PREDICTING HOUSING PRICES USING REGRESSION TECHNIQUES

Sundar, Shreecharran
ssundar48@gatech.edu

Jeyaprakash, Sarni
jayaprakashsarni@gatech.edu

December 6, 2021

ABSTRACT

How can we leverage regression techniques for estimation sales prices of houses ? This type of price modelling comes under hedonic pricing method in econometrics. We have used the housing data collected on residential properties sold in Ames, Iowa between 2006 and 2010 for answering this question. We combine insights gained through the application machine learning and regression techniques with traditional domain knowledge (e.g.: a larger/recently constructed house is expected to cost more than a smaller/older house) to develop a useful estimation framework for accurate prediction of housing prices.

1 INTRODUCTION

Determining the sale price of a house is often complicated due to the numerous variables influencing real estate valuation.This is a problem for real estate agents and prospective home-buyers who want to maximize the value of the home that they're trying to buy or sell. We seek to answer the following questions using our model.

1. Can a regression based framework be constructed to accurately predict housing prices? If we are able to develop such a model, a buyer can use this tool to assess whether the asking price of a house is fair.There are various websites, for example, Zillow.com, that provide similar tools for estimating the market value of a house

2.What are the features that drive up the housing sales price? The size of the house? No of Bedrooms? Real estate agents can leverage this tool to market houses that will likely fetch them a higher price

2 DATASET DESCRIPTION

We use the Ames housing dataset, which describes the sale of individual residential properties in Ames, Iowa from 2006 to 2010. The data set contains 2930 observations and 82 explanatory variables (23 nominal, 23 ordinal, 14 discrete, and 20 continuous) involved in assessing housing price. These variables are both quantitative (e.g garage area, number of fireplaces) and qualitative (e.g. heating quality, exterior quality).

- Data Source: `http://jse.amstat.org/v19n3/decock/AmesHousing.txt`.
- Variable Description: `http://jse.amstat.org/v19n3/decock/DataDocumentation.txt`

	Order	PID	MS_Zoning	Lot_Frontage	Bldg_Type	House_Style	Overall_Qual	Overall_Cond	Year_Built	Sale_Condition	SalePrice
0	1	0526301100	RL	141	1Fam	1Story	6	5	1960	Normal	215000
1	2	0526350040	RH	80	1Fam	1Story	5	6	1961	Normal	105000
2	3	0526351010	RL	81	1Fam	1Story	6	6	1958	Normal	172000
3	4	0526353030	RL	93	1Fam	1Story	7	5	1968	Normal	244000
4	5	0527105010	RL	74	1Fam	2Story	5	5	1997	Normal	189900

Figure 1: The first five rows of data.

3 METHODOLOGY

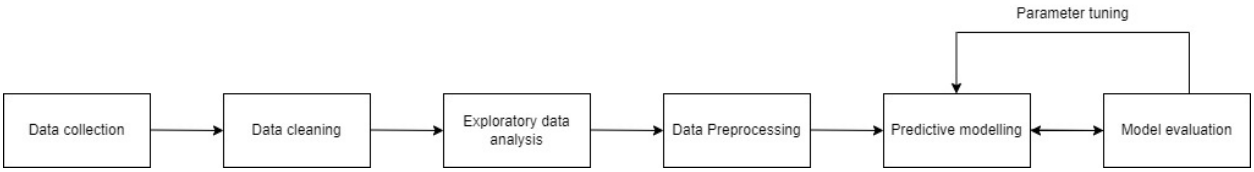


Figure 2: flowchart of the methodology

4 DATA CLEANING

Since the dataset was obtained from Assesor's office, there were a lot of missing values.

- Most of the features with missing values meant the absence of that feature in the house. Hence, these null values were imputed with NONE (or 0 depending on the type of variable).For eg: A missing values in Pool_QC meant the absence of a pool in the house and was imputed to None. The corresponding pool area was set to 0.

- A systematic approach was adopted for imputing the null values in other missing fields. Lot frontage was found to be related to the lot shape, so the missing values in Lot frontage column were filled with the average "LotFrontage" values obtained by grouping different lot shapes.

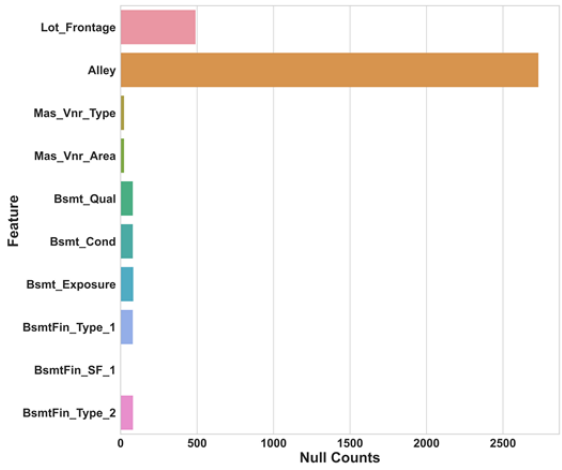


Figure 3: Missing Value counts

	Lot_Shape	Lot_Frontage
0	IR1	74.814985
1	IR2	67.437500
2	IR3	117.636364
3	Reg	66.848871

Figure 4: Imputation strategy for Lot_Frontage

5 EXPLORATORY DATA ANALYSIS (DATA VISUALIZATION)

Exploratory Data Analysis allowed us to discover patterns in data,to spot anomalies,to test hypothesis and to check the underlying model assumptions with the help of summary statistics and graphical representations. Visualization of categorical data will help us answer question like: How many sub-categories were found in each category? What was the frequency of occurrence of the different sub-categories? What was the distribution of sales price among different categories.These visualizations helped in setting up our predictive pipeline.

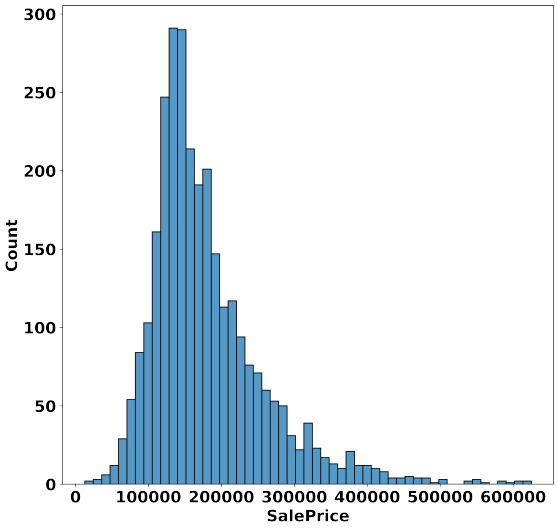


Figure 5: Distribution of Sale Price

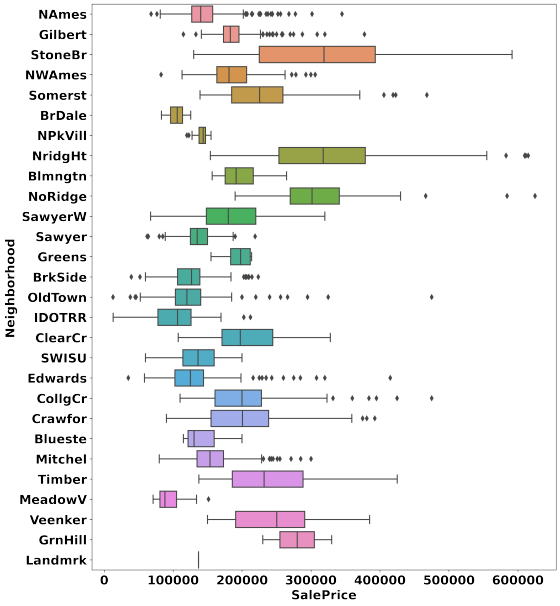


Figure 6: Sales Price by Neighborhood

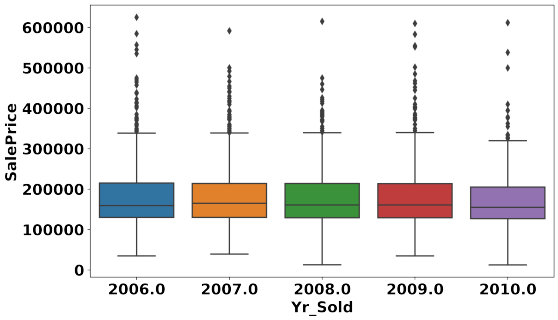


Figure 7: Box plot of Sales Price by year sold

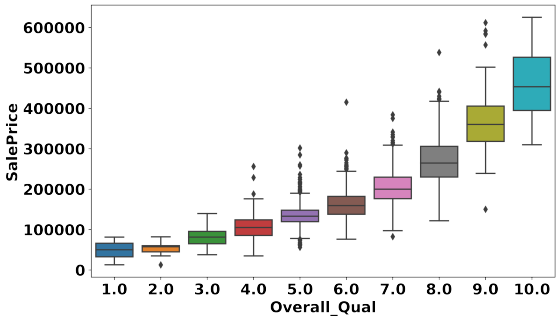


Figure 8: Sales Price dist based on overall quality

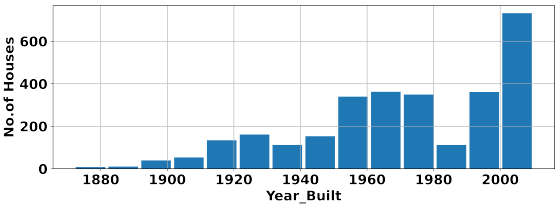


Figure 9: Housing frequency by year of construction

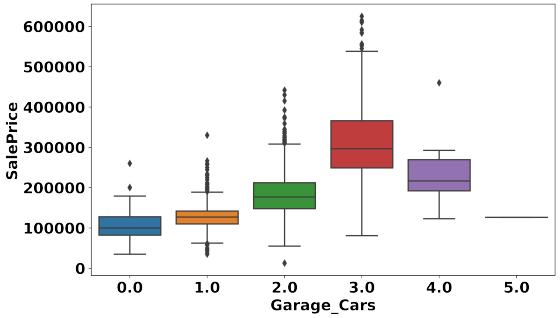


Figure 10: Sales Price dist based on overall cars.

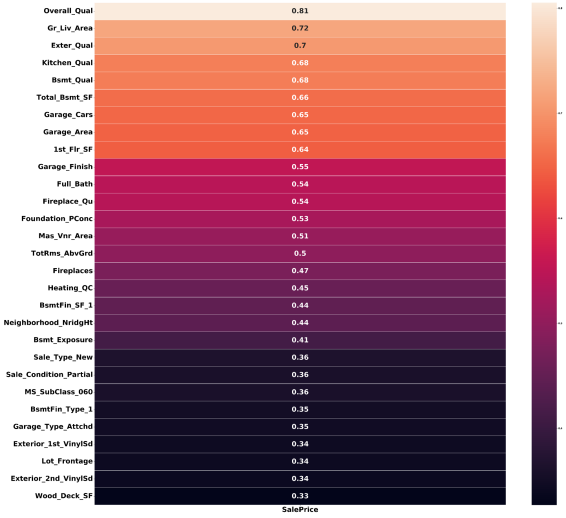


Figure 11: feature correlation with target variable

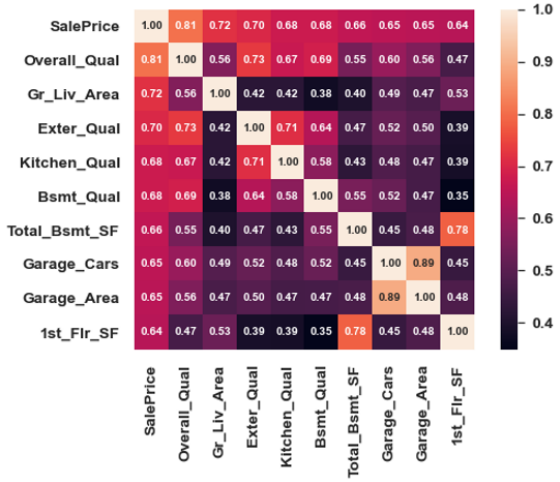


Figure 12: Correlation Matrix

Some key insights from our EDA were:

- The distribution of Sale price was skewed to the right with heavy tails
- Most of the houses were constructed in the 1950's and 2000's
- There was no great effect of recession on the housing prices. (Housing prices were similar across all four years)
- Higher the overall condition of the house, higher was its sale price
- Houses with garage area that accommodated 3 cars were fetching the highest prices.
- Overall_Qual (overall quality) , Gr_Liv_Area (living area size), Kitchen_ Qual (Kitchen quality) are highly correlated to saleprice
- Many of the features were highly correlated (Multi-Collinearity)

Using our exploratory data analysis, we were able to drop 14 features from the dataset that were distributed entirely in just one category and hence did not add any additional discriminative information to our model.

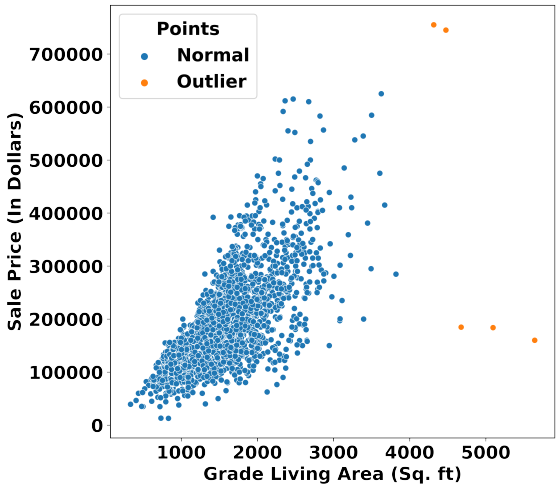
6 DATA PRE-PROCESSING ROUTINES

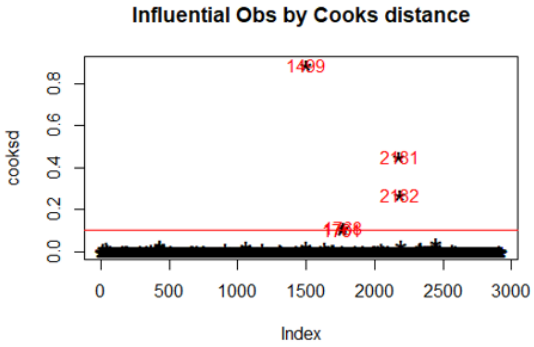
6.1 CATEGORICAL DATA ENCODING

- **Label Encoding for ordinal variables:** Using this technique, each label was assigned a unique integer based on its ordering.For eg: Garage condition was encoded using (Excel':2,'Good':1,'Typical':0,'Fair':-1,'Poor':-2)
- **One Hot Encoding of Nominal Variables:** It is the process of creating dummy variable for each category in a feature.

6.2 OUTLIER DETECTION

. We removed the 5 obvious outlier data points (using scatter plots of Gr_Liv_Area vs SalePrice).





6.3 Box Cox Transformation

Based on the result of box-cox routine, we log transformed the target variable ("SalePrice") for removing the skewness and getting a better linear model.

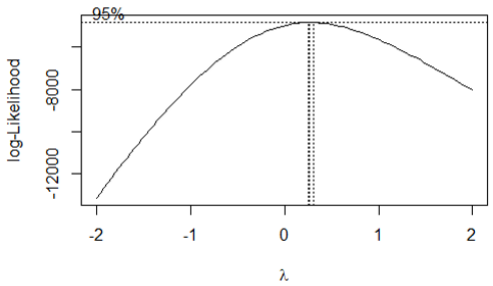


Figure 13: Box-Cox Transformation

7 Model Selection

We decided to use regularized regression techniques to limit the effect of overfitting. Ridge and Lasso regression are powerful techniques generally used for creating parsimonious models in presence of a large number of features (201 features in our case after encoding).They are useful in scenarios where independent variables are highly correlated. In particular, we were interested in LASSO which uses shrinkage to generate interpretable and sparse models.

Technique	Objective
Ordinary Least Squares	$\underset{\theta}{argmin}\ SSE$
Ridge Regression	$\underset{\theta}{argmin}\ SSE + \lambda \sum_{i=1}^{K} \theta_i^2$
Lasso Regression	$\underset{\theta}{argmin}\ SSE + \lambda \sum_{i=1}^{K} \lvert\theta_i\rvert$

Figure 14: Objective function for different regression techniques. We choose the optimal λ using grid search over the parameter space

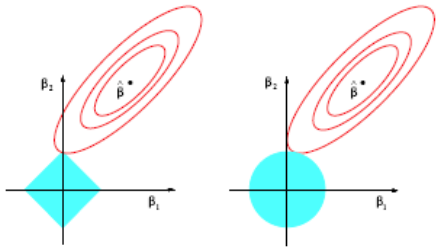


Figure *Estimation picture for the lasso (left) and ridge regression (right). Shown are contours of the error and constraint functions. The solid blue areas are the constraint regions* $|\beta_1| + |\beta_2| \leq t$ *and* $\beta_1^2 + \beta_2^2 \leq t^2$*, respectively, while the red ellipses are the contours of the least squares error function.*

8 Results

8.1 LASSO MODEL

For choosing the LASSO parameter, the grid search technique with 10-fold cross validation method is used. (More details about Cross validation and Grid search in appendix). The best regularization coefficient was $\lambda = 0.001$. The model was highly accurate and generalized well to the test dataset. The lasso model chose 134 features from the 204 predictors. Performance of the model is summarized in the following plots and table.

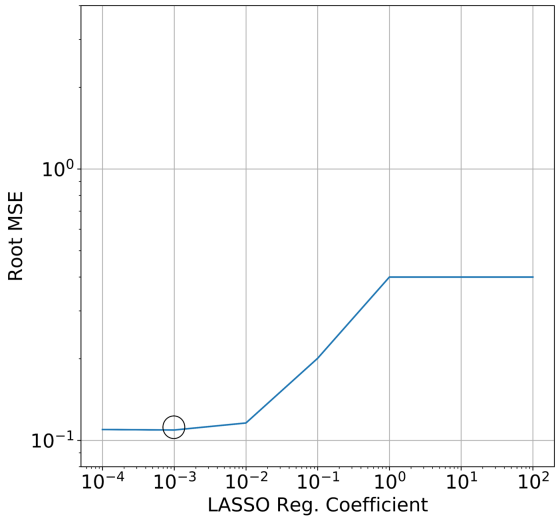


Figure 15: Hyperparameter tuning for LASSO

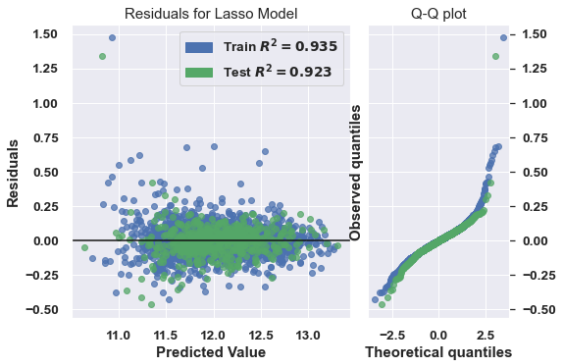


Figure 16: From the residual and QQ plot, the assumption of normality and independence were roughly accurate.

R^2 Train	0.935
R^2 Test	0.923
MSE train	0.01
MSE test	0.0113

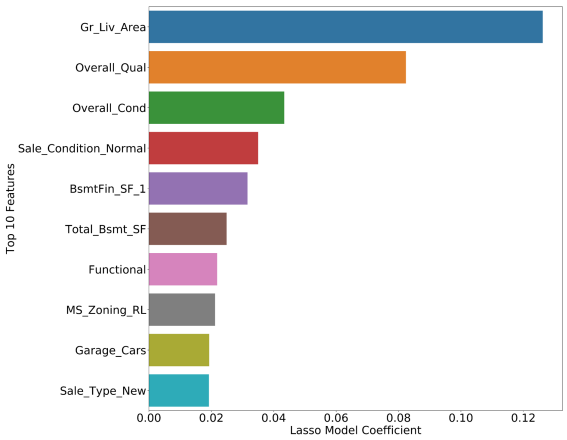


Figure 17: Top 10 feature that positively influence sale price in LASSO model

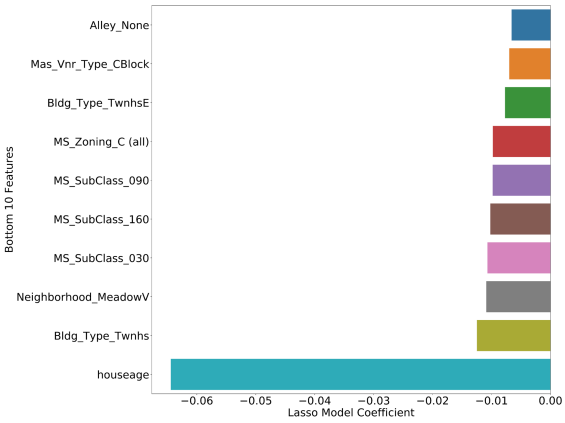


Figure 18: Bottom 10 features that negatively influence sale price in LASSO model

The top predictive features from our model were easily explainable. Our top features were living area and overall quality, followed by different housing features such as basement square footage, normal sale condition . Home functionality and the number of cars that a garage can fit were also important sale price indicators. Certain neighborhoods like Northridge Heights and Stone Brook (high-end neighborhoods) were also strong positive predictors.

On the other end of the spectrum, house age was the biggest negative influence on house price. We also see that houses that were in MS SubClass_030 (1-STORY BUILT 1945 & OLDER) fetched lower prices as one would expect. The Old Town neighborhood also had lower prices in addition to houses not having any alley access

8.2 RIDGE MODEL

For choosing the RIDGE parameter, a grid search technique with 10-fold cross validation method was employed. The best regularization coefficient was $\lambda = 100$. The model has performance very similar to the lasso model. Performance of the model is summarized in the following plots and table.

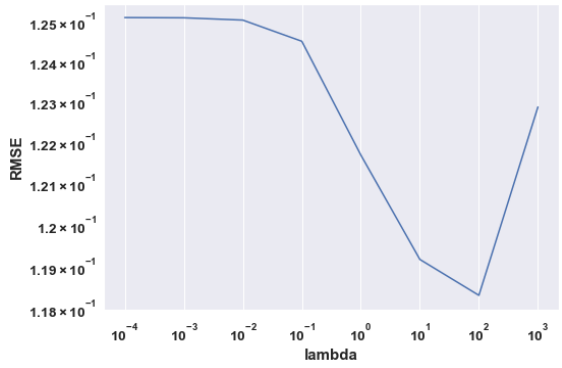


Figure 19: Hyperparameter tuning for Ridge

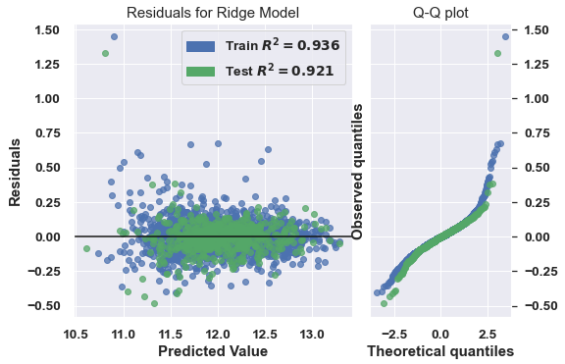


Figure 20: From the residual and QQ plot, the assumption of normality and independence were roughly accurate

R^2 Train	0.936
R^2 Test	0.921
MSE train	0.01
MSE test	0.012

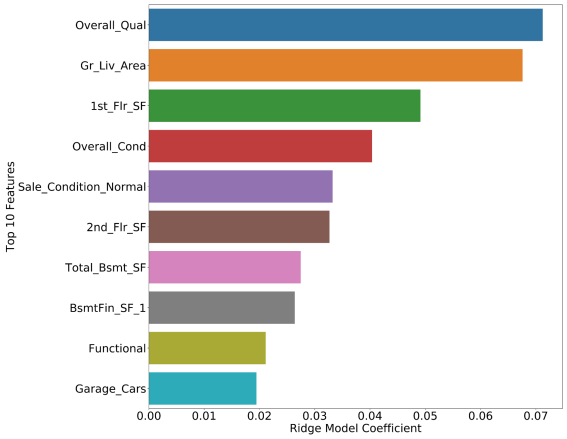


Figure 21: Top 10 feature that positively influence sale price in Ridge model

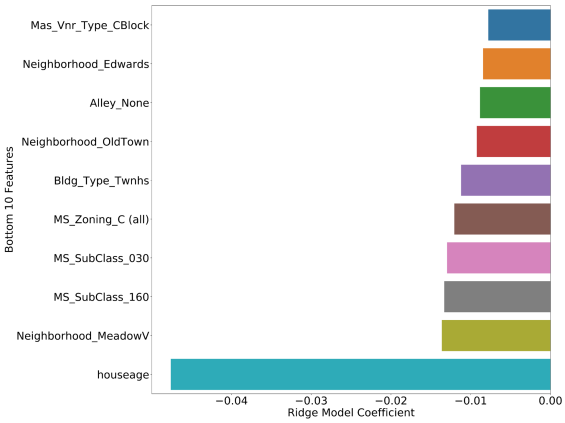


Figure 22: Bottom 10 features that negatively influence sale price in Ridge model

The top and bottom 10 most influential features were largely similar in both the models. The key differenc between the ridge and LASSO model arose in the weights assigned to features by the two models. (For eg: In the lasso model, living area has the biggest positive influence on sale price whereas the ridge model places highest emphasis on the overall condition of the house)

8.3 Comparison between models & Effect of Regularization

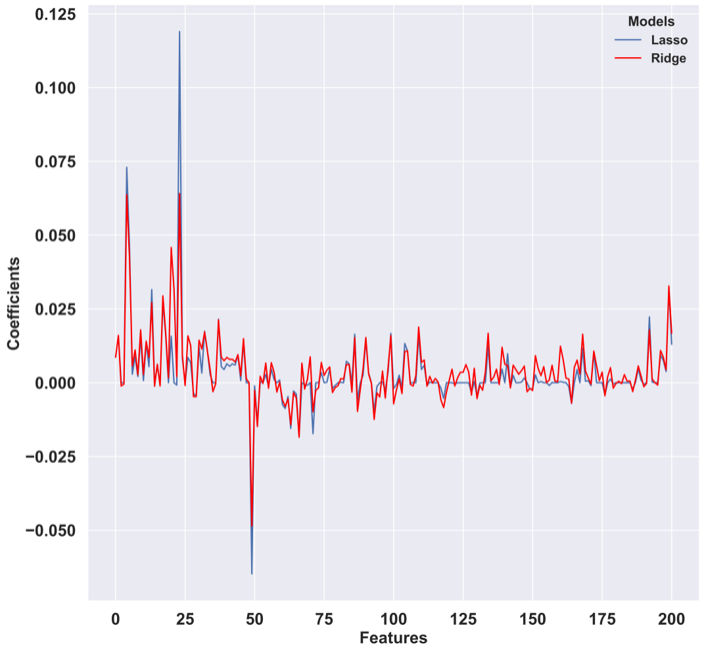


Figure 23: Comparison of regression coefficient in the two models

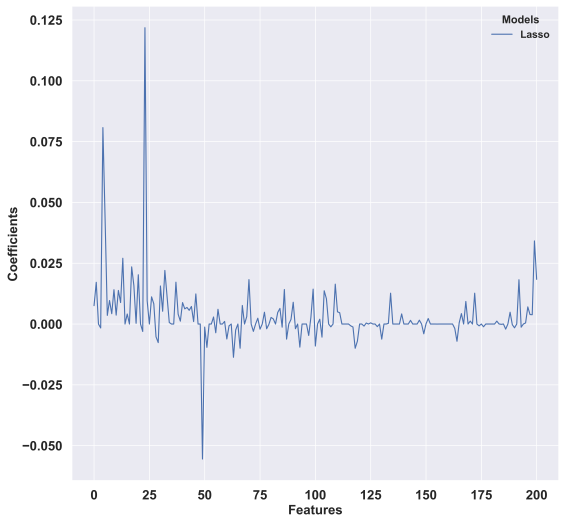


Figure 24: regression coefficient in regularized regression

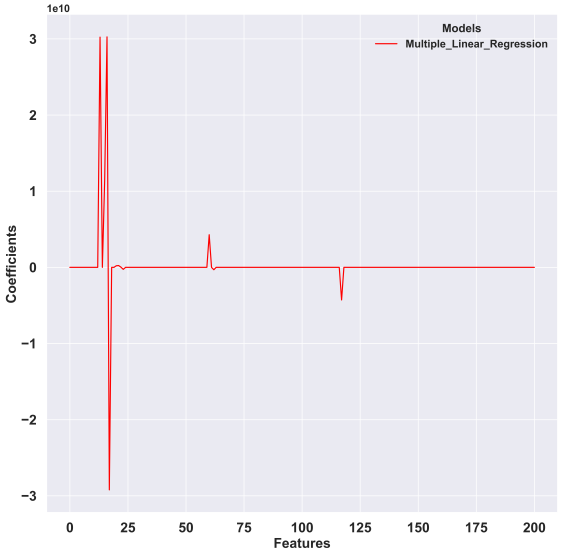


Figure 25: regression coefficient in multiple linear regression

- From figure 23, we clearly observe the sparse nature of lasso models (and thereby its ability to do feature selection) when compared to Ridge model.
- From figure 24 and 25, we observe the effect of regularisation in regression models. In the multiple linear regression model with no regularization, certain coefficient have extremely high values (of the order 10^10) thereby resulting in highly overfit and unstable models.

9 Conclusions

We were able to build generalized linear regression models for predicting housing sales price in Ames Housing dataset. The developed models were able to achieve the dual project objective of accurately predicting the housing sales price while identifying the corresponding key predictors influencing sales price. We also undertook comparative studies between Ridge, LASSO and multi-linear regression models to uncover their key working differences.

An important consideration in this project was the tradeoff between model interpretability and accuracy. We could have opted for a simple model with limited and easily explainable features but we decided against it as there was a considerable drop-off on the prediction accuracy. Due to the presence of a large number of features, it was not easy to interpret all the results in our model without the application of the corresponding domain knowledge. Also, the effects of inflation is not directly considered in our model.

10 Key Takeaways

Working on this project helped us derive business insights through the application of various regression and data processing techniques learnt in this course. This project introduced us to the challenge of working with a real life dataset and the associated complexities such as handling missing values, non representative data points, dataset size, large number of features and bias-variance tradeoff. The project allowed us to explore regularized regression techniques in great detail to supplement the learnings from the course. We learnt the knack of allocating time to different facets of project workflow (proposal, data collection, data modelling, presentation and report generation) and meet the deliverables within the given time duration (This skill will come very handy in a workplace environment). We would like to thank Dr.Yajun Mei for providing us with this enriching opportunity.

References

[1] De Cock, D. (2011). "Ames, Iowa: Alternative to the Boston Housing Data as an End of Semester Regression Project," Journal of Statistics Education, Volume 19, Number 3.

11 APPENDIX

11.1 K-Fold Cross Validation & Grid Search

- Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample.The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into.This approach involves randomly dividing the set of observations into k groups, or folds, of approximately equal size. The first fold is treated as a validation set, and the method is fit on the remaining (k-1) folds.
- Grid search is a tuning technique that attempts to compute the optimum values of hyperparameters. It is an exhaustive search that is performed on all the different parameter values specified for a model.

Figure 26: Working of k-Fold Cross validation

11.2 CODE

```
# Importing Required Library
import pandas as pd
import re
import matplotlib.pyplot as plt
import csv
import numpy as np
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split,cross_val_score, KFold,
    GridSearchCV
from sklearn.linear_model import LassoCV,Ridge,Lasso,RidgeCV, LinearRegression
```

```
from sklearn.metrics import mean_squared_error,r2_score
from sklearn.impute import SimpleImputer
from yellowbrick.regressor import ResidualsPlot
plt.rcParams['font.weight']="bold"
plt.rcParams['axes.labelweight']="bold"
plt.rcParams['font.size']="18"

#Reading data

with open('AmesHousing.txt') as f:
    reader = csv.reader(f, delimiter="\t")
    data = list(reader)
feature=data[0]
feature = [re.sub(r'\s+', '_', item) for item in feature]
housing_data=pd.DataFrame.from_records(data[1:],columns=feature)


# # Missing Values

for item in housing_data.columns:
    housing_data[item].replace(['','NA'],np.nan, inplace=True)

numeric_features = ['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']
housing_data[numeric_features] = housing_data[numeric_features].astype('float'
    )

Null_vals=housing_data.isnull().sum()
print("Missing␣Values␣is␣encountered␣in␣the␣following␣variables:␣\n",
    Null_vals[housing_data.isnull().sum()>0])

#Imputation strategy for Missing Values
df_lots = housing_data[['Lot_Frontage','Lot_Config', 'Lot_Shape']]
mean_df = df_lots.groupby(['Lot_Shape']).mean().reset_index()
#mean_df
housing_data['Lot_Frontage'] = housing_data.apply(lambda x : mean_df[mean_df['
    Lot_Shape']== x['Lot_Shape']]['Lot_Frontage'].values[0]
                                                        if x['Lot_Frontage']!=x['
    Lot_Frontage'] else x['Lot_Frontage'],axis=1)
#mean_df
#del housing_data['Lot_Frontage_Mod']
#housing_data[['Lot_Frontage_Mod','Lot_Frontage']]
#housing_data.loc[housing_data['Lot_Frontage'].isnull(),['Lot_Frontage_Mod','
    Lot_Frontage','Lot_Shape']]
# Group 1:
group_1 = [
    'Pool_QC', 'Misc_Feature', 'Alley', 'Fence', 'Fireplace_Qu', 'Garage_Type'
        ,
    'Garage_Finish', 'Garage_Qual', 'Garage_Cond', 'Bsmt_Qual', 'Bsmt_Cond',
    'Bsmt_Exposure', 'BsmtFin_Type_1', 'BsmtFin_Type_2', 'Mas_Vnr_Type'
```

```
]
housing_data[group_1] = housing_data[group_1].fillna("None")
# Group 2:
group_2 = [
    'Garage_Area', 'Garage_Cars', 'BsmtFin_SF_1', 'BsmtFin_SF_2', 'Bsmt_Unf_SF
        ',
    'Total_Bsmt_SF', 'Bsmt_Full_Bath', 'Bsmt_Half_Bath', 'Mas_Vnr_Area'
]
housing_data[group_2] = housing_data[group_2].fillna(0)
# Group 3:
group_3a = ['Electrical']
imputer = SimpleImputer(strategy='most_frequent')
housing_data[group_3a] = pd.DataFrame(imputer.fit_transform(housing_data[
    group_3a]), index=housing_data.index)
housing_data.Garage_Yr_Blt = housing_data.Garage_Yr_Blt.fillna(0)

# # Data Visualization / Outlier Detection


corr_mat = housing_data.corr().SalePrice.sort_values(ascending=False)
corr_mat.head(5)

plt.figure(figsize=(6, 6))
sns.scatterplot(x='Gr_Liv_Area', y='SalePrice',data=housing_data)
title = plt.title('House␣Price␣vs.␣Living␣Area')

outlier_index = housing_data[(housing_data.Gr_Liv_Area > 4000)].index
#print(len(outlier_index))
housing_data.drop(index=outlier_index , inplace= True)
#print(len(housing_data))

# # Categorical Data Visualization
# Answer questions like : How many categories in each feature ? How rare is
    each category ? These visualizations will help in developing our
    prediction pipeline

from math import ceil
from itertools import zip_longest
categorical_feature=housing_data.select_dtypes(include=["object"]).columns.
    to_list()
categorical_data=housing_data[categorical_feature]

n_string_features = categorical_data.shape[1]
nrows, ncols = ceil(n_string_features / 2), 2
fig, axs = plt.subplots(ncols=ncols, nrows=nrows, figsize=(14, 80))

for feature_name, ax in zip_longest(categorical_data, axs.ravel()):
    categorical_data[feature_name].value_counts().plot.barh(ax=ax)
    ax.set_title(feature_name)
plt.subplots_adjust(hspace=0.2, wspace=0.8)

housing_data['houseage'] = housing_data.Yr_Sold - housing_data.Year_Built
housing_data['IsRemodeled'] = np.where(housing_data["Year_Remod/Add"] ==
    housing_data.Year_Built, 0, 1)
housing_data.drop(["Yr_Sold","Year_Built","Year_Remod/Add"],inplace=True, axis
    =1)

##Dropping Features
```

```
housing_data.drop(["Street", "Utilities", "Condition_2", "Roof_Matl", "Heating
    ", "Pool_QC", "Pool_Area"], inplace=True , axis=1)
#housing_data.select_dtypes(include='object').info()

##Encoding Ordinal Variables
lot_shape_dict = {'Reg':0,'IR1':1,'IR2':1,'IR3':1}
housing_data['Lot_Shape'] = housing_data['Lot_Shape'].map(lot_shape_dict)

land_shape_dict = {'Gtl':0,'Mod':1,'Sev':2}
housing_data['Land_Slope'] = housing_data['Land_Slope'].map(land_shape_dict)

ext_qual = {'Ex':2,'Fa':-1,'Gd':1,'TA':0,'Po':-2,'None':-1.25}
housing_data['Exter_Qual'] = housing_data['Exter_Qual'].map(ext_qual)
housing_data['Exter_Cond'] = housing_data['Exter_Cond'].map(ext_qual)
housing_data['Bsmt_Qual'] = housing_data['Bsmt_Qual'].map(ext_qual)
housing_data['Bsmt_Cond'] = housing_data['Bsmt_Cond'].map(ext_qual)
housing_data['Heating_QC'] = housing_data['Heating_QC'].map(ext_qual)
housing_data['Kitchen_Qual'] = housing_data['Kitchen_Qual'].map(ext_qual)
housing_data['Fireplace_Qu'] = housing_data['Fireplace_Qu'].map(ext_qual)
housing_data['Garage_Qual'] = housing_data['Garage_Qual'].map(ext_qual)
housing_data['Garage_Cond'] = housing_data['Garage_Cond'].map(ext_qual)

bsmt_expo = {'Gd':3, 'No':0 , 'Mn':1, 'Av':2, 'None':-1}
housing_data['Bsmt_Exposure'] = housing_data['Bsmt_Exposure'].map(bsmt_expo)

fin_map_type = {'BLQ':2, 'Rec':1, 'ALQ':3, 'GLQ':4, 'Unf':-0.5, 'LwQ':0, 'None
    ':-1}
housing_data['BsmtFin_Type_1'] = housing_data['BsmtFin_Type_1'].map(
    fin_map_type)
housing_data['BsmtFin_Type_2'] = housing_data['BsmtFin_Type_2'].map(
    fin_map_type)

elec_type = {'SBrkr':3, 'FuseA':2, 'FuseF':1, 'FuseP':0, 'Mix':-1}
housing_data['Electrical'] = housing_data['Electrical'].map(elec_type)

fun_type = {'Typ':4, 'Mod':1, 'Min1':3, 'Min2':2, 'Maj1':0, 'Maj2':-1, 'Sev'
    :-2, 'Sal':-3}
housing_data['Functional'] = housing_data['Functional'].map(fun_type)

gar_fin = {'Fin':2, 'Unf':0, 'RFn':1, 'None':-1}
housing_data['Garage_Finish'] = housing_data['Garage_Finish'].map(gar_fin)

pavd = {'P':0, 'Y':1, 'N':-1}
housing_data['Paved_Drive'] = housing_data['Paved_Drive'].map(pavd)
nom_fi = list(housing_data.select_dtypes(include='object'))
X = pd.get_dummies(housing_data,columns=nom_fi,drop_first=True)
corr_mat=X.corr()
k = 10 #number of variables for heatmap
cols = corr_mat.nlargest(k, 'SalePrice')['SalePrice'].index
cm = np.corrcoef(housing_data[cols].values.T)
sns.set(font_scale=1.25)
hm = sns.heatmap(cm, cbar=True, annot=True, square=True, fmt='.2f', annot_kws
    ={'size': 10}, yticklabels=cols.values, xticklabels=cols.values)
plt.show()
X.to_csv("final_data.csv")
columns = list(X)
Y = np.log(X['SalePrice'].to_numpy())
X.drop(columns='SalePrice',inplace=True)
columns = list(X)
```

```
X = X.to_numpy()
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2)
ss = StandardScaler()
X_train = ss.fit_transform(X_train)
X_test = ss.transform(X_test)

##Model Building
model = Lasso()
# define model evaluation method
cv = KFold(n_splits=10)
# define grid
grid = dict()
grid['alpha'] =[1e-4,1e-3, 1e-2, 0.1, 1, 10, 100]
# define search
search = GridSearchCV(model, grid, scoring='neg_mean_squared_error', cv=cv,
    n_jobs=-1)
# perform the search
results = search.fit(X_train, Y_train)
# summarize
print('MSE: %.3f' % np.abs(results.best_score_))
print('Config: %s' % results.best_params_)

blank_array = np.zeros((len(grid['alpha']),10))
#print(blank_array.shape)
split = ['split{}_test_score'.format(no) for no in range(10)]
for index,elem in enumerate(split):
    blank_array[:,index] = results.cv_results_[elem]

x = np.array(grid['alpha'])
y = np.sqrt(-blank_array.mean(axis=1))
plt.loglog(x,y)

final_model = Lasso(alpha=0.001)
final_model.fit(X_train,Y_train)
Y_pred = final_model.predict(X_test)
print("test rmse",mean_squared_error(Y_test,Y_pred))
Y_pred= final_model.predict(X_train)
print("train rmse",mean_squared_error(Y_train,Y_pred))
np.sum(final_model.coef_!=0)
lasso_df = pd.DataFrame(columns=['Feature','Coef'])
lasso_df['Feature'] = columns
lasso_df['Coef'] = final_model.coef_
data = lasso_df.sort_values(by='Coef',ascending=False)
plt.figure(figsize=(40,40))
sns.barplot(data=data[:40],y='Feature',x='Coef')
len(data[data["Coef"]>0]) #No of features selected by LASSO
model = Lasso(alpha=0.001)
visualizer = ResidualsPlot(model, hist=False, qqplot=True)
visualizer.fit(X_train, Y_train)
visualizer.score(X_test, Y_test)
visualizer.show()


# # Ridge Regression

model = Ridge()
# define model evaluation method
cv = KFold(n_splits=10)
```

```
# define grid
grid = dict()
grid['alpha'] =[1e-4,1e-3, 1e-2, 0.1, 1, 10, 100, 1000]
# define search
search = GridSearchCV(model, grid, scoring='neg_mean_squared_error', cv=cv,
    n_jobs=-1)
# perform the search
results = search.fit(X_train, Y_train)
# summarize
print('RMSE: %.3f' % np.abs(results.best_score_))
print('Config: %s' % results.best_params_)


blank_array = np.zeros((len(grid['alpha']),10))
#print(blank_array.shape)
split = ['split{}_test_score'.format(no) for no in range(10)]
for index,elem in enumerate(split):
    blank_array[:,index] = results.cv_results_[elem]
x = np.array(grid['alpha'])
y = np.sqrt(-blank_array.mean(axis=1))
plt.xlabel("lambda")
plt.ylabel("RMSE")
plt.loglog(x,y)
#plt.grid(axis='both',color=0.1)

final_model_ridge = Ridge(alpha=100)
final_model_ridge.fit(X_train,Y_train)
Y_pred_ridge = final_model_ridge.predict(X_test)
print(mean_squared_error(Y_test,Y_pred_ridge))

Y_pred=final_model_ridge.predict(X_train)
print(mean_squared_error(Y_train,Y_pred))

Ridge_df = pd.DataFrame(columns=['Feature','Coef'])
Ridge_df['Feature'] = columns
Ridge_df['Coef'] = final_model_ridge.coef_
data = Ridge_df.sort_values(by='Coef',ascending=False)
plt.figure(figsize=(40,40))
sns.barplot(data=data[:40],y='Feature',x='Coef')

model = Ridge(alpha=100)
visualizer = ResidualsPlot(model, hist=False, qqplot=True)
visualizer.fit(X_train, Y_train)
visualizer.score(X_test, Y_test)
visualizer.show()


# # Linear Regression

lr= LinearRegression().fit(X_train,Y_train)
prediction= lr.predict(X_test)
print(np.sqrt(mean_squared_error(prediction,Y_test)))

plt.figure(figsize=(12,12), dpi=800)
g=sns.lineplot(data=final_model.coef_)
plt.tick_params(axis="both",labelsize=18)
g.set_xlabel('Features', fontsize=18)
g.set_ylabel('Coefficients', fontsize=18)
g.legend(title='Models', loc='upper right', labels=['Lasso'], fontsize=14)
```

```
plt.savefig("Lasso.tiff")

plt.figure(figsize=(12,12), dpi=800)
g=sns.lineplot(data=lr.coef_, color='red')
plt.tick_params(axis="both",labelsize=18)
g.set_xlabel('Features', fontsize=18)
g.set_ylabel('Coefficients', fontsize=18)
g.legend(title='Models', loc='upper right', labels=['
    Multiple_Linear_Regression'], fontsize=14)
plt.savefig("LinearReg.tiff")

plt.figure(figsize=(12,12), dpi=800)
g=sns.lineplot(data=final_model.coef_)
sns.lineplot(data=final_model_ridge.coef_, color='red')
plt.tick_params(axis="both",labelsize=18)
g.set_xlabel('Features', fontsize=18)
g.set_ylabel('Coefficients', fontsize=18)
g.legend(title='Models', loc='upper right', labels=['Lasso', 'Ridge'],
    fontsize=14)
plt.savefig("LassovRidge.tiff")
```