House Price Prediction

GR5293 EODS – Project 1
Contributer: Lichun He / Yuqi Zhang / Haishan Mei / Lei Huang

Outline

- Dataset Overview
- Data Cleaning
- Feature Selection
- Model Evaluation
- Best Model

Dataset Overview

- House Prices dataset generated from Kaggle[1]
  - 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa;
  - 1 target variable - SalePrice
- Pre-split two dataset
  - Train dataset: (1460, 80) - with ‘SalePrice’
  - Test dataset: (1459, 79) - without ‘SalePrice’

```
df_train.head()
```

Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	MoSold	YrSold	SaleType	SaleCondition	SalePrice
1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	Inside	...	0	NaN	NaN	NaN	0	2	2008	WD	Normal	208500
2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	FR2	...	0	NaN	NaN	NaN	0	5	2007	WD	Normal	181500
3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	Inside	...	0	NaN	NaN	NaN	0	9	2008	WD	Normal	223500
4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	Corner	...	0	NaN	NaN	NaN	0	2	2006	WD	Abnorml	140000
5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	FR2	...	0	NaN	NaN	NaN	0	12	2008	WD	Normal	250000

5 rows × 80 columns

1: https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/data

Dataset Overview

- Mixture of numeric and categorical variables
  - Nominal variable
    - RoofStyle: Type of roof

Flat	Flat
Gable	Gable
Gambrel	Gabrel (Barn)
Hip	Hip
Mansard	Mansard
Shed	Shed

  - Ordinal variable
    - OverallQual: Rates the overall material and finish of the house

10	Very Excellent
9	Excellent
...	
2	Poor
1	Very Poor

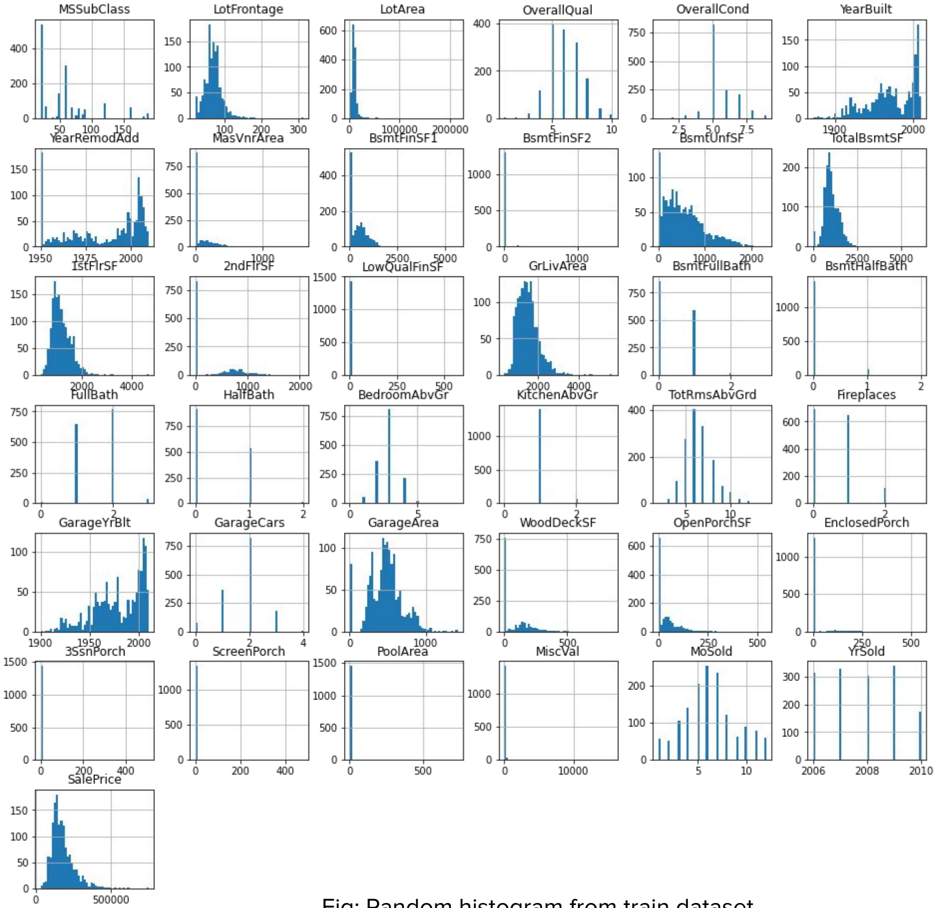


Fig: Random histogram from train dataset

Data Cleaning

Step 0: Extra target variable and drop it from train data;
Merge pre-split datasets together into our self-use train dataset

```
df_train = pd.concat((X, df_test))
df_train.shape
```

```
(2919, 79)
```

Step 1: Check duplicate data

```
df_train[df_train.duplicated(keep='first')]
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	...
Id											

0 rows × 80 columns

✅ Our dataset doesn't have duplicate data.

Data Cleaning

Step 2: Check missing data

```
def Missingtable(df,missing_col_name ='Missing Data Count'):
    missing_df = pd.DataFrame(df.isnull().sum(), columns=[missing_col_name])
    return missing_df[missing_df[missing_col_name]!=0].sort_values(missing_col_name,ascending=True)

print('Number of features having missing values: {}'.format(Missingtable(df_train).shape[0]))
```

```
Number of features having missing values: 19
```

📌 We find that there exists missing datas in our datasets.

However, after taking a closer look into our datas, we find that some NA actually has its meanings. i.e. :

```
Alley: Type of alley access to property

       Grvl     Gravel
       Pave     Paved
       NA       No alley access
```

Data Cleaning

Step 2: Check missing data (cont.)

- Manually handle missing values
  - Based on the features' defination
    - Fill NA with different specific words

```
# Alley : data description says NA means "no alley access"
df_train.loc[:, "Alley"] = df_train.loc[:, "Alley"].fillna("None")
```

    - Convert some "numerical" features into categorical features

```
df_train.replace({"MSSubClass" : {20 : "SC20", 30 : "SC30", 40 : "SC40", 45 : "SC45",
                                   50 : "SC50", 60 : "SC60", 70 : "SC70", 75 : "SC75",
                                   80 : "SC80", 85 : "SC85", 90 : "SC90", 120 : "SC120",
                                   150 : "SC150", 160 : "SC160", 180 : "SC180", 190 : "SC190"},
                  "MoSold" : {1 : "Jan", 2 : "Feb", 3 : "Mar", 4 : "Apr", 5 : "May", 6 : "Jun",
                              7 : "Jul", 8 : "Aug", 9 : "Sep", 10 : "Oct", 11 : "Nov", 12 : "Dec"}
                  })
```

    - Convert some "nominal" features into ordinal features

```
df_train.replace({"Alley" : {"Grvl" : 1, "Pave" : 2},
              "BsmtCond" : {"No" : 0, "Po" : 1, "Fa" : 2, "TA" : 3, "Gd" : 4, "Ex" : 5},
              "BsmtExposure" : {"No" : 0, "Mn" : 1, "Av": 2, "Gd" : 3},
              "BsmtFinType1" : {"No" : 0, "Unf" : 1, "LwQ": 2, "Rec" : 3, "BLQ" : 4,
                                "ALQ" : 5, "GLQ" : 6},
              "BsmtFinType2" : {"No" : 0, "Unf" : 1, "LwQ": 2, "Rec" : 3, "BLQ" : 4,
                                "ALQ" : 5, "GLQ" : 6},
              "BsmtQual" : {"No" : 0, "Po" : 1, "Fa" : 2, "TA": 3, "Gd" : 4, "Ex" : 5},
```

Data Cleaning

Step 2: Check missing values (cont.)

- Create new features
  - Simplifications of existing features

```
df_train["SimplOverallQual"] = df_train.OverallQual.replace({1 : 1, 2 : 1, 3 : 1, # bad
                                                             4 : 2, 5 : 2, 6 : 2, # average
                                                             7 : 3, 8 : 3, 9 : 3, 10 : 3 # good
                                                            })
```

  - Combinations of existing features

```
# Overall kitchen score
df_train["KitchenScore"] = df_train["KitchenAbvGr"] * df_train["KitchenQual"]
# Overall fireplace score
df_train["FireplaceScore"] = df_train["Fireplaces"] * df_train["FireplaceQu"]
```

  - Polynomials on the ***top 10 existing features***

Data Cleaning

Step 2: Check missing values (cont.)

- Polynomials on the ***top 10 existing features***

```
# Find most important features relative to target
print("Find most important features relative to target")
corr = df_train.corr()
corr.sort_values(["SalePrice"], ascending = False, inplace = True)
corr.SalePrice
```

```
Find most important features relative to target
SalePrice        1.000000
OverallQual      0.819240
AllSF            0.817272
AllFlrsSF        0.729421
GrLivArea        0.718844
                   ...
LandSlope       -0.040114
SimplExterCond  -0.042183
KitchenAbvGr    -0.147891
EnclosedPorch   -0.148636
LotShape        -0.285903
Name: SalePrice, Length: 88, dtype: float64
```

```
df_train["OverallQual-s2"] = df_train["OverallQual"] ** 2
df_train["OverallQual-s3"] = df_train["OverallQual"] ** 3
df_train["OverallQual-Sq"] = np.sqrt(df_train["OverallQual"])
df_train["AllSF-2"] = df_train["AllSF"] ** 2
df_train["AllSF-3"] = df_train["AllSF"] ** 3
df_train["AllSF-Sq"] = np.sqrt(df_train["AllSF"])
df_train["AllFlrsSF-2"] = df_train["AllFlrsSF"] ** 2
df_train["AllFlrsSF-3"] = df_train["AllFlrsSF"] ** 3
df_train["AllFlrsSF-Sq"] = np.sqrt(df_train["AllFlrsSF"])
df_train["GrLivArea-2"] = df_train["GrLivArea"] ** 2
df_train["GrLivArea-3"] = df_train["GrLivArea"] ** 3
df_train["GrLivArea-Sq"] = np.sqrt(df_train["GrLivArea"])
df_train["SimplOverallQual-s2"] = df_train["SimplOverallQual"] ** 2
df_train["SimplOverallQual-s3"] = df_train["SimplOverallQual"] ** 3
df_train["SimplOverallQual-Sq"] = np.sqrt(df_train["SimplOverallQual"])
df_train["ExterQual-2"] = df_train["ExterQual"] ** 2
df_train["ExterQual-3"] = df_train["ExterQual"] ** 3
df_train["ExterQual-Sq"] = np.sqrt(df_train["ExterQual"])
df_train["GarageCars-2"] = df_train["GarageCars"] ** 2
df_train["GarageCars-3"] = df_train["GarageCars"] ** 3
df_train["GarageCars-Sq"] = np.sqrt(df_train["GarageCars"])
df_train["TotalBath-2"] = df_train["TotalBath"] ** 2
df_train["TotalBath-3"] = df_train["TotalBath"] ** 3
df_train["TotalBath-Sq"] = np.sqrt(df_train["TotalBath"])
df_train["KitchenQual-2"] = df_train["KitchenQual"] ** 2
df_train["KitchenQual-3"] = df_train["KitchenQual"] ** 3
df_train["KitchenQual-Sq"] = np.sqrt(df_train["KitchenQual"])
df_train["GarageScore-2"] = df_train["GarageScore"] ** 2
df_train["GarageScore-3"] = df_train["GarageScore"] ** 3
df_train["GarageScore-Sq"] = np.sqrt(df_train["GarageScore"])
```

Data Cleaning

Step 2: Check missing values (cont.)

- Differentiate numerical features (minus the target) and categorical features
- Inpute remaining missing values for numerical features using median

```
train_num = train_num.fillna(train_num.median())
print("Remaining NAs for numerical features in train : " + str(train_num.isnull().values.sum()))
```

```
Remaining NAs for numerical features in train : 0
```

- Create dummy features for categorical values via one-hot encoding

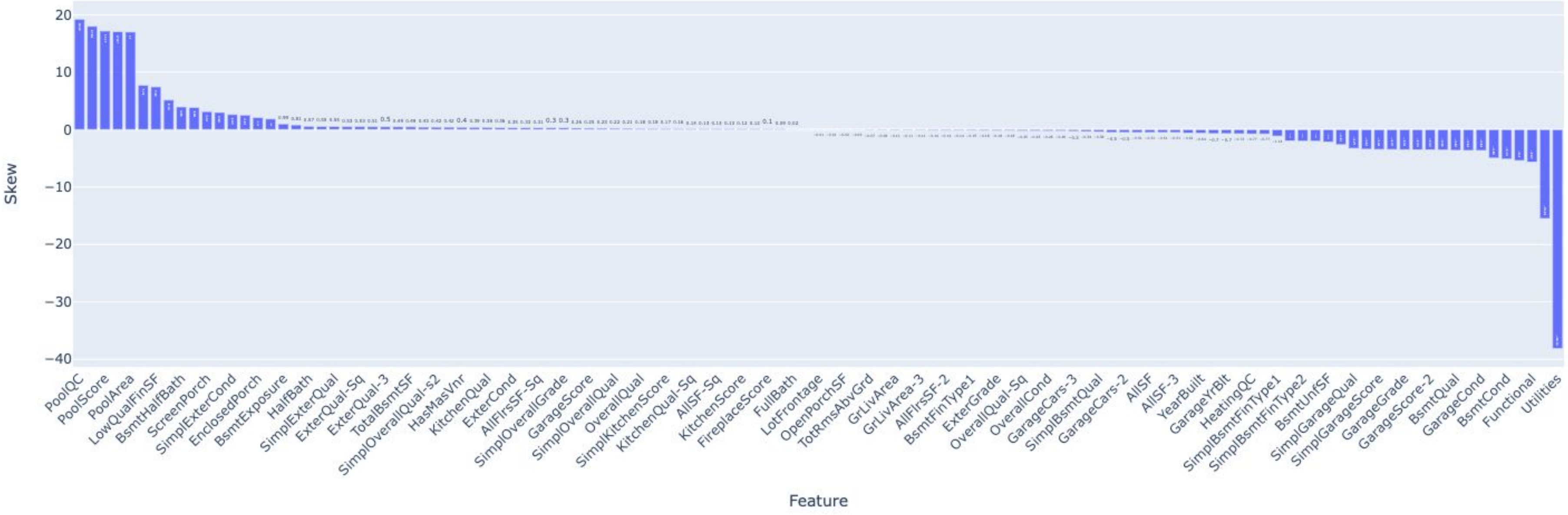
```
print("NAs for categorical features in train : " + str(train_cat.isnull().values.sum()))
train_cat = pd.get_dummies(train_cat)
print("Remaining NAs for categorical features in train : " + str(train_cat.isnull().values.sum()))
```

```
NAs for categorical features in train : 2
Remaining NAs for categorical features in train : 0
```

✅ Our dataset are ready for next step!

Data Cleaning

Step 3: Check skewness



Data Cleaning

Step 3: Check skewness (cont.)

- Log transform of the skewed numerical features to lessen impact of outliers
  - Inspired by Alexandru Papiu's script[1]
  - | skewness | > 0.5 is considered at least moderately skewed

```
skewness = train_num.apply(lambda x: skew(x))
skewness = skewness[abs(skewness) > 0.5]
print(str(skewness.shape[0]) + " skewed numerical features to log transform")
skewed_features = skewness.index
train_num[skewed_features] = np.log1p(train_num[skewed_features])
```

```
87 skewed numerical features to log transform
```

✅ Our dataset are ready for fitting model!

1: https://www.kaggle.com/apapiu/house-prices-advanced-regression-techniques/regularized-linear-models

Fitting Models

- Split cleaned data into training and testing dataset based on their Id
  - for predict target test

```
new_train = train.iloc[:1460,:]
new_test = train.iloc[1460:,:]
```

- Split train data with 80% as training set & 20% as testing set
  - for validation test

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(new_train, y,
                                                    test_size = 0.2,
                                                    random_state = 123)

print("X_train : " + str(X_train.shape))
print("X_test : " + str(X_test.shape))
print("y_train : " + str(y_train.shape))
print("y_test : " + str(y_test.shape))
```

```
X_train : (1168, 321)
X_test : (292, 321)
y_train : (1168,)
y_test : (292,)
```

Feature Selection

- LASSO regularization drives the coefficient of uninformative features to 0

```
[406] from sklearn.linear_model import LogisticRegression

      # Now with LASSO
      logr = LogisticRegression(C=0.1, penalty="l1", solver="liblinear", random_state=123)
      logr.fit(X_train,y_train.astype('int'))

      sorted_tuples = sorted(list(zip(X_train.columns.values,logr.coef_[0])),key=lambda x:x[1],reverse=True)
      for feature,coef in sorted_tuples:
          print(f'{feature:30s} : {coef: 0.3f}')
```

```
MiscVal                        :  0.216
BsmtUnfSF                      :  0.169
EnclosedPorch                  :  0.137
OpenPorchSF                    :  0.081
YrSold                         :  0.012
AllPorchSF                     :  0.011
GarageScore-Sq                 :  0.003
LotFrontage                    :  0.000
LotArea                        :  0.000
Street                         :  0.000
...
```

```
[407] # which columns were kept?
      X_train.columns[logr.coef_[0] != 0]
```

```
Index(['YearRemodAdd', 'BsmtFinSF1', 'BsmtUnfSF', 'TotalBsmtSF', 'GarageArea',
       'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', 'MiscVal', 'YrSold',
       'OverallGrade', 'AllPorchSF', 'GrLivArea-Sq', 'GarageScore-Sq'],
      dtype='object')
```

Keeping 14 features !

Model Evaluation

Base Line

```
[411] from sklearn.dummy import DummyRegressor
      dummyr = DummyRegressor(strategy='mean')
      dummyr.fit(X_train,y_train)
      dummy_pred_test = dummyr.predict(X_test)
      dummy_pred_train = dummyr.predict(X_train)
```

```
[422] from sklearn.metrics import mean_squared_error, r2_score
0s    dummy_mse_test = mean_squared_error(y_test,dummy_pred_test)
      dummy_mse_train = mean_squared_error(y_train,dummy_pred_train)
      dummy_rmse_test = np.sqrt(dummy_mse_test)
      dummy_rmse_train = np.sqrt(dummy_mse_train)
      dummy_r2 = r2_score(y_test,dummy_pred_test)
      print(f'dummy_mse on Test set:{dummy_mse_test:0.10f}')
      print(f'dummy_mse on Traing set:{dummy_mse_train:0.10f}')
      print(f'dummy_rmse on Test set: {dummy_rmse_test:0.10f}')
      print(f'dummy_rmse on Traing set: {dummy_rmse_train:0.10f}')
      print(f'dummy_r2: {dummy_r2:0.10f}')
```

```
dummy_mse on Test set:0.1438106398
dummy_mse on Traing set:0.1600165010
dummy_rmse on Test set: 0.3792237331
dummy_rmse on Traing set: 0.4000206257
dummy_r2: -0.0004422117
```

Model Evaluation

- Linear regression without regularization

```
[31] from sklearn.linear_model import LinearRegression
     lr_model = LinearRegression()
     lr_model.fit(X_train, y_train)
     lr_pred_test = lr_model.predict(X_test)
     lr_pred_train = lr_model.predict(X_train)
```

```
[32] lr_mse_test = mean_squared_error(y_test,lr_pred_test)
     lr_mse_train = mean_squared_error(y_train,lr_pred_train)
     lr_rmse_test = np.sqrt(lr_mse_test)
     lr_rmse_train = np.sqrt(lr_mse_train)
     lr_r2_test = r2_score(y_test,lr_pred_test)
     print(f'lr_mse_test:{lr_mse_test:0.10f}')
     print(f'lr_mse_train:{lr_mse_train:0.10f}')
     print(f'lr_rmse_test:{lr_rmse_test:0.10f}')
     print(f'lr_rmse_train:{lr_rmse_train:0.10f}')
     print(f'lr_r2:{lr_r2_test:0.10f}')
```

```
lr_mse_test:0.0181849503
lr_mse_train:0.0082070715
lr_rmse_test:0.1348515862
lr_rmse_train:0.0905928890
lr_r2:0.8775446516
```

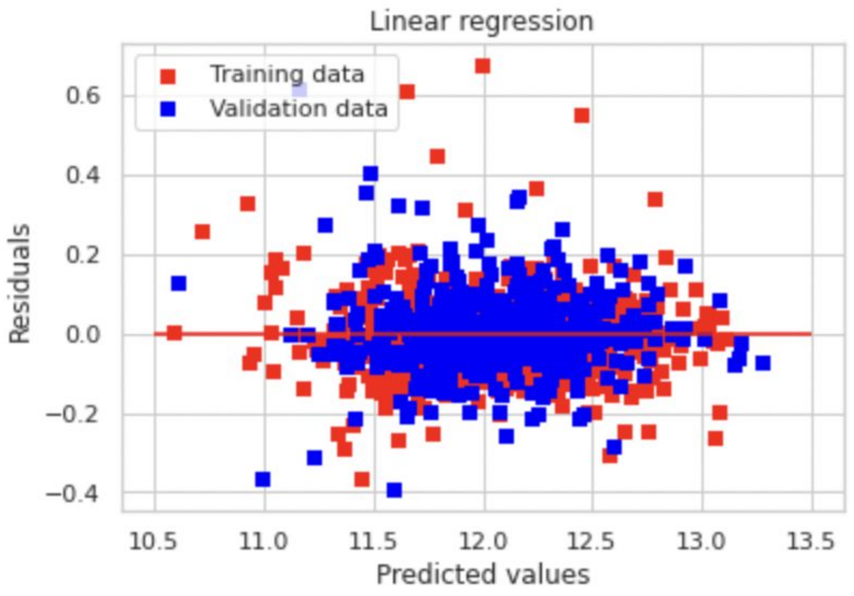
MSE for test set: 0.01818

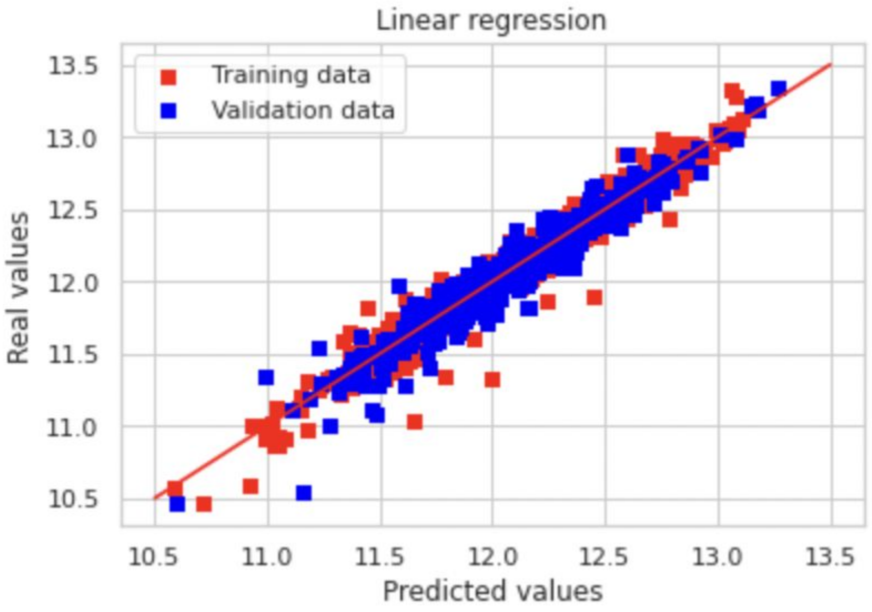
MSE for training set: 0.00821

RMSE for test set: 0.13485

RMSE for training set: 0.09059

Model Evaluation





- Errors are seem to be randomly distributed and randomly scattered around the mean
- Most data points are on or close to the the predicted regression line.

Model Evaluation

💡 Trying to find the best hyperparameter!

Ridge

```
[72] from sklearn.linear_model import RidgeCV
     ridge = RidgeCV(alphas = [0.01, 0.03, 0.06, 0.1, 0.3, 0.6, 1, 3, 6, 10, 30, 60])
     ridge.fit(X_train, y_train)
     alpha = ridge.alpha_
     print("Best alpha :", alpha)

     Best alpha : 3.0
```

```
[74] ridge_mse_test = mean_squared_error(y_test,ridge_pred_test)
     ridge_mse_train = mean_squared_error(y_train,ridge_pred_train)
     ridge_rmse_test = np.sqrt(ridge_mse_test)
     ridge_rmse_train = np.sqrt(ridge_mse_train)
     ridge_r2_test = r2_score(y_test,ridge_pred_test)
     print(f'ridge_mse_test:{ridge_mse_test:0.10f}')
     print(f'ridge_mse_train:{ridge_mse_train:0.10f}')
     print(f'ridge_rmse_test:{ridge_rmse_test:0.10f}')
     print(f'ridge_rmse_train:{ridge_rmse_train:0.10f}')
     print(f'ridge_r2:{ridge_r2_test:0.2f}')

     ridge_mse_test:0.0112391139
     ridge_mse_train:0.0088031850
     ridge_rmse_test:0.1060146873
     ridge_rmse_train:0.0938252899
     ridge_r2:0.92
```

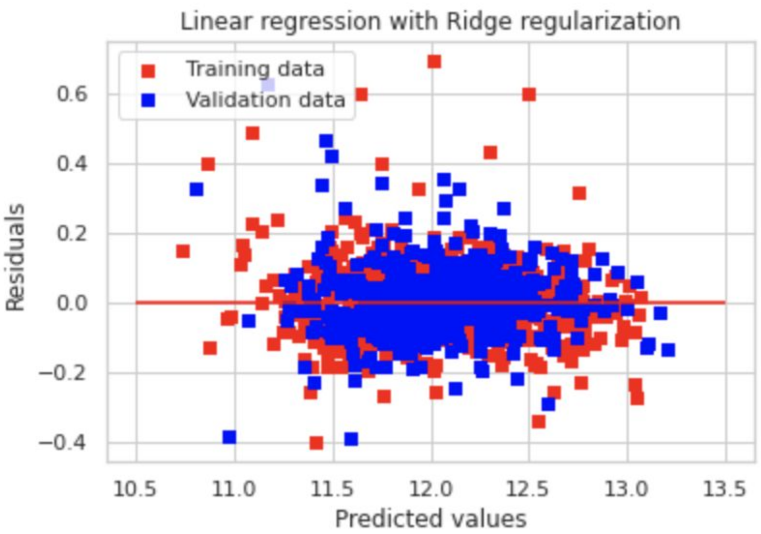
MSE for test set: 0.011239

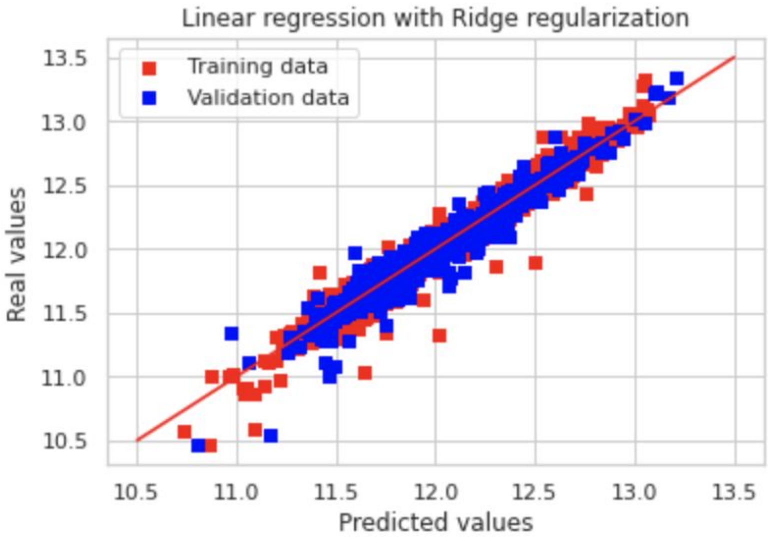
MSE for training set: 0.008803

RMSE for test set: 0.106015

RMSE for training set: 0.093825

Model Evaluation






```
[85] ridge_model.score(X_train, y_train)

0.9449857673236512
```

```
[138] ridge_scores = cross_val_score(Ridge(alpha=10),X_train, y_train,cv=5)
      print(f'mean cv accuracy:{ridge_scores.mean():0.5f}')

mean cv accuracy:0.93192
```

Summary

Model 1

```
import statsmodels.formula.api as smf
model1 = smf.ols('SalePrice ~ LotFrontage+YearRemodAdd+BsmtFinSF1+BsmtUnfSF+TotalBsmtSF
print(model1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              SalePrice   R-squared:                       0.843
Model:                            OLS   Adj. R-squared:                  0.843
Method:                 Least Squares   F-statistic:                     1115.
Date:                Tue, 29 Mar 2022   Prob (F-statistic):               0.00
Time:                        01:49:06   Log-Likelihood:                 1265.6
No. Observations:                2912   AIC:                            -2501.
Df Residuals:                    2897   BIC:                            -2412.
Df Model:                          14
Covariance Type:            nonrobust
=================================================================================
                    coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------
Intercept        19.9463      4.416      4.517      0.000      11.288      28.605
LotFrontage       0.0001   8.91e-05      1.264      0.206   -6.21e-05       0.000
YearRemodAdd      0.0038      0.000     22.355      0.000       0.003       0.004
BsmtFinSF1     9.351e-05   1.92e-05      4.879      0.000    5.59e-05       0.000
BsmtUnfSF     -2.333e-05   1.85e-05     -1.259      0.208   -5.97e-05     1.3e-05
TotalBsmtSF       0.0002   1.91e-05     11.720      0.000       0.000       0.000
GarageArea        0.0002   8.31e-05      2.737      0.006    6.45e-05       0.000
WoodDeckSF        0.0001    2.5e-05      5.209      0.000    8.11e-05       0.000
OpenPorchSF     8.96e-05    4.8e-05      1.868      0.062   -4.46e-06       0.000
EnclosedPorch    -0.0003   4.96e-05     -5.981      0.000      -0.000      -0.000
MiscVal       -9.894e-06   5.88e-06     -1.682      0.093   -2.14e-05    1.64e-06
YrSold           -0.0083      0.002     -3.768      0.000      -0.013      -0.004
OverallGrade      0.0090      0.000     23.219      0.000       0.008       0.010
GarageScore    4.334e-05   2.67e-05      1.624      0.105      -9e-06    9.57e-05
GrLivArea         0.0003   7.39e-06     40.146      0.000       0.000       0.000
=================================================================================
Omnibus:                      547.140   Durbin-Watson:                   1.934
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1447.482
Skew:                          -1.006   Prob(JB):                         0.00
Kurtosis:                       5.807   Cond. No.                     5.75e+06
==============================================================================
```

Model 2

```
[143] model2 = smf.ols('SalePrice ~ YearRemodAdd+BsmtFinSF1+TotalBsmtSF+GarageArea+WoodDeckSF+E
print(model2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              SalePrice   R-squared:                       0.843
Model:                            OLS   Adj. R-squared:                  0.842
Method:                 Least Squares   F-statistic:                     1728.
Date:                Tue, 29 Mar 2022   Prob (F-statistic):               0.00
Time:                        01:46:24   Log-Likelihood:                 1259.1
No. Observations:                2912   AIC:                            -2498.
Df Residuals:                    2902   BIC:                            -2439.
Df Model:                           9
Covariance Type:            nonrobust
=================================================================================
                    coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------
Intercept        20.2432      4.415      4.586      0.000      11.587      28.899
YearRemodAdd      0.0038      0.000     22.541      0.000       0.003       0.004
BsmtFinSF1        0.0001   7.76e-06     14.800      0.000    9.97e-05       0.000
TotalBsmtSF       0.0002   9.09e-06     22.534      0.000       0.000       0.000
GarageArea        0.0004   1.71e-05     21.171      0.000       0.000       0.000
WoodDeckSF        0.0001   2.48e-05      5.144      0.000    7.89e-05       0.000
EnclosedPorch    -0.0003   4.93e-05     -6.208      0.000      -0.000      -0.000
YrSold           -0.0085      0.002     -3.842      0.000      -0.013      -0.004
OverallGrade      0.0091      0.000     23.763      0.000       0.008       0.010
GrLivArea         0.0003   7.24e-06     41.387      0.000       0.000       0.000
==============================================================================
Omnibus:                      525.787   Durbin-Watson:                   1.943
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1328.566
Skew:                          -0.985   Prob(JB):                    3.20e-289
Kurtosis:                       5.658   Cond. No.                     5.22e+06
==============================================================================
```

Thank you!

image from: https://siewlinyap.github.io/portfolio/project3-analysis-on-housing-data/