

lasso

April 4, 2024

0.1 Predicting House Sale Prices

```
[1]: #importing necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import Lasso
from sklearn.metrics import mean_squared_error, r2_score
```

```
[42]: # Loading the dataset
data = pd.read_csv(r'C:\Users\ntpc\Desktop\HousePrices.csv')
data.head()
```

```
[42]:
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	\
0	1	60	RL	65.0	8450	Pave	NaN	Reg	
1	2	20	RL	80.0	9600	Pave	NaN	Reg	
2	3	60	RL	68.0	11250	Pave	NaN	IR1	
3	4	70	RL	60.0	9550	Pave	NaN	IR1	
4	5	60	RL	84.0	14260	Pave	NaN	IR1	

	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	MoSold	\
0	Lvl	AllPub	...	0	NaN	NaN	NaN	0	2	
1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	5	
2	Lvl	AllPub	...	0	NaN	NaN	NaN	0	9	
3	Lvl	AllPub	...	0	NaN	NaN	NaN	0	2	
4	Lvl	AllPub	...	0	NaN	NaN	NaN	0	12	

	YrSold	SaleType	SaleCondition	SalePrice
0	2008	WD	Normal	208500
1	2007	WD	Normal	181500
2	2008	WD	Normal	223500
3	2006	WD	Abnorml	140000
4	2008	WD	Normal	250000

[5 rows x 81 columns]

```
[43]: # getting the null values
data.isnull().sum()
```

```
[43]: Id          0
      MSSubClass  0
      MSZoning   0
      LotFrontage 259
      LotArea     0
      ...
      MoSold     0
      YrSold     0
      SaleType   0
      SaleCondition 0
      SalePrice  0
      Length: 81, dtype: int64
```

```
[44]: data.shape
```

```
[44]: (1460, 81)
```

```
[45]: data.dtypes
```

```
[45]: Id          int64
      MSSubClass  int64
      MSZoning   object
      LotFrontage float64
      LotArea     int64
      ...
      MoSold     int64
      YrSold     int64
      SaleType   object
      SaleCondition object
      SalePrice  int64
      Length: 81, dtype: object
```

```
[46]: #checking how much percent of that column is missing
      data.isnull().mean().sort_values(ascending = False)
```

```
[46]: PoolQC      0.995205
      MiscFeature 0.963014
      Alley      0.937671
      Fence      0.807534
      MasVnrType 0.597260
      ...
      ExterQual   0.000000
      Exterior2nd 0.000000
      Exterior1st 0.000000
      RoofMatl    0.000000
      SalePrice   0.000000
      Length: 81, dtype: float64
```

```
[47]: # getting the null values
data.isnull().sum().sort_values(ascending = False)
```

```
[47]: PoolQC          1453
MiscFeature       1406
Alley            1369
Fence            1179
MasVnrType        872
...
ExterQual         0
Exterior2nd       0
Exterior1st       0
RoofMatl          0
SalePrice         0
Length: 81, dtype: int64
```

```
[48]: # Converting categorical variables to numerical using one-hot encoding
data = pd.get_dummies(data, columns=['MSZoning', 'Street', 'Alley', 'LotShape', '
↳ 'LandContour', 'Utilities', 'SaleType', 'SaleCondition'])

# Drop non-numeric columns with non-numeric data
data = data.select_dtypes(include=['number'])

# Compute the correlation matrix
corr_matrix = data.corr()

# Sort the correlation values of the target variable 'SalePrice' with other
↳ variables
corr_ser = corr_matrix['SalePrice'].sort_values(ascending=False)

print(corr_ser)
```

```
SalePrice          1.000000
OverallQual        0.790982
GrLivArea          0.708624
GarageCars         0.640409
GarageArea         0.623431
TotalBsmtSF        0.613581
1stFlrSF           0.605852
FullBath           0.560664
TotRmsAbvGrd       0.533723
YearBuilt          0.522897
YearRemodAdd       0.507101
GarageYrBlt        0.486362
MasVnrArea         0.477493
Fireplaces         0.466929
BsmtFinSF1         0.386420
```

```

LotFrontage      0.351799
WoodDeckSF       0.324413
2ndFlrSF         0.319334
OpenPorchSF      0.315856
HalfBath         0.284108
LotArea          0.263843
BsmtFullBath     0.227122
BsmtUnfSF        0.214479
BedroomAbvGr     0.168213
ScreenPorch      0.111447
PoolArea         0.092404
MoSold           0.046432
3SsnPorch        0.044584
BsmtFinSF2       -0.011378
BsmtHalfBath     -0.016844
MiscVal          -0.021190
Id               -0.021917
LowQualFinSF     -0.025606
YrSold           -0.028923
OverallCond      -0.077856
MSSubClass       -0.084284
EnclosedPorch    -0.128578
KitchenAbvGr     -0.135907
Name: SalePrice, dtype: float64

```

```

[49]: #selecting top 10 predictors
      columns = corr_ser.index[:10]
      columns

```

```

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

```

```

[50]: df2 = data.loc[:,columns]

```

```

[51]: df2.head()

```

```

[51]:   SalePrice  OverallQual  GrLivArea  GarageCars  GarageArea  TotalBsmtSF  \
0    208500           7      1710           2         548           856
1    181500           6      1262           2         460          1262
2    223500           7      1786           2         608           920
3    140000           7      1717           3         642           756
4    250000           8      2198           3         836          1145

      1stFlrSF  FullBath  TotRmsAbvGrd  YearBuilt
0         856         2             8        2003
1        1262         2             6        1976

```

2	920	2	6	2001
3	961	1	7	1915
4	1145	2	9	2000

```
[52]: df2.isna().sum()
```

```
[52]: SalePrice      0
      OverallQual   0
      GrLivArea     0
      GarageCars    0
      GarageArea    0
      TotalBsmtSF   0
      1stFlrSF      0
      FullBath      0
      TotRmsAbvGrd  0
      YearBuilt     0
      dtype: int64
```

```
[53]: #seperating X and y
      X = df2.iloc[:,1:].values
      y = df2.iloc[:,0].values
```

```
[54]: #splitting train and test values
      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,random_state = 0)
```

0.2 Model building

```
[55]: lasso = Lasso()

      # Defining hyperparameters for grid search
      param_grid = {'alpha': [0.1, 1, 10, 100]}
```

```
[56]: # Performing grid search with cross-validation
      lasso_cv = GridSearchCV(lasso, param_grid, cv=5)
      lasso_cv.fit(X_train, y_train)
```

```
[56]: GridSearchCV(cv=5, estimator=Lasso(), param_grid={'alpha': [0.1, 1, 10, 100]})
```

```
[57]: # Getting the best hyperparameters from grid search
      best_alpha = lasso_cv.best_params_['alpha']
```

```
[58]: # Initializing Lasso Regression model with the best hyperparameters
      lasso_model = Lasso(alpha=best_alpha)
      lasso_model.fit(X_train, y_train)
```

```
[58]: Lasso(alpha=100)
```

```
[59]: # Making predictions on the testing data  
y_pred = lasso_model.predict(X_test)
```

```
[60]: # Evaluating the model  
mse = mean_squared_error(y_test, y_pred)  
rmse = mse**0.5  
r2 = r2_score(y_test, y_pred)
```

```
[61]: print(f'Best Alpha: {best_alpha}')
```

```
print(f'Root Mean Squared Error (RMSE): {rmse}')
```

```
print(f'R-squared (R^2): {r2}')
```

Best Alpha: 100

Root Mean Squared Error (RMSE): 50142.429749786126

R-squared (R²): 0.6359226924538295