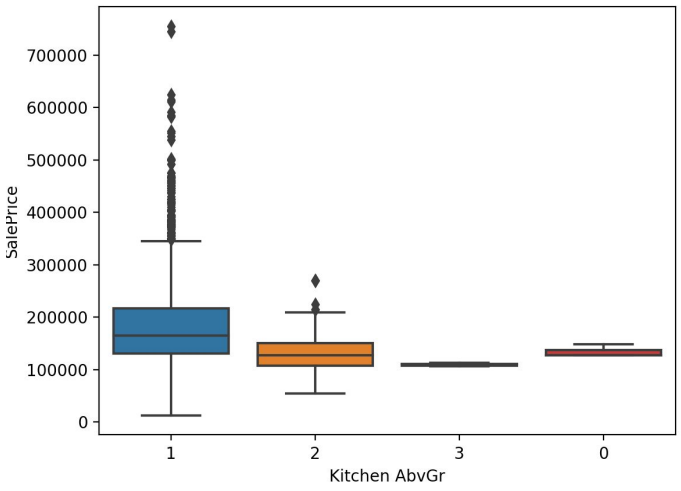
'Heating', 'Heating QC', 'Central Air', 'Electrical', 'Kitchen Qual', 'Functional', 'Fireplace Qu', 'Garage Type', 'Garage Finish', 'Garage Qual', 'Garage Cond', 'Paved Drive', 'Pool QC', 'Fence', 'Misc Feature', 'Sale Type', 'Sale Condition']

numerical =
['PID', 'MS SubClass', 'Lot Frontage', 'Lot Area', 'Overall Qual', 'Overall Cond', 'Year Built', 'Year Remod/Add', 'Mas Vnr Area', 'BsmtFin SF 1', 'BsmtFin SF 2', 'Bsmt Unf SF', 'Total Bsmt SF', '1st Flr SF', '2nd Flr SF', 'Low Qual Fin SF', 'Gr Liv Area', 'Bsmt Full Bath', 'Bsmt Half Bath', 'Full Bath', 'Half Bath', 'Bedroom AbvGr', 'Kitchen AbvGr', 'TotRms AbvGrd', 'Fireplaces', 'Garage Yr Blt', 'Garage Cars', 'Garage Area', 'Wood Deck SF', 'Open Porch SF', 'Enclosed Porch', '3Ssn Porch', 'Screen Porch', 'Pool Area', 'Misc Val', 'Mo Sold', 'Yr Sold', 'SalePrice']

numerical features

The numerical features are either discrete or continuous. Some of the discrete numerical values act as categories. For example, 'Overall Cond' ranks from 1 to 10. I tried turning some of these features into categorical features.

The following boxplot shows this transformation to a categorical variable.



Missing Data

There are many columns where the missing value is important. Columns such as Alley, Fence and Pool QC, are missing more than 70% of the data in the column. I replaced the missing values for these columns with None as a missing value is not a missing value, it actually indicates that there is no pool e.g. Replacing these NaN with the mean would not make sense in this case. I did this for both training data and testing data.

New columns

I tried to combine columns from the initial data into new columns

```
X_train2["totalSF"] = X_train2['Gr Liv Area'] + X_train2['Total Bsmt SF']
```

```
X_train2["AllfloorSF"] = X_train2["1st Flr SF"] + X_train2["2nd Flr SF"]
X_train2["Qual+Cond"] = X_train2["Overall Qual"] + X_train2["Overall Cond"]
```

Ordinal encode

Using the ordinal encode function I turned some of the categorical values into numerical values. I did this for both the training and test set. I created a dictionary which assigned numerical values to each category. I then checked for categorical values using train_categorical, which shows that there are 0 categorical features in the training & training set after I transformed these columns.

Scaling

At first, in the Modelling file I tried scaling one column using pandas. I did this by subtracting the minimum value, and then dividing the whole column by the maximum value.

I then tried scaling using a Pipeline. I wanted make a pipeline that treats numerical and categorical values separately. For this I needed to use a columntransformer. I did not manage to make this work. Instead I transformed some columns myself and created the list more_colums. However, this list only contained 14 features whereas the actual data set is much bigger. To create a model with more complexity I will have to include more features next time.

Regularization

Ridge Regression

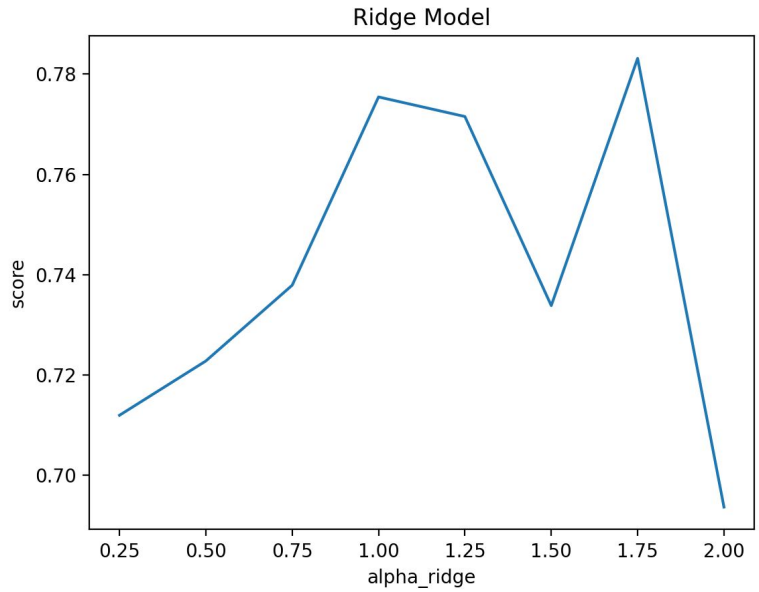
Ridge regression is a technique for analyzing multiple regression data with multicollinearity. When this happens, the least square estimates are unbiased, but their variances are large. The ridge regression reduces standard errors by adding a degree of bias to the regression estimates. The loss function is a least square function and the regularization is given by the L2-norm. Alpha indicates regularization strength, this reduces the variance of the estimates.

I used two different techniques.

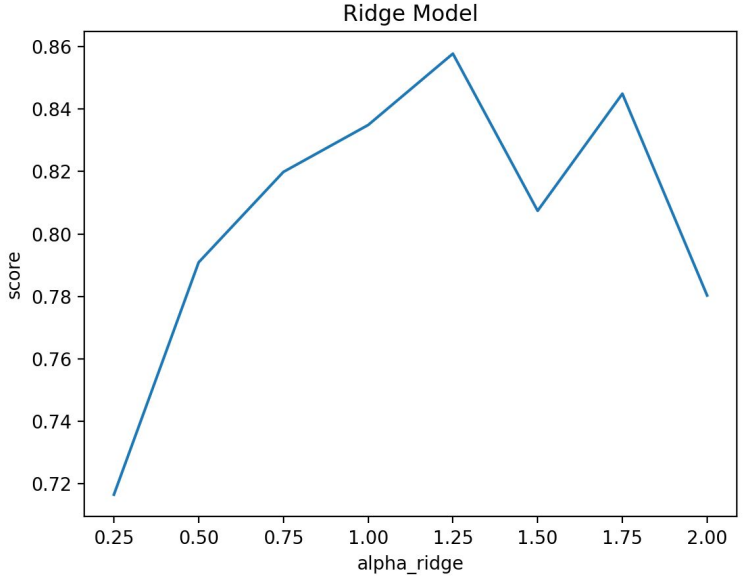
1. First
   I created an empty list that stores cross value scores, and another empty list which stores the corresponding alpha values. Then with a for loop I looped through different values of alpha. Then stored the mean values in the list.

   The cross_val_score splits the data repeatedly into a training and a testing set, trains the estimator using the training set and computes the scores based on the testing set for each iteration of cross-validation. cross_val_score uses the scoring provided in the given estimator which is in this case, Ridge(). The scoring in this case is the R^2 for both Ridge and Lasso regression.

Using only 5 features this resulted in the following curve: We can see the error for each value of alpha.



Using more features this resulted in the following curve:
It is visible that the R^2 scores are higher when using more features.



I will be looking for higher scoring values, because in general, the higher the R^2, the better the model fits the data. R-squared is a statistical measure of how close the data are to the fitted regression line. After looking at the R2 score I need to make a residual plot to check if coefficient estimates and predictions are biased.

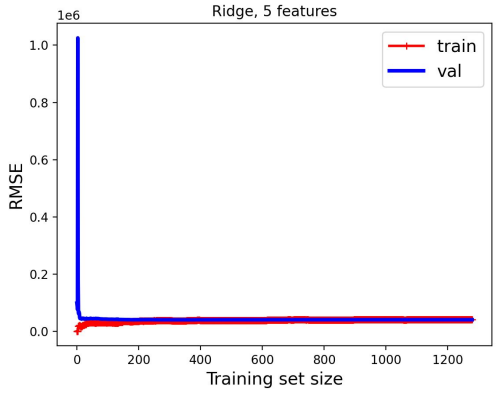
2. GridSearchCV
   In the second method I used a GridSearchCV. I was able to not only look at different values for alpha, but also other hyperparameters such as the “solver”. I was able to find the best parameters and plotted the learning curve of this model.

The hyperparameters of this chosen ridge function are
alpha: regularization strength, large values = stronger regularization

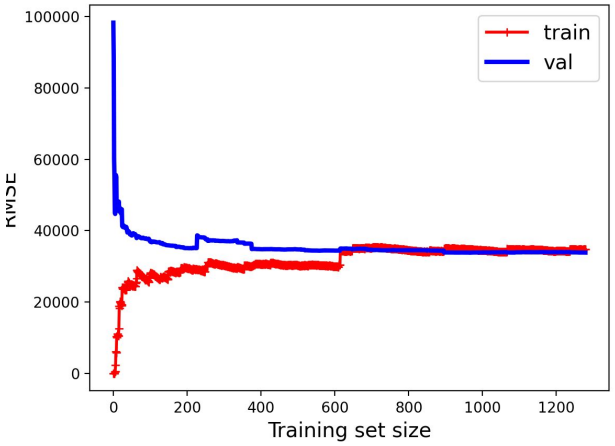
fit_intercept: whether to fit the intercept for this model, if not the y_intercept will be set to 0. If True the y_int is determined by the line of best fit.

solver: the optimal ridge model uses “cholesky” as a solver. It indicates the type of computational routine that is used, some solvers are better for certain data sets (e.g. sparse ones) I did not know the exact differences and tested a few in the GridSearch to observe differences.

Ridge Learning Curve
From the 5 features the learning curve from the ridge regression looks like this:



When using (more_columns) I get the following learning curve:



I can see that the RSME has increased. Indicating that the model is learning from the training data as the training set increases. The errors in training and validation seem to plateau. The validation error starts high since a few points are unlikely to be the same as a few training points, and then drops as the model is not linear. The curves plateau new each other at a high level, it is most likely that the model is underfitting.

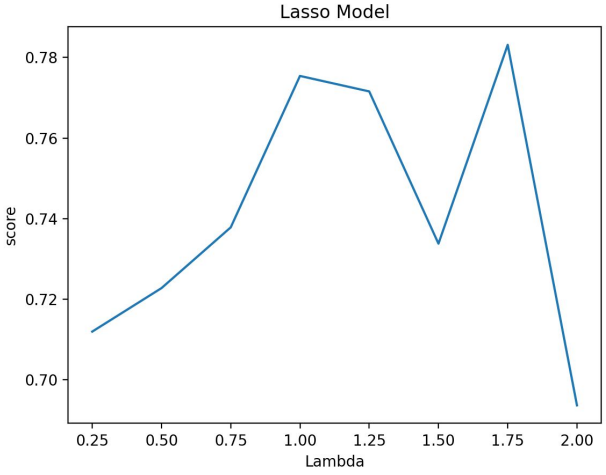
Lasso Regression

This regression method can shrink coefficient estimates towards zero and can thereby avoid the risk of overfitting. The result is a model with less features. Compared to the ridge regression, ridge regression cannot eliminate coefficients.

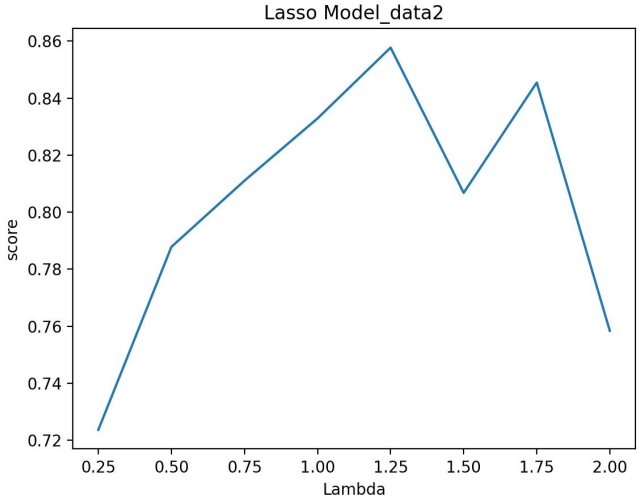
I used two different techniques.

3. First
   I created an empty list that stores cross value scores, and another empty list which stores the corresponding alpha values. Then with a for loop I looped through different values of Lambda. Then stored the mean values in the list.

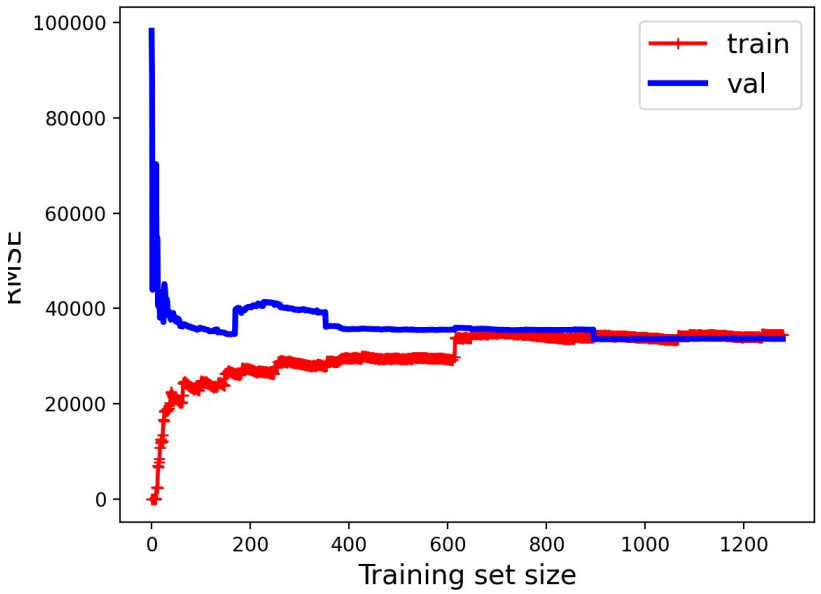
Using only 5 features this resulted in the following curve:



Using more features this resulted in the following curve:



I was curious how the learning curve of the best model according to the first method would compare to the gridsearchCV. I plotted the learning curve of a Lasso Model where alpha = 1.25. The model trained on the data 2. (with more_columns)
The score of this model on the test data is 0.7688.



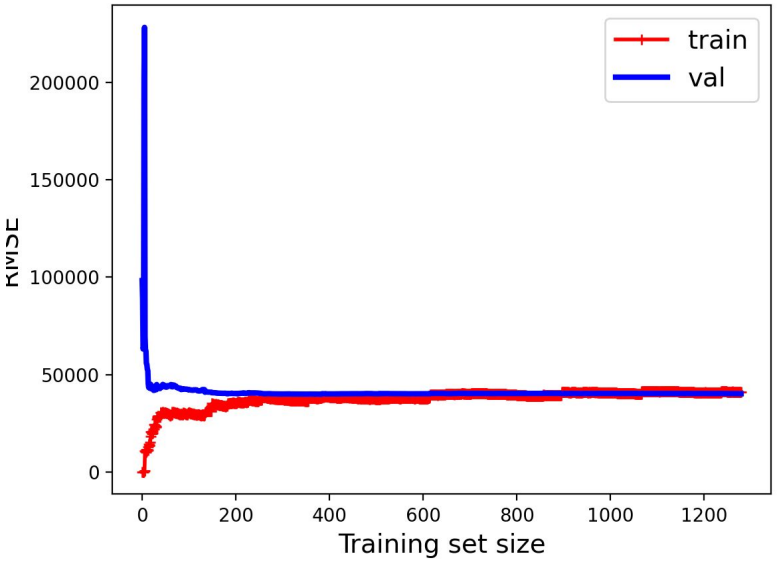
4. GridsearchCV

After performing a GridSearchCV on different hyperparameters values I fitted it on the training data. I then printed the best parameters, and best score for this model. The learning curve is the one from the best model.

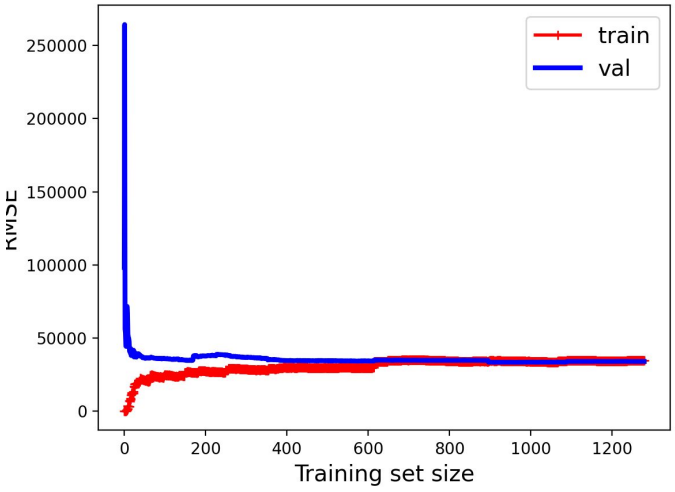
Besides alpha and fit_intercept the lasso parameters included copy_x and selection as well. The optimal model contains copy_x = True and selection = random. This means that a random coefficient is updated every iteration. This leads to a faster convergence.

Lasso learning curve

From the 5 features the learning curve from the lasso regression looks like this:
This model underfits because both curves plateau near each other at a relatively high level, I need a model with more complexity.



When using (more_columns) I get the following learning curve:

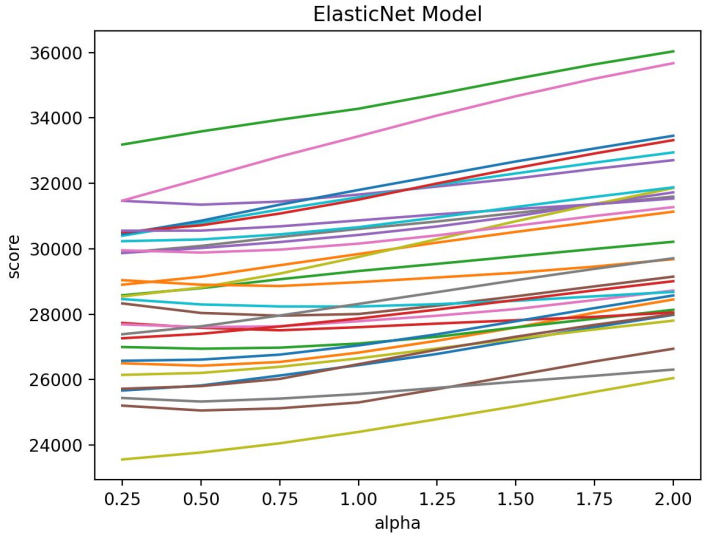


Compared to the initial graph, the learning curve has not changed much. What both curves shows is that adding more training data won't improve the model. However, adding more features is a different thing. This is likely to help improve the model as it would increase the complexity of the current model.

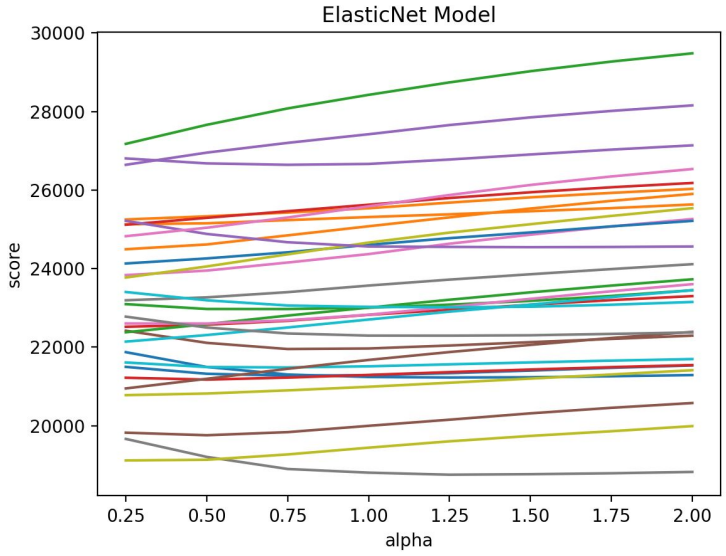
Net Elastic Regression

5. First
   My first method involved only looking at the alpha values, and a fixed l1_ratio of 0.5. I tested the cross validation score for different values of alpha, and using a RepeatedKFold to split. The score was the mean absolute error. From this graph it is not clear to me what exactly happened, but I can see that on average the minimum mean absolute error is smaller for the second data set.

Using only 5 features resulted in the following curve:



Using more features resulted in the following curve:



6. GridsearchCV

   After a while I tried adding more hyperparameters to the ElasticNet gridsearch such as the maximum number of iterations, and the L1_ratio. Elastic net is a combination of L1 and L2. The ration is tunable, in this case a ratio of 0.5 was used.

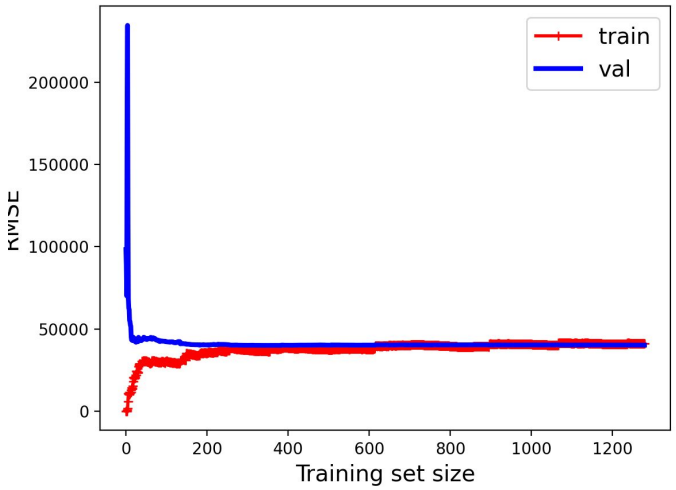
   By default the maximum number of iterations is 1000. When I tried adding more hyperparameters to the gridsearch I got the following error: Objective did not converge. I ended up using only the gridsearch with different values for alpha.

Net Elastic Model Learning Curve

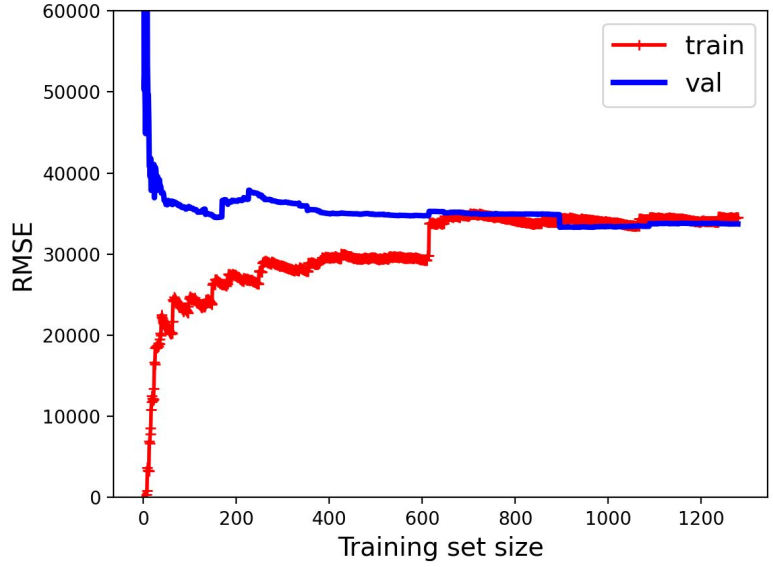
The error is decreasing, showing that parameters are being learned. The learning curve plots training + test error over the training set size. It shows how the model

would benefit from being fed more data. Only later I restricted the range of the learning curve to read RMSE values with more precision.

From the 5 features the learning curve from the Net Elastic regression looks like this:



When using (more_columns) I get the following learning curve:



Overfitting is when the more specialized the model becomes to the training data, the less well it can generalize to new data. None of the learning curves that I showed in this document has a significant gap between validation data and training data. It seems that none of the models is overfitting.

The more narrow the gap between the validation and training error, the lower the variance. But the high RMSE already revealed the low variance in this case. Because these models are overly simplified, they can't even fit the training data well. In other words, all learning curves in this document show underfitting models.

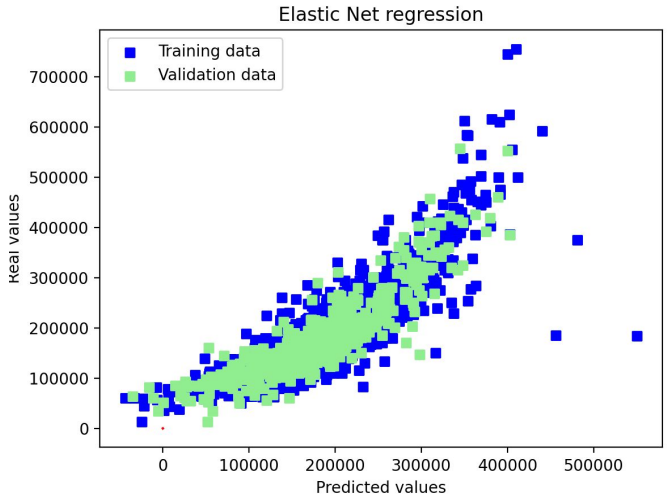
To solve the low variance problem would be to train the current model on more features, this would increase the model's complexity. The second option is to decrease the regularization of the current model. Because the regularization prevents the algorithm from fitting the training data too well. Therefore instead of spending more time comparing regularization methods and adding more hyperparameters I should be adding more features to the training and test data sets.

Plotting Real Values against Predicted values

Some examples from ElasticNet:

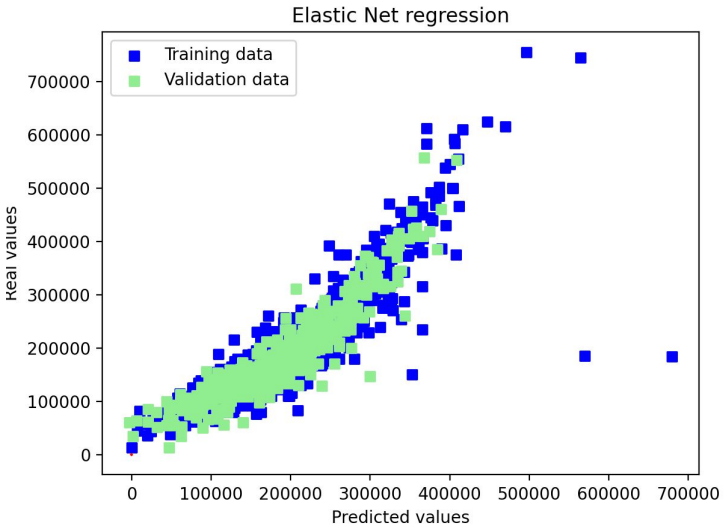
From the GridSearchCV, using only 5 features, I found that the best model is `ElasticNet(alpha = 0.03)`. Its predicted values are visible in the following graph.

r2: 0.7365676423939314
mae: 27742.144981506528
mse: 1684093198.4258037



From the GridSearchCV, using (more_columns), found that the best model is `ElasticNet(alpha = 0.03)`. This resulted in the following graph:

r2: 0.7687999181438743
mae: 23388.14976314864
mse: 1478035913.5366163



Chosen model

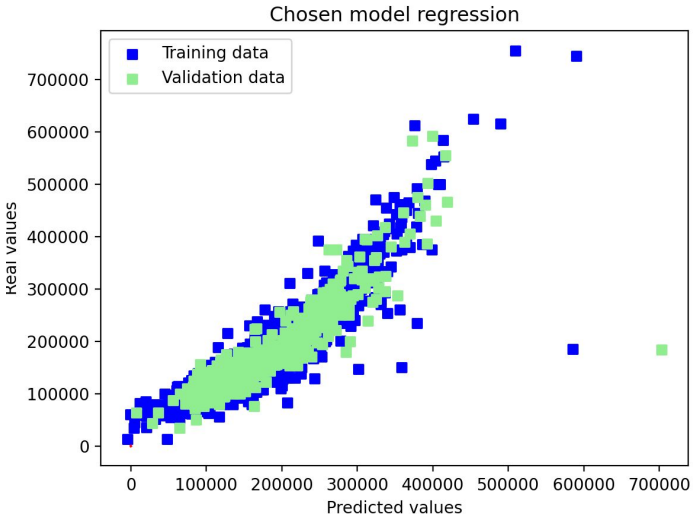
I compared the optimal models from the Gridsearch techniques. The difference is not big, the Lasso model seems to work only a little bit better.

r2 score best elastic net: 0.7687999181438743
mae score best elastic net: 23388.14976314864
mse score best elastic net: 1478035913.5366163

r2 score best Lasso: 0.7690052775120226
mae score best Lasso: 23378.44848323617
mse score best Lasso: 1476723074.376407

r2 score best Ridge: 0.7686605845334239
mae score best Ridge: 23149.05040877751
mse score best Ridge: 1478926657.5128083

For this final chosen model I plotted predicted values against actual values.



I spent most time trying to apply the pipeline, but unfortunately I did not manage to get this to work. I spent a lot of time looking at the different features and fixing errors. I'm slowly learning how to use python and pycharm and feel like I have learnt a lot trying to fix small errors. I should have been more patient with the preprocessing first before starting with the regression, because now the code is not structured at all and there are bits of different steps (data processing etc.) throughout. I tried to structure it a bit more but I was afraid I would run into errors.