Prediction the Sale Price of Houses

Springboard Capstone 2 | Swecha Kranthi

Project Goal

The goal of the project is to use data science tools and statistical analysis to better understand the relationship between house features and how these can be used to predict house prices.

Stakeholders

- Home Buyers - consumers can make better informed decisions about purchasing a home
- Real Estate Agents - Agents can be more informed about shifting trends in the market
- Companies - companies like Zillow use similar analysis to calculate an estimated value of a house based on its features and surrounding facts.
- Banks - Banks can use similar analysis to assess the risk that comes with a give mortgage on a house

Why it matters?

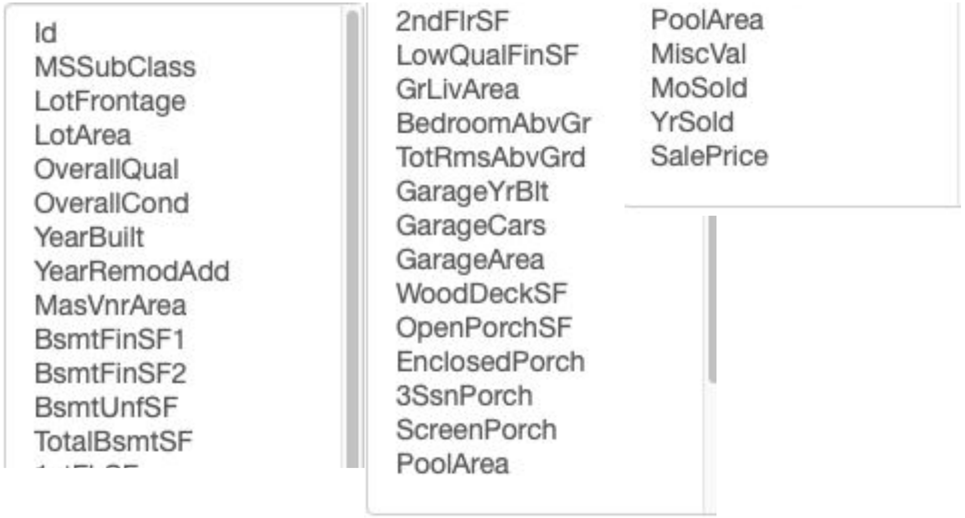
Weather you are buying a home or not, Home prices are a key indicator on how well the broader economy is performing. In addition, such analysis can help the key stakeholders to make a more informed decision. This information can help people plan their finances, assess the risk that comes with their mortgage and secure a better financial position for the future.

Data Information

- The Ames housing data set was compiled by Dr. Dean DeCock of Truman State University. It can be found on Kaggle competition site: https://www.kaggle.com/c/house-prices-advanced-regression-techniques/overview
- The Data set is comprised of 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa
- There are 2 data sets: Train and Test
  - Train set contains 1460 rows of data with 80 columns. The final column being the Sale Price
  - Test set contain 1459 rows of data with 79 feature columns which we use to predict the final Sale Price.

Features

- 48 categorical columns
- 32 numerical columns as seen on the right:
- 1 boolean column



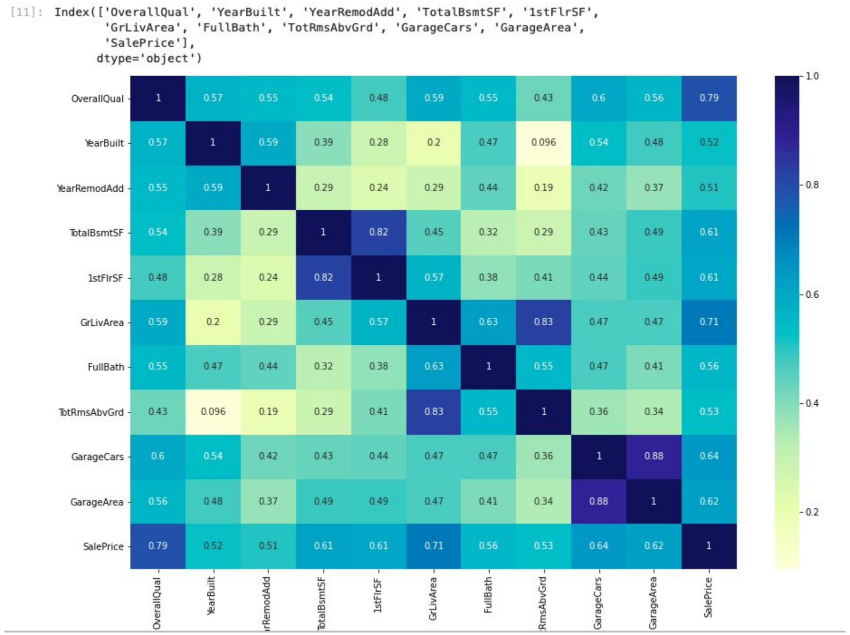
Id
MSSubClass
LotFrontage
LotArea
OverallQual
OverallCond
YearBuilt
YearRemodAdd
MasVnrArea
BsmtFinSF1
BsmtFinSF2
BsmtUnfSF
TotalBsmtSF

2ndFlrSF
LowQualFinSF
GrLivArea
BedroomAbvGr
TotRmsAbvGrd
GarageYrBlt
GarageCars
GarageArea
WoodDeckSF
OpenPorchSF
EnclosedPorch
3SsnPorch
ScreenPorch
PoolArea

PoolArea
MiscVal
MoSold
YrSold
SalePrice

Exploratory Data Analysis

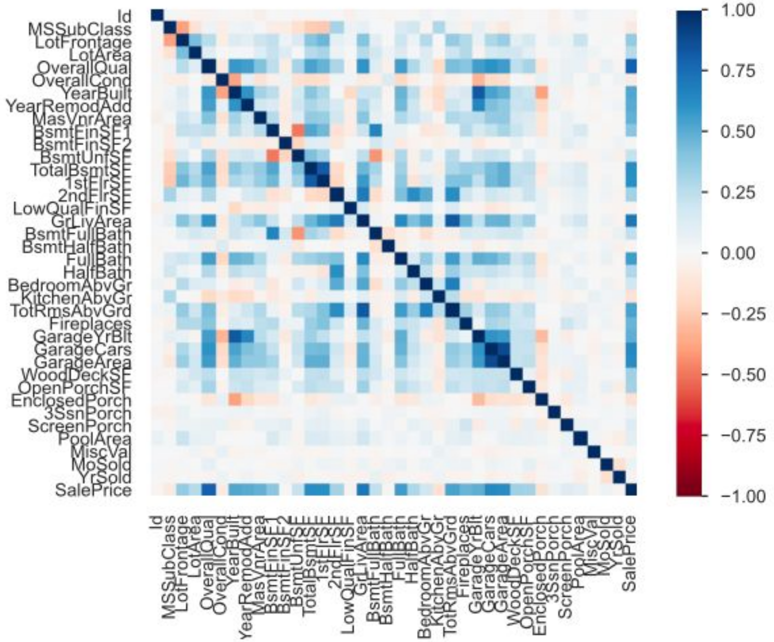
Pearson Correlations of features of correlations > 0.5 with Sale Price



```
[11]: Index(['OverallQual', 'YearBuilt', 'YearRemodAdd', 'TotalBsmtSF', '1stFlrSF',
       'GrLivArea', 'FullBath', 'TotRmsAbvGrd', 'GarageCars', 'GarageArea',
       'SalePrice'],
      dtype='object')
```

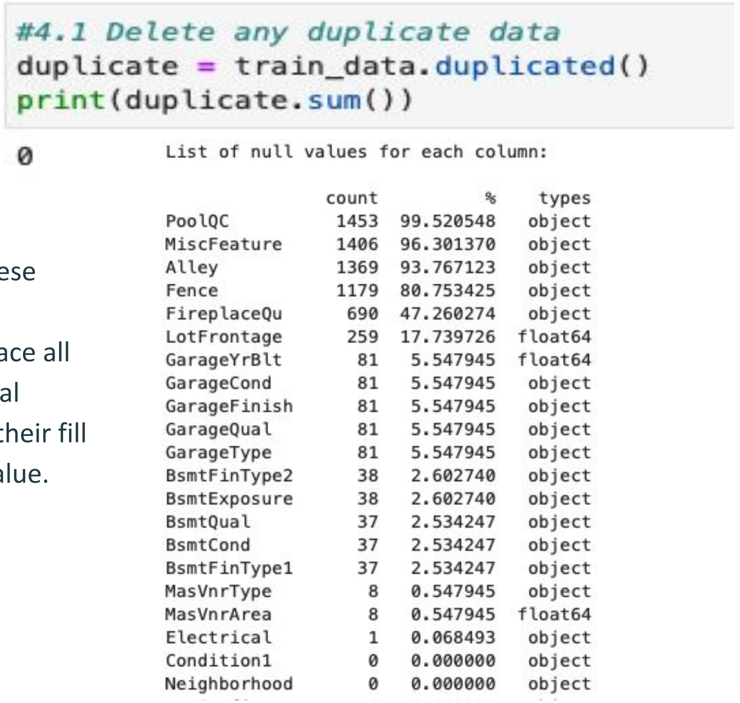
	OverallQual	YearBuilt	rRemodAdd	TotalBsmtSF	1stFlrSF	GrLivArea	FullBath	RmsAbvGrd	GarageCars	GarageArea	SalePrice
OverallQual	1	0.57	0.55	0.54	0.48	0.59	0.55	0.43	0.6	0.56	0.79
YearBuilt	0.57	1	0.59	0.39	0.28	0.2	0.47	0.096	0.54	0.48	0.52
YearRemodAdd	0.55	0.59	1	0.29	0.24	0.29	0.44	0.19	0.42	0.37	0.51
TotalBsmtSF	0.54	0.39	0.29	1	0.82	0.45	0.32	0.29	0.43	0.49	0.61
1stFlrSF	0.48	0.28	0.24	0.82	1	0.57	0.38	0.41	0.44	0.49	0.61
GrLivArea	0.59	0.2	0.29	0.45	0.57	1	0.63	0.83	0.47	0.47	0.71
FullBath	0.55	0.47	0.44	0.32	0.38	0.63	1	0.55	0.47	0.41	0.56
TotRmsAbvGrd	0.43	0.096	0.19	0.29	0.41	0.83	0.55	1	0.36	0.34	0.53
GarageCars	0.6	0.54	0.42	0.43	0.44	0.47	0.47	0.36	1	0.88	0.64
GarageArea	0.56	0.48	0.37	0.49	0.49	0.47	0.41	0.34	0.88	1	0.62
SalePrice	0.79	0.52	0.51	0.61	0.61	0.71	0.56	0.53	0.64	0.62	1

Pearson Correlations of all features



Data Cleaning 1

- Identify and Delete that Contain Duplicate Data
  - No duplicates in data
- Address Missing Values
  - We had 19 columns with some missing values. 4 of these columns had more than 50% missing values: Pool, Miscellaneous Features, Alley, Fence. I decided to replace all null values with their respective fill values. All numerical columns were either filled with mean, median or 0 as their fill value. All categorical columns had 'None' as their fill value.



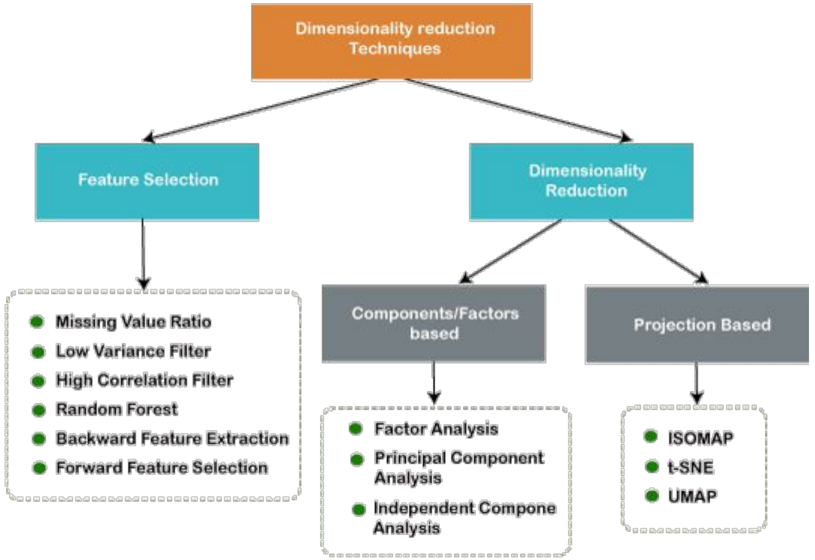
```python
#4.1 Delete any duplicate data
duplicate = train_data.duplicated()
print(duplicate.sum())
```

```
0          List of null values for each column:

                  count          %    types
           PoolQC          1453  99.520548   object
           MiscFeature     1406  96.301370   object
           Alley           1369  93.767123   object
           Fence           1179  80.753425   object
           FireplaceQu      690  47.260274   object
           LotFrontage      259  17.739726  float64
           GarageYrBlt       81   5.547945  float64
           GarageCond        81   5.547945   object
           GarageFinish      81   5.547945   object
           GarageQual        81   5.547945   object
           GarageType        81   5.547945   object
           BsmtFinType2      38   2.602740   object
           BsmtExposure      38   2.602740   object
           BsmtQual          37   2.534247   object
           BsmtCond          37   2.534247   object
           BsmtFinType1      37   2.534247   object
           MasVnrType         8   0.547945   object
           MasVnrArea         8   0.547945  float64
           Electrical         1   0.068493   object
           Condition1         0   0.000000   object
           Neighborhood       0   0.000000   object
```

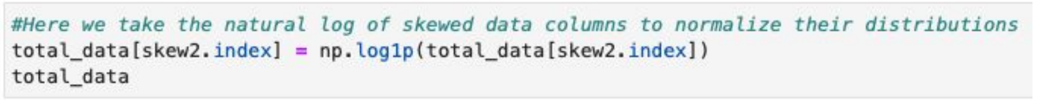
Data Cleaning 2

- Dimensionality Reduction - To reduce the number of variable used to train data
  - There are many ways of doing this, but I used my two favorites: High Correlation Filtering, and Random Forest.
    - High Correlation Filtering is a method where we look at the correlations between each feature and our target value. We decide to keep the once that correlate highly with sale price.
    - Random Forest is a regression algorithm that has built in feature importance, thus we can avoid manually selecting for features and use everything to build our model.



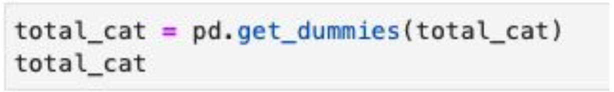
Preprocessing

- **Numerical Data** needs to be adjusted for any skewness in it's distribution. This is to prevent misleading the models and generating wrong predictions.
  - We adjust the skewness of the sale price(right skewed) column by taking the natural log of the data
  - We also adjust the skewness of all numerical columns that have skewness > 0.5 using natural log



```
#Here we take the natural log of skewed data columns to normalize their distributions
total_data[skew2.index] = np.log1p(total_data[skew2.index])
total_data
```

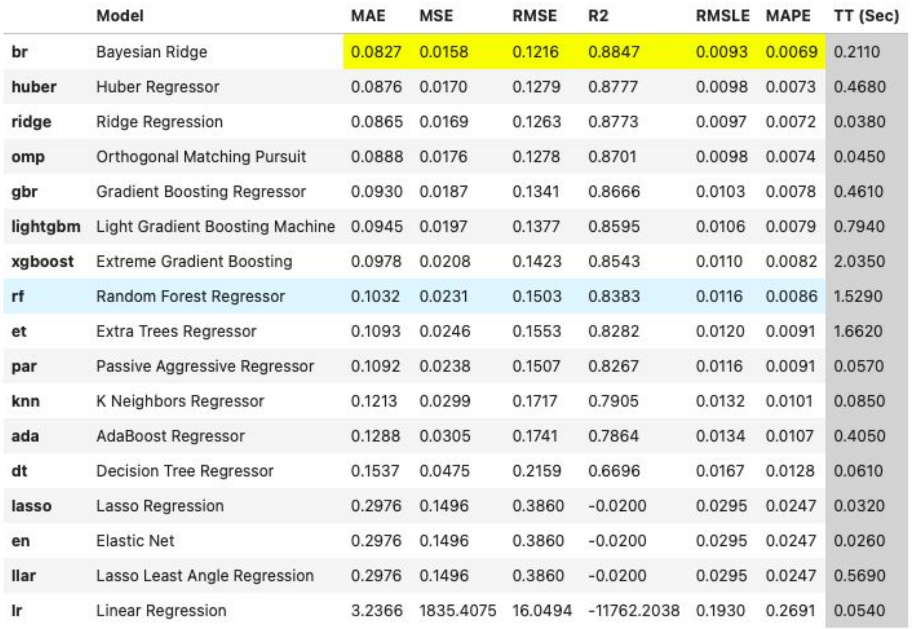
- **Categorical Data** is better designed for classification problems. It is hard to use categorical data to creation regressions.
  - To solve this problem, we encode each of our categorical features into a series of numerical features to represent its respective category.



```
total_cat = pd.get_dummies(total_cat)
total_cat
```

Modeling 1 - all models

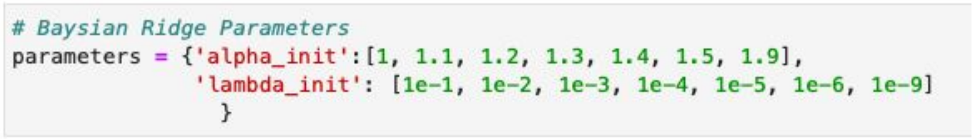
- For Modeling, I used the automation tool - **Pycaret** to test every regression model and find the best one.
- We optimized for lowest Mean Squared Error and High R² value
- The result ruled that **Bayesian Ridge Regression** yielded the best model with 88.47% accuracy and the lowest root mean squared error of 0.1216



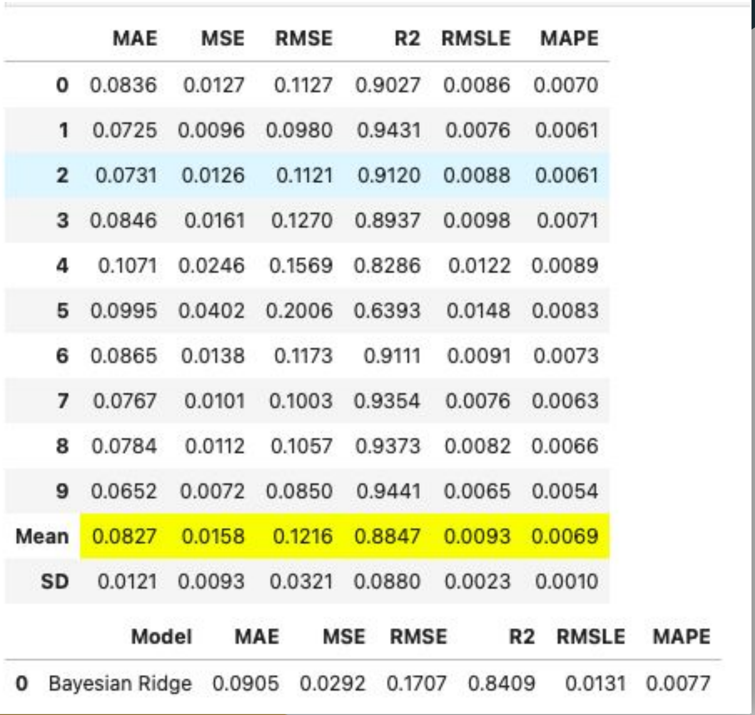
	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
br	Bayesian Ridge	0.0827	0.0158	0.1216	0.8847	0.0093	0.0069	0.2110
huber	Huber Regressor	0.0876	0.0170	0.1279	0.8777	0.0098	0.0073	0.4680
ridge	Ridge Regression	0.0865	0.0169	0.1263	0.8773	0.0097	0.0072	0.0380
omp	Orthogonal Matching Pursuit	0.0888	0.0176	0.1278	0.8701	0.0098	0.0074	0.0450
gbr	Gradient Boosting Regressor	0.0930	0.0187	0.1341	0.8666	0.0103	0.0078	0.4610
lightgbm	Light Gradient Boosting Machine	0.0945	0.0197	0.1377	0.8595	0.0106	0.0079	0.7940
xgboost	Extreme Gradient Boosting	0.0978	0.0208	0.1423	0.8543	0.0110	0.0082	2.0350
rf	Random Forest Regressor	0.1032	0.0231	0.1503	0.8383	0.0116	0.0086	1.5290
et	Extra Trees Regressor	0.1093	0.0246	0.1553	0.8282	0.0120	0.0091	1.6620
par	Passive Aggressive Regressor	0.1092	0.0238	0.1507	0.8267	0.0116	0.0091	0.0570
knn	K Neighbors Regressor	0.1213	0.0299	0.1717	0.7905	0.0132	0.0101	0.0850
ada	AdaBoost Regressor	0.1288	0.0305	0.1741	0.7864	0.0134	0.0107	0.4050
dt	Decision Tree Regressor	0.1537	0.0475	0.2159	0.6696	0.0167	0.0128	0.0610
lasso	Lasso Regression	0.2976	0.1496	0.3860	-0.0200	0.0295	0.0247	0.0320
en	Elastic Net	0.2976	0.1496	0.3860	-0.0200	0.0295	0.0247	0.0260
llar	Lasso Least Angle Regression	0.2976	0.1496	0.3860	-0.0200	0.0295	0.0247	0.5690
lr	Linear Regression	3.2366	1835.4075	16.0494	-11762.2038	0.1930	0.2691	0.0540

Modeling 2 - Bayesian Ridge Optimization

- To optimize the BR model, we tested out and tuned the following basic parameters of the BR model:
  - alpha_init : Initial value for alpha (precision of the noise). If not set, alpha_init is 1/Var(y).
  - lambda_init : Initial value for lambda (precision of the weights). If not set, lambda_init is 1.



```
# Baysian Ridge Parameters
parameters = {'alpha_init':[1, 1.1, 1.2, 1.3, 1.4, 1.5, 1.9],
              'lambda_init': [1e-1, 1e-2, 1e-3, 1e-4, 1e-5, 1e-6, 1e-9]
               }
```

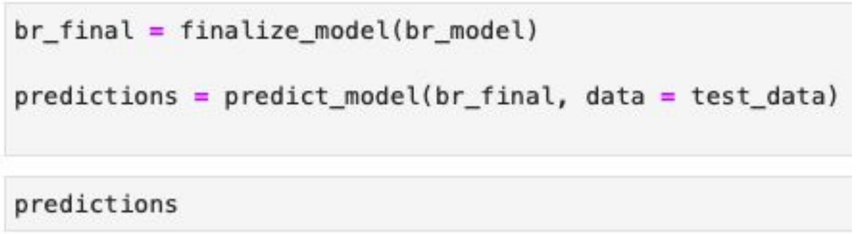


	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	0.0836	0.0127	0.1127	0.9027	0.0086	0.0070
1	0.0725	0.0096	0.0980	0.9431	0.0076	0.0061
2	0.0731	0.0126	0.1121	0.9120	0.0088	0.0061
3	0.0846	0.0161	0.1270	0.8937	0.0098	0.0071
4	0.1071	0.0246	0.1569	0.8286	0.0122	0.0089
5	0.0995	0.0402	0.2006	0.6393	0.0148	0.0083
6	0.0865	0.0138	0.1173	0.9111	0.0091	0.0073
7	0.0767	0.0101	0.1003	0.9354	0.0076	0.0063
8	0.0784	0.0112	0.1057	0.9373	0.0082	0.0066
9	0.0652	0.0072	0.0850	0.9441	0.0065	0.0054
Mean	0.0827	0.0158	0.1216	0.8847	0.0093	0.0069
SD	0.0121	0.0093	0.0321	0.0880	0.0023	0.0010

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	Bayesian Ridge	0.0905	0.0292	0.1707	0.8409	0.0131	0.0077

Predictions

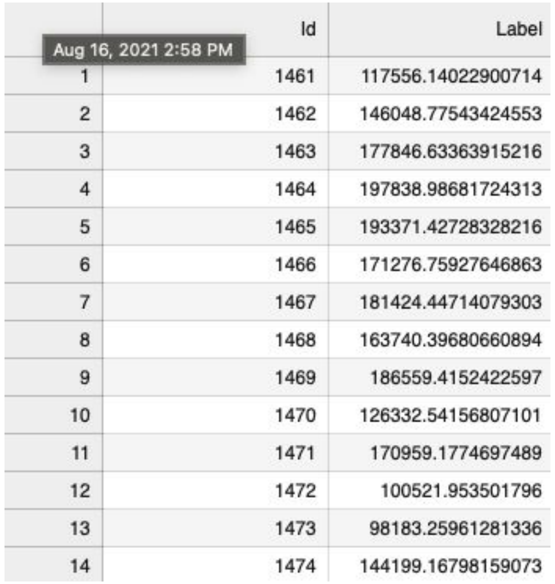
- After finding the optimal Bayesian Ridge Model. WE finalized it and used it to predict the sale price values on the test data. The predictions can be seen on the right.



```
br_final = finalize_model(br_model)

predictions = predict_model(br_final, data = test_data)
```

```
predictions
```



Aug 16, 2021 2:58 PM

	Id	Label
1	1461	117556.14022900714
2	1462	146048.77543424553
3	1463	177846.63363915216
4	1464	197838.98681724313
5	1465	193371.42728328216
6	1466	171276.75927646863
7	1467	181424.44714079303
8	1468	163740.39680660894
9	1469	186559.4152422597
10	1470	126332.54156807101
11	1471	170959.1774697489
12	1472	100521.953501796
13	1473	98183.25961281336
14	1474	144199.16798159073

Limitations

- Time
  - The data is sampled between 2006 - 2010; thus, it is not wise to use this model to predict recent home evaluations.
- Location
  - The data represents only 1 city - Ames, Iowa. We cannot use this model to represent housing in other parts of the county.
- Data variance
  - There are a few data columns that have less than 5 unique values. This lack of variance wasn't properly addressing in my cleaning process.
- Low sample size
  - There were 1460 and 1459 samples in Train and Test Respectively. This data is also skewed right. This low sample size means that we cannot assume our data follows a gaussian distributions.
- Outliers
  - I just normalize the distribution of the data. Instead, I should remove the outliers first. This may have led to a decreased accuracy of my model.

Future Improvements

- In the future, I would love to spend more time creating a cleaner data set. This is my second time properly cleaning large datasets, I could improve my process by more closely understanding the data and optimizing for data skews better.
- This modeling could also be improved my better optimizing for all the parameters that Bayesian Ridge Regression offers.
- I would love to expand this project to larger data sets of sales from Zillow and Trulia detailing all the sales from the last 3 years and include data from the covid years.

Credits

- Thank you to Raghunandhan Patthar for being an awesome Springboard Mentor.
- Thank you Moez Ali for making such an amazing tool like PyCaret.
- Thank you Dr. Dean DeCock from Truman State University for creating the data set that I was able to work on and learn from.