Australia Housing

November 28, 2023

1 Australia Housing Project

1.0.1 Import Necessary Libraries

[206]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import linear_model, metrics
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error, r2_score

import os

# hide warnings
import warnings
warnings.filterwarnings('ignore')
```

1.1 Step 1:

- Data Preparation

[329]:
```
housing = pd.read_csv("train.csv")
```

[208]:
```
# check the data
housing.head()
```

[208]:
```
   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]
```

[209]:
```python
#Check the number of rows and columns
housing.shape
```

[209]: (1460, 81)

[210]:
```python
#Check for Null Values
housing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 81 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1460 non-null   int64
 1   MSSubClass     1460 non-null   int64
 2   MSZoning       1460 non-null   object
 3   LotFrontage    1201 non-null   float64
 4   LotArea        1460 non-null   int64
 5   Street         1460 non-null   object
 6   Alley          91 non-null     object
 7   LotShape       1460 non-null   object
 8   LandContour    1460 non-null   object
 9   Utilities      1460 non-null   object
 10  LotConfig      1460 non-null   object
 11  LandSlope      1460 non-null   object
 12  Neighborhood   1460 non-null   object
 13  Condition1     1460 non-null   object
 14  Condition2     1460 non-null   object
 15  BldgType       1460 non-null   object
 16  HouseStyle     1460 non-null   object
 17  OverallQual    1460 non-null   int64
 18  OverallCond    1460 non-null   int64
 19  YearBuilt      1460 non-null   int64
```

20	YearRemodAdd	1460 non-null	int64
21	RoofStyle	1460 non-null	object
22	RoofMatl	1460 non-null	object
23	Exterior1st	1460 non-null	object
24	Exterior2nd	1460 non-null	object
25	MasVnrType	1452 non-null	object
26	MasVnrArea	1452 non-null	float64
27	ExterQual	1460 non-null	object
28	ExterCond	1460 non-null	object
29	Foundation	1460 non-null	object
30	BsmtQual	1423 non-null	object
31	BsmtCond	1423 non-null	object
32	BsmtExposure	1422 non-null	object
33	BsmtFinType1	1423 non-null	object
34	BsmtFinSF1	1460 non-null	int64
35	BsmtFinType2	1422 non-null	object
36	BsmtFinSF2	1460 non-null	int64
37	BsmtUnfSF	1460 non-null	int64
38	TotalBsmtSF	1460 non-null	int64
39	Heating	1460 non-null	object
40	HeatingQC	1460 non-null	object
41	CentralAir	1460 non-null	object
42	Electrical	1459 non-null	object
43	1stFlrSF	1460 non-null	int64
44	2ndFlrSF	1460 non-null	int64
45	LowQualFinSF	1460 non-null	int64
46	GrLivArea	1460 non-null	int64
47	BsmtFullBath	1460 non-null	int64
48	BsmtHalfBath	1460 non-null	int64
49	FullBath	1460 non-null	int64
50	HalfBath	1460 non-null	int64
51	BedroomAbvGr	1460 non-null	int64
52	KitchenAbvGr	1460 non-null	int64
53	KitchenQual	1460 non-null	object
54	TotRmsAbvGrd	1460 non-null	int64
55	Functional	1460 non-null	object
56	Fireplaces	1460 non-null	int64
57	FireplaceQu	770 non-null	object
58	GarageType	1379 non-null	object
59	GarageYrBlt	1379 non-null	float64
60	GarageFinish	1379 non-null	object
61	GarageCars	1460 non-null	int64
62	GarageArea	1460 non-null	int64
63	GarageQual	1379 non-null	object
64	GarageCond	1379 non-null	object
65	PavedDrive	1460 non-null	object
66	WoodDeckSF	1460 non-null	int64
67	OpenPorchSF	1460 non-null	int64

```
 68  EnclosedPorch  1460 non-null   int64
 69  3SsnPorch      1460 non-null   int64
 70  ScreenPorch    1460 non-null   int64
 71  PoolArea       1460 non-null   int64
 72  PoolQC         7 non-null      object
 73  Fence          281 non-null    object
 74  MiscFeature    54 non-null     object
 75  MiscVal        1460 non-null   int64
 76  MoSold         1460 non-null   int64
 77  YrSold         1460 non-null   int64
 78  SaleType       1460 non-null   object
 79  SaleCondition  1460 non-null   object
 80  SalePrice      1460 non-null   int64
dtypes: float64(3), int64(35), object(43)
memory usage: 924.0+ KB
```

Null Values Present. We will clean them in Step 4(Data Cleaning)

[211]:
```
#Cheeck the standard calculations of the data
housing.describe()
```

[211]:
```
                Id   MSSubClass  LotFrontage        LotArea  OverallQual  \
count  1460.000000  1460.000000  1201.000000    1460.000000  1460.000000
mean    730.500000    56.897260    70.049958   10516.828082     6.099315
std     421.610009    42.300571    24.284752    9981.264932     1.382997
min       1.000000    20.000000    21.000000    1300.000000     1.000000
25%     365.750000    20.000000    59.000000    7553.500000     5.000000
50%     730.500000    50.000000    69.000000    9478.500000     6.000000
75%    1095.250000    70.000000    80.000000   11601.500000     7.000000
max    1460.000000   190.000000   313.000000  215245.000000    10.000000

       OverallCond    YearBuilt  YearRemodAdd   MasVnrArea   BsmtFinSF1  …  \
count  1460.000000  1460.000000   1460.000000  1452.000000  1460.000000  …
mean      5.575342  1971.267808   1984.865753   103.685262   443.639726  …
std       1.112799    30.202904     20.645407   181.066207   456.098091  …
min       1.000000  1872.000000   1950.000000     0.000000     0.000000  …
25%       5.000000  1954.000000   1967.000000     0.000000     0.000000  …
50%       5.000000  1973.000000   1994.000000     0.000000   383.500000  …
75%       6.000000  2000.000000   2004.000000   166.000000   712.250000  …
max       9.000000  2010.000000   2010.000000  1600.000000  5644.000000  …

        WoodDeckSF  OpenPorchSF  EnclosedPorch    3SsnPorch  ScreenPorch  \
count  1460.000000  1460.000000    1460.000000  1460.000000  1460.000000
mean     94.244521    46.660274      21.954110     3.409589    15.060959
std     125.338794    66.256028      61.119149    29.317331    55.757415
min       0.000000     0.000000       0.000000     0.000000     0.000000
25%       0.000000     0.000000       0.000000     0.000000     0.000000
50%       0.000000    25.000000       0.000000     0.000000     0.000000
```

```
75%      168.000000    68.000000       0.000000      0.000000      0.000000
max      857.000000   547.000000     552.000000    508.000000    480.000000

            PoolArea       MiscVal        MoSold        YrSold      SalePrice
count  1460.000000   1460.000000   1460.000000   1460.000000    1460.000000
mean      2.758904     43.489041      6.321918   2007.815753  180921.195890
std      40.177307    496.123024      2.703626      1.328095   79442.502883
min       0.000000      0.000000      1.000000   2006.000000   34900.000000
25%       0.000000      0.000000      5.000000   2007.000000  129975.000000
50%       0.000000      0.000000      6.000000   2008.000000  163000.000000
75%       0.000000      0.000000      8.000000   2009.000000  214000.000000
max     738.000000  15500.000000     12.000000   2010.000000  755000.000000

[8 rows x 38 columns]
```

1.2 Step 2:

- Understanding the Data Dictionary

[278]:
```
# MSSubClass: Identifies the type of dwelling involved in the sale.

#          20        1-STORY 1946 & NEWER ALL STYLES
#          30        1-STORY 1945 & OLDER
#          40        1-STORY W/FINISHED ATTIC ALL AGES
#          45        1-1/2 STORY - UNFINISHED ALL AGES
#          50        1-1/2 STORY FINISHED ALL AGES
#          60        2-STORY 1946 & NEWER
#          70        2-STORY 1945 & OLDER
#          75        2-1/2 STORY ALL AGES
#          80        SPLIT OR MULTI-LEVEL
#          85        SPLIT FOYER
#          90        DUPLEX - ALL STYLES AND AGES
#         120        1-STORY PUD (Planned Unit Development) - 1946 & NEWER
#         150        1-1/2 STORY PUD - ALL AGES
#         160        2-STORY PUD - 1946 & NEWER
#         180        PUD - MULTILEVEL - INCL SPLIT LEV/FOYER
#         190        2 FAMILY CONVERSION - ALL STYLES AND AGES

# MSZoning: Identifies the general zoning classification of the sale.

#         A          Agriculture
#         C          Commercial
#         FV         Floating Village Residential
#         I          Industrial
#         RH         Residential High Density
#         RL         Residential Low Density
#         RP         Residential Low Density Park
```

```
#        RM       Residential Medium Density

# LotFrontage: Linear feet of street connected to property

# LotArea: Lot size in square feet

# Street: Type of road access to property

#        Grvl     Gravel
#        Pave     Paved

# Alley: Type of alley access to property

#        Grvl     Gravel
#        Pave     Paved
#        NA       No alley access

# LotShape: General shape of property

#        Reg      Regular
#        IR1      Slightly irregular
#        IR2      Moderately Irregular
#        IR3      Irregular

# LandContour: Flatness of the property

#        Lvl      Near Flat/Level
#        Bnk      Banked - Quick and significant rise from street grade to␣
↪building
#        HLS      Hillside - Significant slope from side to side
#        Low      Depression

# Utilities: Type of utilities available

#        AllPub   All public Utilities (E,G,W,& S)
#        NoSewr   Electricity, Gas, and Water (Septic Tank)
#        NoSeWa   Electricity and Gas Only
#        ELO      Electricity only

# LotConfig: Lot configuration

#        Inside   Inside lot
#        Corner   Corner lot
#        CulDSac  Cul-de-sac
#        FR2      Frontage on 2 sides of property
#        FR3      Frontage on 3 sides of property
```

```
# LandSlope: Slope of property

#        Gtl      Gentle slope
#        Mod      Moderate Slope
#        Sev      Severe Slope

# Neighborhood: Physical locations within Ames city limits

#        Blmngtn  Bloomington Heights
#        Blueste  Bluestem
#        BrDale   Briardale
#        BrkSide  Brookside
#        ClearCr  Clear Creek
#        CollgCr  College Creek
#        Crawfor  Crawford
#        Edwards  Edwards
#        Gilbert  Gilbert
#        IDOTRR   Iowa DOT and Rail Road
#        MeadowV  Meadow Village
#        Mitchel  Mitchell
#        Names    North Ames
#        NoRidge  Northridge
#        NPkVill  Northpark Villa
#        NridgHt  Northridge Heights
#        NWAmes   Northwest Ames
#        OldTown  Old Town
#        SWISU    South & West of Iowa State University
#        Sawyer   Sawyer
#        SawyerW  Sawyer West
#        Somerst  Somerset
#        StoneBr  Stone Brook
#        Timber   Timberland
#        Veenker  Veenker

# Condition1: Proximity to various conditions

#        Artery   Adjacent to arterial street
#        Feedr    Adjacent to feeder street
#        Norm     Normal
#        RRNn     Within 200' of North-South Railroad
#        RRAn     Adjacent to North-South Railroad
#        PosN     Near positive off-site feature--park, greenbelt, etc.
#        PosA     Adjacent to postive off-site feature
#        RRNe     Within 200' of East-West Railroad
#        RRAe     Adjacent to East-West Railroad

# Condition2: Proximity to various conditions (if more than one is present)
```

```
#        Artery   Adjacent to arterial street
#        Feedr    Adjacent to feeder street
#        Norm     Normal
#        RRNn     Within 200' of North-South Railroad
#        RRAn     Adjacent to North-South Railroad
#        PosN     Near positive off-site feature--park, greenbelt, etc.
#        PosA     Adjacent to postive off-site feature
#        RRNe     Within 200' of East-West Railroad
#        RRAe     Adjacent to East-West Railroad

# BldgType: Type of dwelling

#        1Fam     Single-family Detached
#        2FmCon   Two-family Conversion; originally built as one-family dwelling
#        Duplx    Duplex
#        TwnhsE   Townhouse End Unit
#        TwnhsI   Townhouse Inside Unit

# HouseStyle: Style of dwelling

#        1Story   One story
#        1.5Fin   One and one-half story: 2nd level finished
#        1.5Unf   One and one-half story: 2nd level unfinished
#        2Story   Two story
#        2.5Fin   Two and one-half story: 2nd level finished
#        2.5Unf   Two and one-half story: 2nd level unfinished
#        SFoyer   Split Foyer
#        SLvl     Split Level

# OverallQual: Rates the overall material and finish of the house

#        10       Very Excellent
#        9        Excellent
#        8        Very Good
#        7        Good
#        6        Above Average
#        5        Average
#        4        Below Average
#        3        Fair
#        2        Poor
#        1        Very Poor

# OverallCond: Rates the overall condition of the house

#        10       Very Excellent
#        9        Excellent
```

```
#        8        Very Good
#        7        Good
#        6        Above Average
#        5        Average
#        4        Below Average
#        3        Fair
#        2        Poor
#        1        Very Poor

# YearBuilt: Original construction date

# YearRemodAdd: Remodel date (same as construction date if no remodeling or
 ↪additions)

# RoofStyle: Type of roof

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

# RoofMatl: Roof material

#        ClyTile  Clay or Tile
#        CompShg  Standard (Composite) Shingle
#        Membran  Membrane
#        Metal    Metal
#        Roll     Roll
#        Tar&Grv  Gravel & Tar
#        WdShake  Wood Shakes
#        WdShngl  Wood Shingles

# Exterior1st: Exterior covering on house

#        AsbShng  Asbestos Shingles
#        AsphShn  Asphalt Shingles
#        BrkComm  Brick Common
#        BrkFace  Brick Face
#        CBlock   Cinder Block
#        CemntBd  Cement Board
#        HdBoard  Hard Board
#        ImStucc  Imitation Stucco
#        MetalSd  Metal Siding
#        Other    Other
#        Plywood  Plywood
```

```
#        PreCast  PreCast
#        Stone    Stone
#        Stucco   Stucco
#        VinylSd  Vinyl Siding
#        Wd Sdng  Wood Siding
#        WdShing  Wood Shingles

# Exterior2nd: Exterior covering on house (if more than one material)

#        AsbShng  Asbestos Shingles
#        AsphShn  Asphalt Shingles
#        BrkComm  Brick Common
#        BrkFace  Brick Face
#        CBlock   Cinder Block
#        CemntBd  Cement Board
#        HdBoard  Hard Board
#        ImStucc  Imitation Stucco
#        MetalSd  Metal Siding
#        Other    Other
#        Plywood  Plywood
#        PreCast  PreCast
#        Stone    Stone
#        Stucco   Stucco
#        VinylSd  Vinyl Siding
#        Wd Sdng  Wood Siding
#        WdShing  Wood Shingles

# MasVnrType: Masonry veneer type

#        BrkCmn   Brick Common
#        BrkFace  Brick Face
#        CBlock   Cinder Block
#        None     None
#        Stone    Stone

# MasVnrArea: Masonry veneer area in square feet

# ExterQual: Evaluates the quality of the material on the exterior

#        Ex       Excellent
#        Gd       Good
#        TA       Average/Typical
#        Fa       Fair
#        Po       Poor

# ExterCond: Evaluates the present condition of the material on the exterior
```

```
#        Ex       Excellent
#        Gd       Good
#        TA       Average/Typical
#        Fa       Fair
#        Po       Poor

# Foundation: Type of foundation

#        BrkTil   Brick & Tile
#        CBlock   Cinder Block
#        PConc    Poured Contrete
#        Slab     Slab
#        Stone    Stone
#        Wood     Wood

# BsmtQual: Evaluates the height of the basement

#        Ex       Excellent (100+ inches)
#        Gd       Good (90-99 inches)
#        TA       Typical (80-89 inches)
#        Fa       Fair (70-79 inches)
#        Po       Poor (<70 inches
#        NA       No Basement

# BsmtCond: Evaluates the general condition of the basement

#        Ex       Excellent
#        Gd       Good
#        TA       Typical - slight dampness allowed
#        Fa       Fair - dampness or some cracking or settling
#        Po       Poor - Severe cracking, settling, or wetness
#        NA       No Basement

# BsmtExposure: Refers to walkout or garden level walls

#        Gd       Good Exposure
#        Av       Average Exposure (split levels or foyers typically score
 ↪average or above)
#        Mn       Mimimum Exposure
#        No       No Exposure
#        NA       No Basement

# BsmtFinType1: Rating of basement finished area

#        GLQ      Good Living Quarters
#        ALQ      Average Living Quarters
#        BLQ      Below Average Living Quarters
```

```
#        Rec      Average Rec Room
#        LwQ      Low Quality
#        Unf      Unfinshed
#        NA       No Basement

# BsmtFinSF1: Type 1 finished square feet

# BsmtFinType2: Rating of basement finished area (if multiple types)

#        GLQ      Good Living Quarters
#        ALQ      Average Living Quarters
#        BLQ      Below Average Living Quarters
#        Rec      Average Rec Room
#        LwQ      Low Quality
#        Unf      Unfinshed
#        NA       No Basement

# BsmtFinSF2: Type 2 finished square feet

# BsmtUnfSF: Unfinished square feet of basement area

# TotalBsmtSF: Total square feet of basement area

# Heating: Type of heating

#        Floor    Floor Furnace
#        GasA     Gas forced warm air furnace
#        GasW     Gas hot water or steam heat
#        Grav     Gravity furnace
#        OthW     Hot water or steam heat other than gas
#        Wall     Wall furnace

# HeatingQC: Heating quality and condition

#        Ex       Excellent
#        Gd       Good
#        TA       Average/Typical
#        Fa       Fair
#        Po       Poor

# CentralAir: Central air conditioning

#        N        No
#        Y        Yes

# Electrical: Electrical system
```

```
#        SBrkr    Standard Circuit Breakers & Romex
#        FuseA    Fuse Box over 60 AMP and all Romex wiring (Average)
#        FuseF    60 AMP Fuse Box and mostly Romex wiring (Fair)
#        FuseP    60 AMP Fuse Box and mostly knob & tube wiring (poor)
#        Mix      Mixed

# 1stFlrSF: First Floor square feet

# 2ndFlrSF: Second floor square feet

# LowQualFinSF: Low quality finished square feet (all floors)

# GrLivArea: Above grade (ground) living area square feet

# BsmtFullBath: Basement full bathrooms

# BsmtHalfBath: Basement half bathrooms

# FullBath: Full bathrooms above grade

# HalfBath: Half baths above grade

# Bedroom: Bedrooms above grade (does NOT include basement bedrooms)

# Kitchen: Kitchens above grade

# KitchenQual: Kitchen quality

#        Ex       Excellent
#        Gd       Good
#        TA       Typical/Average
#        Fa       Fair
#        Po       Poor

# TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)

# Functional: Home functionality (Assume typical unless deductions are
 ↪warranted)

#        Typ      Typical Functionality
#        Min1     Minor Deductions 1
#        Min2     Minor Deductions 2
#        Mod      Moderate Deductions
#        Maj1     Major Deductions 1
#        Maj2     Major Deductions 2
#        Sev      Severely Damaged
#        Sal      Salvage only
```

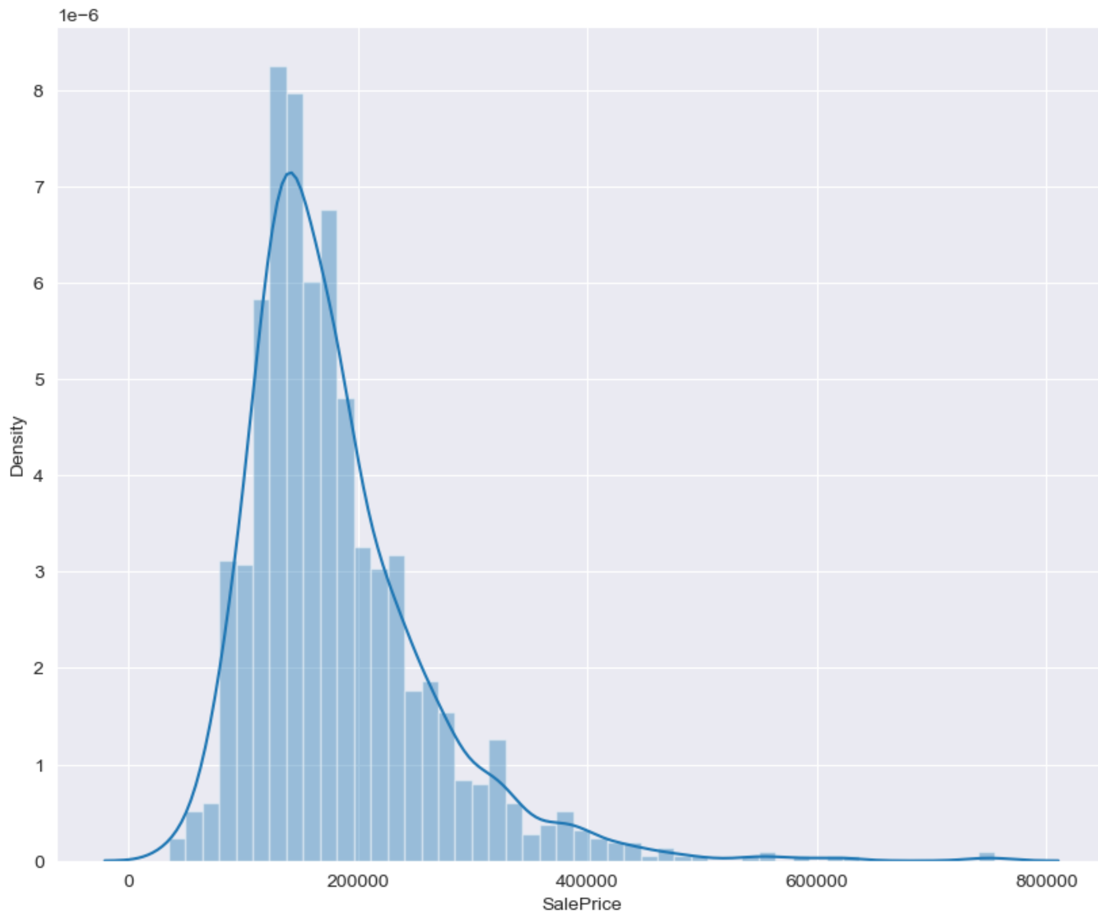
```
# Fireplaces: Number of fireplaces

# FireplaceQu: Fireplace quality

#        Ex       Excellent - Exceptional Masonry Fireplace
#        Gd       Good - Masonry Fireplace in main level
#        TA       Average - Prefabricated Fireplace in main living area or␣
↪Masonry Fireplace in basement
#        Fa       Fair - Prefabricated Fireplace in basement
#        Po       Poor - Ben Franklin Stove
#        NA       No Fireplace

# GarageType: Garage location

#        2Types   More than one type of garage
#        Attchd   Attached to home
#        Basment  Basement Garage
#        BuiltIn  Built-In (Garage part of house - typically has room above␣
↪garage)
#        CarPort  Car Port
#        Detchd   Detached from home
#        NA       No Garage

# GarageYrBlt: Year garage was built

# GarageFinish: Interior finish of the garage

#        Fin      Finished
#        RFn      Rough Finished
#        Unf      Unfinished
#        NA       No Garage

# GarageCars: Size of garage in car capacity

# GarageArea: Size of garage in square feet

# GarageQual: Garage quality

#        Ex       Excellent
#        Gd       Good
#        TA       Typical/Average
#        Fa       Fair
#        Po       Poor
#        NA       No Garage

# GarageCond: Garage condition
```

```
#        Ex       Excellent
#        Gd       Good
#        TA       Typical/Average
#        Fa       Fair
#        Po       Poor
#        NA       No Garage

# PavedDrive: Paved driveway

#        Y        Paved
#        P        Partial Pavement
#        N        Dirt/Gravel

# WoodDeckSF: Wood deck area in square feet

# OpenPorchSF: Open porch area in square feet

# EnclosedPorch: Enclosed porch area in square feet

# 3SsnPorch: Three season porch area in square feet

# ScreenPorch: Screen porch area in square feet

# PoolArea: Pool area in square feet

# PoolQC: Pool quality

#        Ex       Excellent
#        Gd       Good
#        TA       Average/Typical
#        Fa       Fair
#        NA       No Pool

# Fence: Fence quality

#        GdPrv    Good Privacy
#        MnPrv    Minimum Privacy
#        GdWo     Good Wood
#        MnWw     Minimum Wood/Wire
#        NA       No Fence

# MiscFeature: Miscellaneous feature not covered in other categories

#        Elev     Elevator
#        Gar2     2nd Garage (if not described in garage section)
#        Othr     Other
```

```
#        Shed     Shed (over 100 SF)
#        TenC     Tennis Court
#        NA       None

# MiscVal: $Value of miscellaneous feature

# MoSold: Month Sold (MM)

# YrSold: Year Sold (YYYY)

# SaleType: Type of sale

#        WD       Warranty Deed - Conventional
#        CWD      Warranty Deed - Cash
#        VWD      Warranty Deed - VA Loan
#        New      Home just constructed and sold
#        COD      Court Officer Deed/Estate
#        Con      Contract 15% Down payment regular terms
#        ConLw    Contract Low Down payment and low interest
#        ConLI    Contract Low Interest
#        ConLD    Contract Low Down
#        Oth      Other

# SaleCondition: Condition of sale

#        Normal   Normal Sale
#        Abnorml  Abnormal Sale -  trade, foreclosure, short sale
#        AdjLand  Adjoining Land Purchase
#        Alloca   Allocation - two linked properties with separate deeds,␣
 ↪typically condo with a garage unit
#        Family   Sale between family members
#        Partial  Home was not completed when last assessed (associated with␣
 ↪New Homes)
```

[330]:
```
# Columns MSSubClass, OverallQual, OverallCond need to be converted to object␣
 ↪type
# Column LotFrontage and MasVnrArea needs to be converted to numeric type.

## Convert three columns to 'object' type as mentioned above
housing[['MSSubClass', 'OverallQual', 'OverallCond']] = housing[['MSSubClass',␣
 ↪'OverallQual', 'OverallCond']].astype('object')
housing['LotFrontage'] = pd.to_numeric(housing['LotFrontage'], errors='coerce')
housing['MasVnrArea'] = pd.to_numeric(housing['MasVnrArea'], errors='coerce')
```

[331]:
```
#Analyse the target variable 'SalePrice'
plt.figure(figsize=[10,8])
sns.distplot(housing['SalePrice']);
```



[332]:
```
#Sale price is Right Skewed

housing['SalePrice'].skew()
```

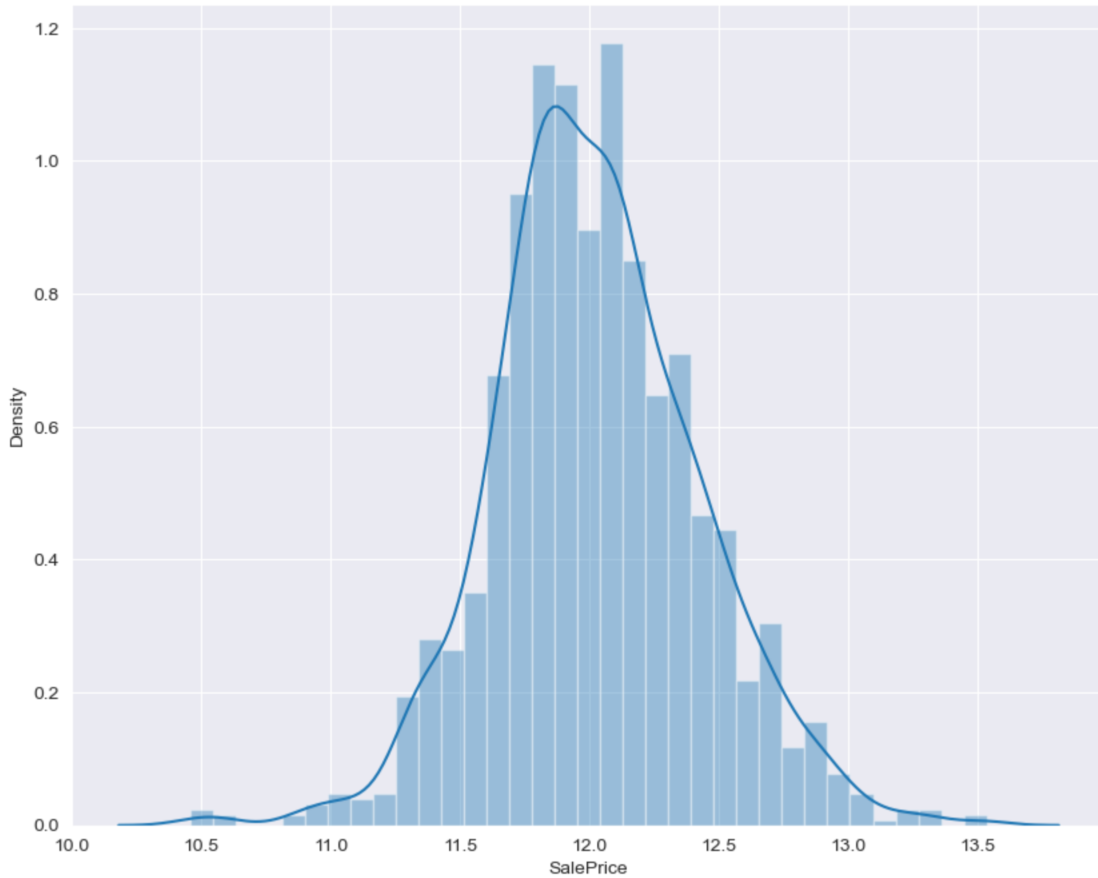
[332]: 1.8828757597682129

[333]:
```
#We should log transform the target variable
housing['SalePrice'] = np.log(housing['SalePrice'])
```

[334]:
```
#Check again the skew
housing['SalePrice'].skew()
```

[334]: 0.12133506220520406

[335]:
```
#Plot 'SalePrice' distribution plot again to check
plt.figure(figsize=[10,8])
sns.distplot(housing['SalePrice']);
```



1.2.1 Step 3:

- Data Exploration

To perform linear regression, the (numeric) target variable should be linearly related to at least one another numeric variable. Let's see whether that's true in this case.

We'll first subset the list of all (independent) numeric variables, and then make a pairwise plot.

[336]:
```
# all numeric (float and int) variables in the dataset
housing_numeric = housing.select_dtypes(include=['float64', 'int64'])
housing_numeric.head()
```

[336]:
```
   Id  LotFrontage  LotArea  YearBuilt  YearRemodAdd  MasVnrArea  BsmtFinSF1  \
0   1         65.0     8450       2003          2003       196.0         706
1   2         80.0     9600       1976          1976         0.0         978
2   3         68.0    11250       2001          2002       162.0         486
3   4         60.0     9550       1915          1970         0.0         216
4   5         84.0    14260       2000          2000       350.0         655
```

```
   BsmtFinSF2  BsmtUnfSF  TotalBsmtSF  …  WoodDeckSF  OpenPorchSF  \
0           0        150          856  …           0           61
1           0        284         1262  …         298            0
2           0        434          920  …           0           42
3           0        540          756  …           0           35
4           0        490         1145  …         192           84

   EnclosedPorch  3SsnPorch  ScreenPorch  PoolArea  MiscVal  MoSold  YrSold  \
0              0          0            0         0        0       2    2008
1              0          0            0         0        0       5    2007
2              0          0            0         0        0       9    2008
3            272          0            0         0        0       2    2006
4              0          0            0         0        0      12    2008

   SalePrice
0  12.247694
1  12.109011
2  12.317167
3  11.849398
4  12.429216

[5 rows x 35 columns]
```

[337]:

```python
#Drop Id

housing_numeric = housing_numeric.drop(['Id'], axis = 1)
housing_numeric.head()
```

[337]:

```
   LotFrontage  LotArea  YearBuilt  YearRemodAdd  MasVnrArea  BsmtFinSF1  \
0         65.0     8450       2003          2003       196.0         706
1         80.0     9600       1976          1976         0.0         978
2         68.0    11250       2001          2002       162.0         486
3         60.0     9550       1915          1970         0.0         216
4         84.0    14260       2000          2000       350.0         655

   BsmtFinSF2  BsmtUnfSF  TotalBsmtSF  1stFlrSF  …  WoodDeckSF  OpenPorchSF  \
0           0        150          856       856  …           0           61
1           0        284         1262      1262  …         298            0
2           0        434          920       920  …           0           42
3           0        540          756       961  …           0           35
4           0        490         1145      1145  …         192           84

   EnclosedPorch  3SsnPorch  ScreenPorch  PoolArea  MiscVal  MoSold  YrSold  \
0              0          0            0         0        0       2    2008
1              0          0            0         0        0       5    2007
2              0          0            0         0        0       9    2008
3            272          0            0         0        0       2    2006
```

```
4              0          0            0         0         0      12     2008

   SalePrice
0  12.247694
1  12.109011
2  12.317167
3  11.849398
4  12.429216

[5 rows x 34 columns]
```

[338]:

```
# correlation matrix
cor = housing_numeric.corr()
cor
```

[338]:

```
               LotFrontage   LotArea  YearBuilt  YearRemodAdd  MasVnrArea  \
LotFrontage       1.000000  0.426095   0.123349      0.088866    0.193458
LotArea           0.426095  1.000000   0.014228      0.013788    0.104160
YearBuilt         0.123349  0.014228   1.000000      0.592855    0.315707
YearRemodAdd      0.088866  0.013788   0.592855      1.000000    0.179618
MasVnrArea        0.193458  0.104160   0.315707      0.179618    1.000000
BsmtFinSF1        0.233633  0.214103   0.249503      0.128451    0.264736
BsmtFinSF2        0.049900  0.111170  -0.049107     -0.067759   -0.072319
BsmtUnfSF         0.132644 -0.002618   0.149040      0.181133    0.114442
TotalBsmtSF       0.392075  0.260833   0.391452      0.291066    0.363936
1stFlrSF          0.457181  0.299475   0.281986      0.240379    0.344501
2ndFlrSF          0.080177  0.050986   0.010308      0.140024    0.174561
LowQualFinSF      0.038469  0.004779  -0.183784     -0.062419   -0.069071
GrLivArea         0.402797  0.263116   0.199010      0.287389    0.390857
BsmtFullBath      0.100949  0.158155   0.187599      0.119470    0.085310
BsmtHalfBath     -0.007234  0.048046  -0.038162     -0.012337    0.026673
FullBath          0.198769  0.126031   0.468271      0.439046    0.276833
HalfBath          0.053532  0.014259   0.242656      0.183331    0.201444
BedroomAbvGr      0.263170  0.119690  -0.070651     -0.040581    0.102821
KitchenAbvGr     -0.006069 -0.017784  -0.174800     -0.149598   -0.037610
TotRmsAbvGrd      0.352096  0.190015   0.095589      0.191740    0.280682
Fireplaces        0.266639  0.271364   0.147716      0.112581    0.249070
GarageYrBlt       0.070250 -0.024947   0.825667      0.642277    0.252691
GarageCars        0.285691  0.154871   0.537850      0.420622    0.364204
GarageArea        0.344997  0.180403   0.478954      0.371600    0.373066
WoodDeckSF        0.088521  0.171698   0.224880      0.205726    0.159718
OpenPorchSF       0.151972  0.084774   0.188686      0.226298    0.125703
EnclosedPorch     0.010700 -0.018340  -0.387268     -0.193919   -0.110204
3SsnPorch         0.070029  0.020423   0.031355      0.045286    0.018796
ScreenPorch       0.041383  0.043160  -0.050364     -0.038740    0.061466
PoolArea          0.206167  0.077672   0.004950      0.005829    0.011723
MiscVal           0.003368  0.038068  -0.034383     -0.010286   -0.029815
```

MoSold	0.011200	0.001205	0.012398	0.021490	-0.005965
YrSold	0.007450	-0.014261	-0.013618	0.035743	-0.008201
SalePrice	0.355878	0.257320	0.586570	0.565608	0.430809

	BsmtFinSF1	BsmtFinSF2	BsmtUnfSF	TotalBsmtSF	1stFlrSF	… \
LotFrontage	0.233633	0.049900	0.132644	0.392075	0.457181	…
LotArea	0.214103	0.111170	-0.002618	0.260833	0.299475	…
YearBuilt	0.249503	-0.049107	0.149040	0.391452	0.281986	…
YearRemodAdd	0.128451	-0.067759	0.181133	0.291066	0.240379	…
MasVnrArea	0.264736	-0.072319	0.114442	0.363936	0.344501	…
BsmtFinSF1	1.000000	-0.050117	-0.495251	0.522396	0.445863	…
BsmtFinSF2	-0.050117	1.000000	-0.209294	0.104810	0.097117	…
BsmtUnfSF	-0.495251	-0.209294	1.000000	0.415360	0.317987	…
TotalBsmtSF	0.522396	0.104810	0.415360	1.000000	0.819530	…
1stFlrSF	0.445863	0.097117	0.317987	0.819530	1.000000	…
2ndFlrSF	-0.137079	-0.099260	0.004469	-0.174512	-0.202646	…
LowQualFinSF	-0.064503	0.014807	0.028167	-0.033245	-0.014241	…
GrLivArea	0.208171	-0.009640	0.240257	0.454868	0.566024	…
BsmtFullBath	0.649212	0.158678	-0.422900	0.307351	0.244671	…
BsmtHalfBath	0.067418	0.070948	-0.095804	-0.000315	0.001956	…
FullBath	0.058543	-0.076444	0.288886	0.323722	0.380637	…
HalfBath	0.004262	-0.032148	-0.041118	-0.048804	-0.119916	…
BedroomAbvGr	-0.107355	-0.015728	0.166643	0.050450	0.127401	…
KitchenAbvGr	-0.081007	-0.040751	0.030086	-0.068901	0.068101	…
TotRmsAbvGrd	0.044316	-0.035227	0.250647	0.285573	0.409516	…
Fireplaces	0.260011	0.046921	0.051575	0.339519	0.410531	…
GarageYrBlt	0.153484	-0.088011	0.190708	0.322445	0.233449	…
GarageCars	0.224054	-0.038264	0.214175	0.434585	0.439317	…
GarageArea	0.296970	-0.018227	0.183303	0.486665	0.489782	…
WoodDeckSF	0.204306	0.067898	-0.005316	0.232019	0.235459	…
OpenPorchSF	0.111761	0.003093	0.129005	0.247264	0.211671	…
EnclosedPorch	-0.102303	0.036543	-0.002538	-0.095478	-0.065292	…
3SsnPorch	0.026451	-0.029993	0.020764	0.037384	0.056104	…
ScreenPorch	0.062021	0.088871	-0.012579	0.084489	0.088758	…
PoolArea	0.140491	0.041709	-0.035092	0.126053	0.131525	…
MiscVal	0.003571	0.004940	-0.023837	-0.018479	-0.021096	…
MoSold	-0.015727	-0.015211	0.034888	0.013196	0.031372	…
YrSold	0.014359	0.031706	-0.041258	-0.014969	-0.013604	…
SalePrice	0.372023	0.004832	0.221985	0.612134	0.596981	…

	WoodDeckSF	OpenPorchSF	EnclosedPorch	3SsnPorch	ScreenPorch \
LotFrontage	0.088521	0.151972	0.010700	0.070029	0.041383
LotArea	0.171698	0.084774	-0.018340	0.020423	0.043160
YearBuilt	0.224880	0.188686	-0.387268	0.031355	-0.050364
YearRemodAdd	0.205726	0.226298	-0.193919	0.045286	-0.038740
MasVnrArea	0.159718	0.125703	-0.110204	0.018796	0.061466
BsmtFinSF1	0.204306	0.111761	-0.102303	0.026451	0.062021

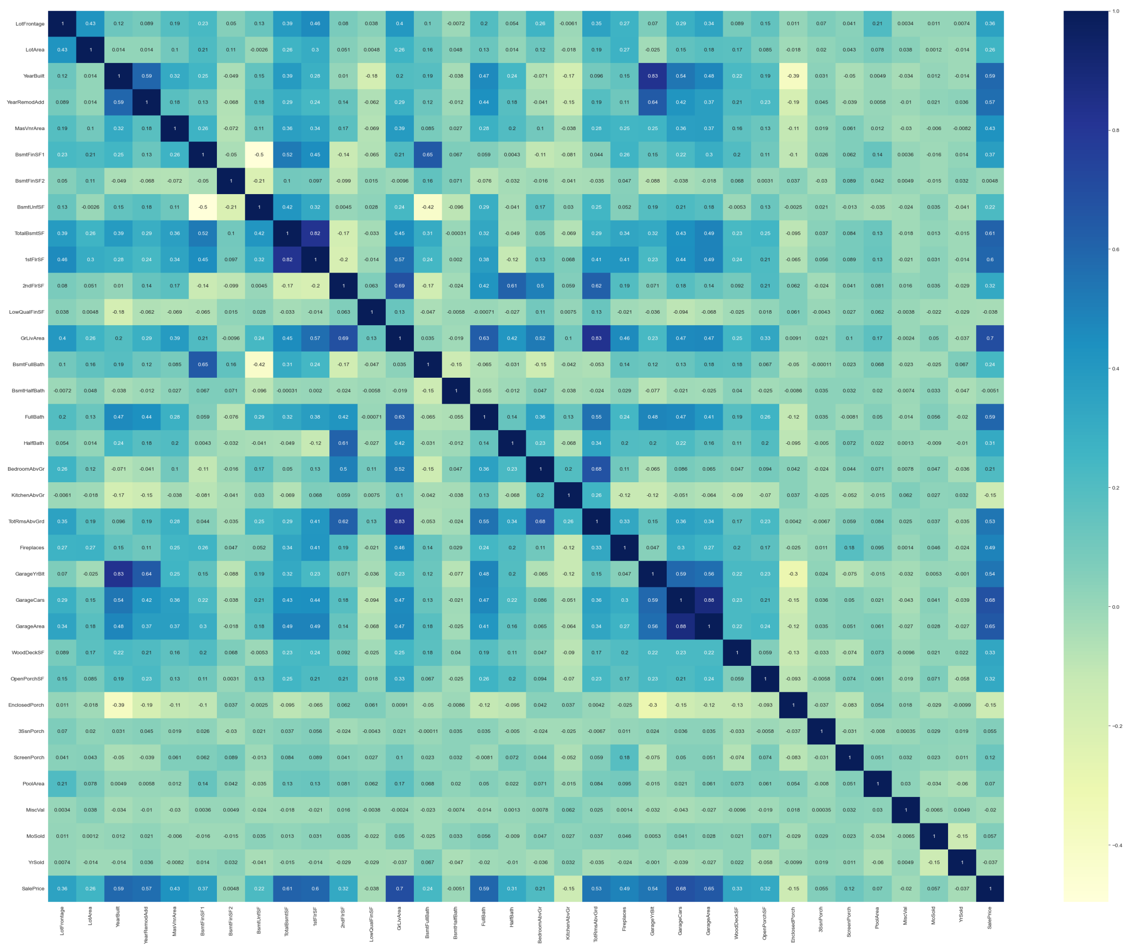
BsmtFinSF2	0.067898	0.003093	0.036543	-0.029993	0.088871
BsmtUnfSF	-0.005316	0.129005	-0.002538	0.020764	-0.012579
TotalBsmtSF	0.232019	0.247264	-0.095478	0.037384	0.084489
1stFlrSF	0.235459	0.211671	-0.065292	0.056104	0.088758
2ndFlrSF	0.092165	0.208026	0.061989	-0.024358	0.040606
LowQualFinSF	-0.025444	0.018251	0.061081	-0.004296	0.026799
GrLivArea	0.247433	0.330224	0.009113	0.020643	0.101510
BsmtFullBath	0.175315	0.067341	-0.049911	-0.000106	0.023148
BsmtHalfBath	0.040161	-0.025324	-0.008555	0.035114	0.032121
FullBath	0.187703	0.259977	-0.115093	0.035353	-0.008106
HalfBath	0.108080	0.199740	-0.095317	-0.004972	0.072426
BedroomAbvGr	0.046854	0.093810	0.041570	-0.024478	0.044300
KitchenAbvGr	-0.090130	-0.070091	0.037312	-0.024600	-0.051613
TotRmsAbvGrd	0.165984	0.234192	0.004151	-0.006683	0.059383
Fireplaces	0.200019	0.169405	-0.024822	0.011257	0.184530
GarageYrBlt	0.224577	0.228425	-0.297003	0.023544	-0.075418
GarageCars	0.226342	0.213569	-0.151434	0.035765	0.050494
GarageArea	0.224666	0.241435	-0.121777	0.035087	0.051412
WoodDeckSF	1.000000	0.058661	-0.125989	-0.032771	-0.074181
OpenPorchSF	0.058661	1.000000	-0.093079	-0.005842	0.074304
EnclosedPorch	-0.125989	-0.093079	1.000000	-0.037305	-0.082864
3SsnPorch	-0.032771	-0.005842	-0.037305	1.000000	-0.031436
ScreenPorch	-0.074181	0.074304	-0.082864	-0.031436	1.000000
PoolArea	0.073378	0.060762	0.054203	-0.007992	0.051307
MiscVal	-0.009551	-0.018584	0.018361	0.000354	0.031946
MoSold	0.021011	0.071255	-0.028887	0.029474	0.023217
YrSold	0.022270	-0.057619	-0.009916	0.018645	0.010694
SalePrice	0.334135	0.321053	-0.149050	0.054900	0.121208

	PoolArea	MiscVal	MoSold	YrSold	SalePrice
LotFrontage	0.206167	0.003368	0.011200	0.007450	0.355878
LotArea	0.077672	0.038068	0.001205	-0.014261	0.257320
YearBuilt	0.004950	-0.034383	0.012398	-0.013618	0.586570
YearRemodAdd	0.005829	-0.010286	0.021490	0.035743	0.565608
MasVnrArea	0.011723	-0.029815	-0.005965	-0.008201	0.430809
BsmtFinSF1	0.140491	0.003571	-0.015727	0.014359	0.372023
BsmtFinSF2	0.041709	0.004940	-0.015211	0.031706	0.004832
BsmtUnfSF	-0.035092	-0.023837	0.034888	-0.041258	0.221985
TotalBsmtSF	0.126053	-0.018479	0.013196	-0.014969	0.612134
1stFlrSF	0.131525	-0.021096	0.031372	-0.013604	0.596981
2ndFlrSF	0.081487	0.016197	0.035164	-0.028700	0.319300
LowQualFinSF	0.062157	-0.003793	-0.022174	-0.028921	-0.037963
GrLivArea	0.170205	-0.002416	0.050240	-0.036526	0.700927
BsmtFullBath	0.067616	-0.023047	-0.025361	0.067049	0.236224
BsmtHalfBath	0.020025	-0.007367	0.032873	-0.046524	-0.005149
FullBath	0.049604	-0.014290	0.055872	-0.019669	0.594771
HalfBath	0.022381	0.001290	-0.009050	-0.010269	0.313982

```
BedroomAbvGr   0.070703  0.007767  0.046544 -0.036014   0.209044
KitchenAbvGr  -0.014525  0.062341  0.026589  0.031687  -0.147548
TotRmsAbvGrd   0.083757  0.024763  0.036907 -0.034516   0.534422
Fireplaces     0.095074  0.001409  0.046357 -0.024096   0.489449
GarageYrBlt   -0.014501 -0.032417  0.005337 -0.001014   0.541073
GarageCars     0.020934 -0.043080  0.040522 -0.039117   0.680625
GarageArea     0.061047 -0.027400  0.027974 -0.027378   0.650888
WoodDeckSF     0.073378 -0.009551  0.021011  0.022270   0.334135
OpenPorchSF    0.060762 -0.018584  0.071255 -0.057619   0.321053
EnclosedPorch  0.054203  0.018361 -0.028887 -0.009916  -0.149050
3SsnPorch     -0.007992  0.000354  0.029474  0.018645   0.054900
ScreenPorch    0.051307  0.031946  0.023217  0.010694   0.121208
PoolArea       1.000000  0.029669 -0.033737 -0.059689   0.069798
MiscVal        0.029669  1.000000 -0.006495  0.004906  -0.020021
MoSold        -0.033737 -0.006495  1.000000 -0.145721   0.057329
YrSold        -0.059689  0.004906 -0.145721  1.000000  -0.037263
SalePrice      0.069798 -0.020021  0.057329 -0.037263   1.000000

[34 rows x 34 columns]
```

[339]:

```python
# plotting correlations on a heatmap

# figure size
plt.figure(figsize=(40,30))

# heatmap
sns.heatmap(cor, cmap="YlGnBu", annot = True)
plt.show()
```



Inference from HeatMap:

- Sale Price is highly positively correlated with OverallQual, GrLivArea, TotalBsmtSF, 1stFlrSF, GarageCars, GarageArea
- Sale Price is negatively correlated with OverallCond, BsmtFinSF2, LowQualFinSF, KitchenAbvGr, EnclosedPorch, MscVal, YrSold

1.3 Step 4:

- Data Cleaning

[340]:
```
## Create new column for the age of the house
housing['Age'] = housing['YrSold'] - housing['YearBuilt']
```

[341]:
```
## Drop the two columns from which we created new one
housing.drop(['YrSold', 'YearBuilt'], axis=1, inplace=True)
```

[342]:
```
#Drop Alley, FirePlaceQu, PoolQc, Fence, MiscFeature
```

```python
housing = housing.drop(['Alley', 'FireplaceQu', 'PoolQC', 'Fence',␣
↪'MiscFeature'], axis = 1)
housing.head()
```

[342]:
```
   Id MSSubClass MSZoning  LotFrontage  LotArea Street LotShape LandContour  \
0   1         60       RL         65.0     8450   Pave      Reg         Lvl
1   2         20       RL         80.0     9600   Pave      Reg         Lvl
2   3         60       RL         68.0    11250   Pave      IR1         Lvl
3   4         70       RL         60.0     9550   Pave      IR1         Lvl
4   5         60       RL         84.0    14260   Pave      IR1         Lvl

  Utilities LotConfig  … EnclosedPorch 3SsnPorch ScreenPorch PoolArea  \
0    AllPub    Inside  …             0         0           0        0
1    AllPub       FR2  …             0         0           0        0
2    AllPub    Inside  …             0         0           0        0
3    AllPub    Corner  …           272         0           0        0
4    AllPub       FR2  …             0         0           0        0

  MiscVal MoSold SaleType SaleCondition  SalePrice Age
0       0      2       WD        Normal  12.247694   5
1       0      5       WD        Normal  12.109011  31
2       0      9       WD        Normal  12.317167   7
3       0      2       WD       Abnorml  11.849398  91
4       0     12       WD        Normal  12.429216   8

[5 rows x 75 columns]
```

[343]:
```python
housing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 75 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1460 non-null   int64
 1   MSSubClass     1460 non-null   object
 2   MSZoning       1460 non-null   object
 3   LotFrontage    1201 non-null   float64
 4   LotArea        1460 non-null   int64
 5   Street         1460 non-null   object
 6   LotShape       1460 non-null   object
 7   LandContour    1460 non-null   object
 8   Utilities      1460 non-null   object
 9   LotConfig      1460 non-null   object
 10  LandSlope      1460 non-null   object
 11  Neighborhood   1460 non-null   object
 12  Condition1     1460 non-null   object
 13  Condition2     1460 non-null   object
```

```
14  BldgType       1460 non-null   object
15  HouseStyle     1460 non-null   object
16  OverallQual    1460 non-null   object
17  OverallCond    1460 non-null   object
18  YearRemodAdd   1460 non-null   int64
19  RoofStyle      1460 non-null   object
20  RoofMatl       1460 non-null   object
21  Exterior1st    1460 non-null   object
22  Exterior2nd    1460 non-null   object
23  MasVnrType     1452 non-null   object
24  MasVnrArea     1452 non-null   float64
25  ExterQual      1460 non-null   object
26  ExterCond      1460 non-null   object
27  Foundation     1460 non-null   object
28  BsmtQual       1423 non-null   object
29  BsmtCond       1423 non-null   object
30  BsmtExposure   1422 non-null   object
31  BsmtFinType1   1423 non-null   object
32  BsmtFinSF1     1460 non-null   int64
33  BsmtFinType2   1422 non-null   object
34  BsmtFinSF2     1460 non-null   int64
35  BsmtUnfSF      1460 non-null   int64
36  TotalBsmtSF    1460 non-null   int64
37  Heating        1460 non-null   object
38  HeatingQC      1460 non-null   object
39  CentralAir     1460 non-null   object
40  Electrical     1459 non-null   object
41  1stFlrSF       1460 non-null   int64
42  2ndFlrSF       1460 non-null   int64
43  LowQualFinSF   1460 non-null   int64
44  GrLivArea      1460 non-null   int64
45  BsmtFullBath   1460 non-null   int64
46  BsmtHalfBath   1460 non-null   int64
47  FullBath       1460 non-null   int64
48  HalfBath       1460 non-null   int64
49  BedroomAbvGr   1460 non-null   int64
50  KitchenAbvGr   1460 non-null   int64
51  KitchenQual    1460 non-null   object
52  TotRmsAbvGrd   1460 non-null   int64
53  Functional     1460 non-null   object
54  Fireplaces     1460 non-null   int64
55  GarageType     1379 non-null   object
56  GarageYrBlt    1379 non-null   float64
57  GarageFinish   1379 non-null   object
58  GarageCars     1460 non-null   int64
59  GarageArea     1460 non-null   int64
60  GarageQual     1379 non-null   object
61  GarageCond     1379 non-null   object
```

```
 62  PavedDrive     1460 non-null   object
 63  WoodDeckSF     1460 non-null   int64
 64  OpenPorchSF    1460 non-null   int64
 65  EnclosedPorch  1460 non-null   int64
 66  3SsnPorch      1460 non-null   int64
 67  ScreenPorch    1460 non-null   int64
 68  PoolArea       1460 non-null   int64
 69  MiscVal        1460 non-null   int64
 70  MoSold         1460 non-null   int64
 71  SaleType       1460 non-null   object
 72  SaleCondition  1460 non-null   object
 73  SalePrice      1460 non-null   float64
 74  Age            1460 non-null   int64
dtypes: float64(4), int64(30), object(41)
memory usage: 855.6+ KB
```

[344]:
```
#Replace NA values

housing['LotFrontage'] = housing['LotFrontage'].fillna(0)
housing['MasVnrArea'] = housing['MasVnrArea'].fillna(0)
housing['GarageQual'] = housing['GarageQual'].fillna('NG') # NG -> No Garage
housing['GarageCond'] = housing['GarageCond'].fillna('NG')
housing['GarageFinish'] = housing['GarageFinish'].fillna('NG')
housing['GarageFinish'] = housing['GarageFinish'].fillna('NG')
housing['GarageType'] = housing['GarageType'].fillna('NG')
```

[345]:
```
housing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 75 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1460 non-null   int64
 1   MSSubClass     1460 non-null   object
 2   MSZoning       1460 non-null   object
 3   LotFrontage    1460 non-null   float64
 4   LotArea        1460 non-null   int64
 5   Street         1460 non-null   object
 6   LotShape       1460 non-null   object
 7   LandContour    1460 non-null   object
 8   Utilities      1460 non-null   object
 9   LotConfig      1460 non-null   object
 10  LandSlope      1460 non-null   object
 11  Neighborhood   1460 non-null   object
 12  Condition1     1460 non-null   object
 13  Condition2     1460 non-null   object
 14  BldgType       1460 non-null   object
```

```
15  HouseStyle     1460 non-null   object
16  OverallQual    1460 non-null   object
17  OverallCond    1460 non-null   object
18  YearRemodAdd   1460 non-null   int64
19  RoofStyle      1460 non-null   object
20  RoofMatl       1460 non-null   object
21  Exterior1st    1460 non-null   object
22  Exterior2nd    1460 non-null   object
23  MasVnrType     1452 non-null   object
24  MasVnrArea     1460 non-null   float64
25  ExterQual      1460 non-null   object
26  ExterCond      1460 non-null   object
27  Foundation     1460 non-null   object
28  BsmtQual       1423 non-null   object
29  BsmtCond       1423 non-null   object
30  BsmtExposure   1422 non-null   object
31  BsmtFinType1   1423 non-null   object
32  BsmtFinSF1     1460 non-null   int64
33  BsmtFinType2   1422 non-null   object
34  BsmtFinSF2     1460 non-null   int64
35  BsmtUnfSF      1460 non-null   int64
36  TotalBsmtSF    1460 non-null   int64
37  Heating        1460 non-null   object
38  HeatingQC      1460 non-null   object
39  CentralAir     1460 non-null   object
40  Electrical     1459 non-null   object
41  1stFlrSF       1460 non-null   int64
42  2ndFlrSF       1460 non-null   int64
43  LowQualFinSF   1460 non-null   int64
44  GrLivArea      1460 non-null   int64
45  BsmtFullBath   1460 non-null   int64
46  BsmtHalfBath   1460 non-null   int64
47  FullBath       1460 non-null   int64
48  HalfBath       1460 non-null   int64
49  BedroomAbvGr   1460 non-null   int64
50  KitchenAbvGr   1460 non-null   int64
51  KitchenQual    1460 non-null   object
52  TotRmsAbvGrd   1460 non-null   int64
53  Functional     1460 non-null   object
54  Fireplaces     1460 non-null   int64
55  GarageType     1460 non-null   object
56  GarageYrBlt    1379 non-null   float64
57  GarageFinish   1460 non-null   object
58  GarageCars     1460 non-null   int64
59  GarageArea     1460 non-null   int64
60  GarageQual     1460 non-null   object
61  GarageCond     1460 non-null   object
62  PavedDrive     1460 non-null   object
```

```
 63  WoodDeckSF     1460 non-null   int64
 64  OpenPorchSF    1460 non-null   int64
 65  EnclosedPorch  1460 non-null   int64
 66  3SsnPorch      1460 non-null   int64
 67  ScreenPorch    1460 non-null   int64
 68  PoolArea       1460 non-null   int64
 69  MiscVal        1460 non-null   int64
 70  MoSold         1460 non-null   int64
 71  SaleType       1460 non-null   object
 72  SaleCondition  1460 non-null   object
 73  SalePrice      1460 non-null   float64
 74  Age            1460 non-null   int64
dtypes: float64(4), int64(30), object(41)
memory usage: 855.6+ KB
```

- Rest all Data is fine

1.4 Step 5 : Data Preparation

- Prepare the Data and Build the Model

[389]:
```python
# split into X and y

X = housing.loc[:,␣
↪['MSZoning','LotFrontage','LotArea','Street','LotShape','LandContour','Utilities',
                   ␣
↪'LotConfig','LandSlope','Neighborhood','Condition1','Condition2','BldgType','HouseStyle',
                   ␣
↪'OverallQual','OverallCond','Age','YearRemodAdd','RoofStyle','RoofMatl','Exterior1st',
                   ␣
↪'Exterior2nd','MasVnrType','MasVnrArea','ExterQual','ExterCond','Foundation','BsmtQual','Bs
                   ␣
↪'BsmtExposure','BsmtFinType1','BsmtFinSF1','BsmtFinType2','BsmtFinSF2','BsmtUnfSF','TotalBs
                   ␣
↪'Heating','HeatingQC','CentralAir','Electrical','1stFlrSF','2ndFlrSF','LowQualFinSF','GrLiv
                   ␣
↪'BsmtFullBath','BsmtHalfBath','FullBath','HalfBath','BedroomAbvGr','KitchenAbvGr','KitchenQ
                   ␣
↪'TotRmsAbvGrd','Functional','Fireplaces','GarageType','GarageYrBlt','GarageFinish','GarageC
                   ␣
↪'GarageArea','GarageQual','GarageCond','PavedDrive','WoodDeckSF','OpenPorchSF','EnclosedPor
                   ␣
↪'3SsnPorch','ScreenPorch','PoolArea','MiscVal','MoSold','SaleType','SaleCondition'
]]

y = housing['SalePrice']
```

[390]:
```
# creating dummy variables for categorical variables

# subset all categorical variables

housing_categorical = X.select_dtypes(include=['object'])
housing_categorical.head()
```

[390]:
```
  MSZoning Street LotShape LandContour Utilities LotConfig LandSlope  \
0       RL   Pave      Reg         Lvl    AllPub    Inside       Gtl
1       RL   Pave      Reg         Lvl    AllPub       FR2       Gtl
2       RL   Pave      IR1         Lvl    AllPub    Inside       Gtl
3       RL   Pave      IR1         Lvl    AllPub    Corner       Gtl
4       RL   Pave      IR1         Lvl    AllPub       FR2       Gtl

  Neighborhood Condition1 Condition2  … Electrical KitchenQual Functional  \
0      CollgCr       Norm       Norm  …      SBrkr          Gd        Typ
1      Veenker      Feedr       Norm  …      SBrkr          TA        Typ
2      CollgCr       Norm       Norm  …      SBrkr          Gd        Typ
3      Crawfor       Norm       Norm  …      SBrkr          Gd        Typ
4      NoRidge       Norm       Norm  …      SBrkr          Gd        Typ

  GarageType GarageFinish GarageQual GarageCond PavedDrive SaleType  \
0     Attchd          RFn         TA         TA          Y       WD
1     Attchd          RFn         TA         TA          Y       WD
2     Attchd          RFn         TA         TA          Y       WD
3     Detchd          Unf         TA         TA          Y       WD
4     Attchd          RFn         TA         TA          Y       WD

  SaleCondition
0        Normal
1        Normal
2        Normal
3       Abnorml
4        Normal

[5 rows x 40 columns]
```

[391]:
```
# convert into dummies
housing_dummies = pd.get_dummies(housing_categorical, drop_first=True)
housing_dummies.head()
```

[391]:
```
   MSZoning_FV  MSZoning_RH  MSZoning_RL  MSZoning_RM  Street_Pave  \
0            0            0            1            0            1
1            0            0            1            0            1
2            0            0            1            0            1
3            0            0            1            0            1
4            0            0            1            0            1
```

```
   LotShape_IR2  LotShape_IR3  LotShape_Reg  LandContour_HLS  LandContour_Low  \
0             0             0             1                0                0
1             0             0             1                0                0
2             0             0             0                0                0
3             0             0             0                0                0
4             0             0             0                0                0

   …  SaleType_ConLI  SaleType_ConLw  SaleType_New  SaleType_Oth  \
0  …               0               0             0             0
1  …               0               0             0             0
2  …               0               0             0             0
3  …               0               0             0             0
4  …               0               0             0             0

   SaleType_WD  SaleCondition_AdjLand  SaleCondition_Alloca  \
0            1                      0                     0
1            1                      0                     0
2            1                      0                     0
3            1                      0                     0
4            1                      0                     0

   SaleCondition_Family  SaleCondition_Normal  SaleCondition_Partial
0                     0                     1                      0
1                     0                     1                      0
2                     0                     1                      0
3                     0                     0                      0
4                     0                     1                      0

[5 rows x 217 columns]
```

[392]:
```python
# drop categorical variables
X = X.drop(list(housing_categorical.columns), axis=1)
```

[393]:
```python
# concat dummy variables with X
X = pd.concat([X, housing_dummies], axis=1)
```

[394]:
```python
X.head()
```

[394]:
```
   LotFrontage  LotArea  Age  YearRemodAdd  MasVnrArea  BsmtFinSF1  \
0         65.0     8450    5          2003       196.0         706
1         80.0     9600   31          1976         0.0         978
2         68.0    11250    7          2002       162.0         486
3         60.0     9550   91          1970         0.0         216
4         84.0    14260    8          2000       350.0         655

   BsmtFinSF2  BsmtUnfSF  TotalBsmtSF  1stFlrSF  …  SaleType_ConLI  \
```

```
0             0       150        856       856  ...              0
1             0       284       1262      1262  ...              0
2             0       434        920       920  ...              0
3             0       540        756       961  ...              0
4             0       490       1145      1145  ...              0

   SaleType_ConLw  SaleType_New  SaleType_Oth  SaleType_WD  \
0               0             0             0            1
1               0             0             0            1
2               0             0             0            1
3               0             0             0            1
4               0             0             0            1

   SaleCondition_AdjLand  SaleCondition_Alloca  SaleCondition_Family  \
0                      0                     0                     0
1                      0                     0                     0
2                      0                     0                     0
3                      0                     0                     0
4                      0                     0                     0

   SaleCondition_Normal  SaleCondition_Partial
0                     1                      0
1                     1                      0
2                     1                      0
3                     0                      0
4                     1                      0

[5 rows x 249 columns]
```

[395]:

```python
# scaling the features - necessary before using Ridge or Lasso
from sklearn.preprocessing import scale

# storing column names in cols, since column names are lost after
# scaling (the df is converted to a numpy array)
cols = X.columns
X = pd.DataFrame(scale(X))
X.columns = cols
X.columns
```

[395]:

```
Index(['LotFrontage', 'LotArea', 'Age', 'YearRemodAdd', 'MasVnrArea',
       'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF',
       …
       'SaleType_ConLI', 'SaleType_ConLw', 'SaleType_New', 'SaleType_Oth',
       'SaleType_WD', 'SaleCondition_AdjLand', 'SaleCondition_Alloca',
       'SaleCondition_Family', 'SaleCondition_Normal',
       'SaleCondition_Partial'],
      dtype='object', length=249)
```

[396]:
```
X.fillna(0, inplace=True)
```

[397]:
```
# split into train and test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    train_size=0.7,
                                                    test_size = 0.3,␣
 ↪random_state=100)
```

1.5 Step 6: Model Building and Evaluation

1.5.1 A: Linear Regression

[398]:
```
# Instantiate
lm = LinearRegression()

# Fit a line
lm.fit(X_train, y_train)
```

[398]:
```
LinearRegression()
```

[399]:
```
from sklearn.metrics import r2_score, mean_squared_error
```

[400]:
```
y_pred_train = lm.predict(X_train)
y_pred_test = lm.predict(X_test)

metric = []
r2_train_lr = r2_score(y_train, y_pred_train)
print(r2_train_lr)
metric.append(r2_train_lr)

r2_test_lr = r2_score(y_test, y_pred_test)
print(r2_test_lr)
metric.append(r2_test_lr)

rss1_lr = np.sum(np.square(y_train - y_pred_train))
print(rss1_lr)
metric.append(rss1_lr)

rss2_lr = np.sum(np.square(y_test - y_pred_test))
print(rss2_lr)
metric.append(rss2_lr)

mse_train_lr = mean_squared_error(y_train, y_pred_train)
print(mse_train_lr)
metric.append(mse_train_lr**0.5)

mse_test_lr = mean_squared_error(y_test, y_pred_test)
```

```
print(mse_test_lr)
metric.append(mse_test_lr**0.5)
```

```
0.9594927883515069
-1.5501672628998327e+22
6.501093226762798
1.1171833950904488e+24
0.006367378282823504
2.550647020754449e+21
```

1.5.2 B : Ridge Regression

[401]:
```
# list of alphas to tune - if value too high it will lead to underfitting, if␣
↪it is too low,
# it will not handle the overfitting
params = {'alpha': [0.0001, 0.001, 0.01, 0.05, 0.1,
 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 2.0, 3.0,
 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 20, 50, 100, 500, 1000 ]}

ridge = Ridge()

# cross validation
folds = 5
model_cv = GridSearchCV(estimator = ridge,
                        param_grid = params,
                        scoring= 'neg_mean_absolute_error',
                        cv = folds,
                        return_train_score=True,
                        verbose = 1)
model_cv.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 28 candidates, totalling 140 fits
```

[401]:
```
GridSearchCV(cv=5, estimator=Ridge(),
             param_grid={'alpha': [0.0001, 0.001, 0.01, 0.05, 0.1, 0.2, 0.3,
                                   0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 2.0, 3.0,
                                   4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 20, 50,
                                   100, 500, 1000]},
             return_train_score=True, scoring='neg_mean_absolute_error',
             verbose=1)
```

[402]:
```
# Printing the best hyperparameter alpha
print(model_cv.best_params_)
```

```
{'alpha': 5.0}
```

[403]:
```
#Fitting Ridge model for alpha = 5 and printing coefficients which have been␣
↪penalised
```

```
alpha = 5
ridge = Ridge(alpha=alpha)

ridge.fit(X_train, y_train)
print(ridge.coef_)
```

```
[-3.41553103e-03  2.98441671e-02 -5.60868610e-02  2.40568148e-02
  1.28732532e-03  2.04065414e-02  1.06210711e-02  8.27521966e-03
  3.34559746e-02  4.16483894e-02  5.08303437e-02 -8.73594104e-04
  7.27851532e-02  1.66387881e-02  1.50167171e-03  9.96025619e-03
  4.04843054e-03  6.08946400e-03 -1.98868948e-02  6.50078356e-03
  1.09063209e-02 -5.19384682e-03  2.62446212e-02  1.27485401e-02
  1.06045617e-02  3.73799170e-03  6.23143651e-03  4.95393571e-03
  1.05801543e-02 -1.93895208e-03  1.43448630e-03  4.92936375e-05
  5.72475812e-02  2.94402908e-02  1.11529989e-01  8.27503015e-02
  6.93306907e-03  2.60648619e-03 -2.05819471e-03  3.70885196e-03
  7.20977594e-04 -1.25711958e-03  5.32831427e-03 -1.95237217e-03
  7.15790832e-03 -6.64503246e-03 -1.06101435e-03 -3.53165799e-03
  3.50124065e-03 -1.79810258e-02  1.08331167e-03 -4.36023576e-03
  1.04753118e-02  8.51235463e-03  2.60715080e-03  2.86703615e-02
 -1.14507556e-02  3.67930872e-03 -6.70923983e-04 -1.79860149e-02
 -4.66499199e-03  3.66241144e-03 -1.76311232e-03 -2.27323992e-03
  8.30645842e-03  1.64823042e-02  2.97720700e-03  9.89789958e-03
 -2.43865332e-03  4.53813299e-03  1.76805747e-02  1.26928888e-02
  8.74989262e-04  6.26691678e-03  1.13475014e-02  3.16639940e-02
  6.97925736e-03  7.65992503e-03 -1.87136183e-04  8.37384948e-03
  1.07516470e-03  4.96197660e-03  2.50932323e-03  5.49375434e-03
  7.92955427e-03 -5.46463711e-02 -1.49711195e-02 -1.77660660e-03
  4.33394155e-03 -5.29314229e-03 -2.93166593e-04 -1.24414857e-02
 -7.91115564e-03  1.34230443e-03 -9.55838657e-03 -3.74830542e-03
 -4.31029759e-04 -1.24304376e-02 -4.41095401e-03 -4.51285153e-03
 -1.91222711e-02 -1.24876145e-02 -2.44648577e-02 -1.76416674e-02
 -8.16466679e-03  1.21708071e-02  2.89772237e-02  2.74080837e-02
  1.42389669e-02  5.64342296e-04 -2.21185786e-02 -1.24217084e-02
 -1.49568368e-02  7.91961982e-03  1.44864727e-02  1.31579690e-02
  1.57739426e-02 -1.92015301e-02 -1.36122606e-03 -1.67499221e-02
  5.97629723e-03  1.75633472e-02  2.59595172e-01  5.83440575e-02
  5.65688382e-02  5.24054734e-02  1.73270963e-01  1.03893440e-01
  1.29206228e-01 -2.77621426e-04 -6.62774886e-03  2.17247987e-02
  3.76483440e-04 -3.45156273e-03  7.99243650e-03  1.84356915e-03
  6.93908568e-03  7.50860620e-03  6.35693578e-03  5.35831760e-03
  1.91666992e-02  1.38079110e-03  6.80845960e-03 -1.60394777e-04
  1.65552791e-03 -5.48526612e-03  3.76483440e-04  1.33763617e-02
 -2.78660562e-03 -1.21436782e-03  4.12633521e-03  0.00000000e+00
 -1.98415978e-03 -6.42085157e-03 -3.57951931e-03  1.21262146e-03
  1.35469473e-03 -6.21859874e-03  1.29862928e-03 -2.71618750e-03
  2.49884474e-03 -4.52846564e-03  8.27184306e-03  6.85534460e-03
 -6.12304285e-03 -7.39126193e-03  0.00000000e+00 -1.85019937e-03
```

```
 1.00047475e-02  1.22050596e-02 -7.08174854e-03  4.35011596e-03
-2.26207900e-03 -2.89726454e-04 -2.34805654e-02 -2.48920391e-02
 6.25385721e-03 -3.61657745e-03  7.91098654e-03  1.35486379e-02
-7.24314467e-04 -4.90701344e-03 -4.30245968e-03  4.73980288e-03
-6.28576395e-03 -8.46385574e-03 -1.60307436e-02 -8.62583015e-03
 1.69439807e-03 -1.58033452e-03 -2.32010941e-03  5.60380668e-03
 2.56590261e-02  2.29698817e-02  9.04745228e-04  4.20252374e-03
 1.23878266e-02  1.72366480e-04 -3.75326322e-03 -4.84545489e-03
-9.74235952e-03  1.21744134e-02  6.62027068e-03 -2.79143131e-03
 0.00000000e+00 -2.60214934e-03 -5.04081381e-03 -2.83381019e-02
-2.53320827e-02 -6.67979730e-03  4.74010826e-03  8.96691495e-03
 2.04834301e-04 -1.08127312e-02  2.08624093e-02  1.68137827e-02
-2.43567177e-04  5.21541601e-03  6.00405466e-03  8.23505664e-03
-3.29422151e-03 -3.29422151e-03 -4.53635688e-05 -6.75176837e-03
-1.68361263e-02  3.78880407e-03 -3.29422151e-03 -3.33659412e-03
-7.77913928e-03 -1.01079406e-02 -6.15368899e-03 -3.29422151e-03
 4.87308560e-03 -1.19414424e-02  3.88553815e-03  5.25527127e-03
 4.55452795e-03  4.05977219e-03  1.73993348e-02  8.93073814e-04
 2.15924992e-03  1.56192617e-02  4.39807684e-03  6.70984405e-03
 4.71334781e-03  5.66810142e-03  4.29483389e-03  1.92539914e-02
 1.54480204e-02]
```

[404]:
```
y_pred_train = ridge.predict(X_train)
y_pred_test = ridge.predict(X_test)

metric2 = []
r2_train_lr = r2_score(y_train, y_pred_train)
print(r2_train_lr)
metric2.append(r2_train_lr)

r2_test_lr = r2_score(y_test, y_pred_test)
print(r2_test_lr)
metric2.append(r2_test_lr)

rss1_lr = np.sum(np.square(y_train - y_pred_train))
print(rss1_lr)
metric2.append(rss1_lr)m

rss2_lr = np.sum(np.square(y_test - y_pred_test))
print(rss2_lr)
metric2.append(rss2_lr)

mse_train_lr = mean_squared_error(y_train, y_pred_train)
print(mse_train_lr)
metric2.append(mse_train_lr**0.5)

mse_test_lr = mean_squared_error(y_test, y_pred_test)
```

```
print(mse_test_lr)
metric2.append(mse_test_lr**0.5)
```

```
0.9571517367529878
0.8660664550646793
6.876813847147698
9.652399197712207
0.006735371054992848
0.02203744109066714
```

1.5.3 C : Lasso Regression

[405]:
```
lasso = Lasso()

# cross validation
model_cv = GridSearchCV(estimator = lasso,
                        param_grid = params,
                        scoring= 'neg_mean_absolute_error',
                        cv = folds,
                        return_train_score=True,
                        verbose = 1)

model_cv.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 28 candidates, totalling 140 fits
```

[405]:
```
GridSearchCV(cv=5, estimator=Lasso(),
             param_grid={'alpha': [0.0001, 0.001, 0.01, 0.05, 0.1, 0.2, 0.3,
                                   0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 2.0, 3.0,
                                   4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 20, 50,
                                   100, 500, 1000]},
             return_train_score=True, scoring='neg_mean_absolute_error',
             verbose=1)
```

[406]:
```
# Printing the best hyperparameter alpha
print(model_cv.best_params_)
```

```
{'alpha': 0.001}
```

[407]:
```
#Fitting Ridge model for alpha = 0.001 and printing coefficients which have␣
↪been penalised

alpha = 0.001

lasso = Lasso(alpha=alpha)

lasso.fit(X_train, y_train)
```

[407]:
```
Lasso(alpha=0.001)
```

[408]:
```
lasso.coef_
```

[408]:
```
array([-0.00000000e+00,  2.34753283e-02, -5.67332510e-02,  2.53926689e-02,
        1.46752246e-03,  1.15862770e-02,  1.36040606e-03, -0.00000000e+00,
        4.00812393e-02,  0.00000000e+00,  0.00000000e+00, -6.43766238e-03,
        1.40652154e-01,  1.88757813e-02,  0.00000000e+00,  7.98152665e-03,
        3.42634700e-03,  0.00000000e+00, -2.00459837e-02,  6.57807613e-03,
        1.31454837e-02, -0.00000000e+00,  2.67979433e-02,  8.02006393e-03,
        9.34985754e-03,  2.75704186e-03,  2.48619676e-03,  3.71348489e-03,
        8.86946087e-03, -3.33726760e-03,  6.19479177e-05, -0.00000000e+00,
        2.44860456e-02,  1.29548240e-02,  5.12576434e-02,  2.76047792e-02,
        4.84087464e-03,  1.28981423e-03, -1.64815590e-03,  1.24373869e-03,
        0.00000000e+00, -0.00000000e+00,  2.58723249e-03, -1.34346385e-03,
        7.78089356e-03, -2.65091405e-03, -0.00000000e+00, -3.54054104e-04,
        0.00000000e+00, -1.18824761e-02,  0.00000000e+00, -4.00646465e-03,
        7.39407653e-03,  8.66036709e-03,  0.00000000e+00,  2.66538775e-02,
       -1.23935836e-02, -0.00000000e+00, -4.27754681e-03, -1.65810434e-02,
       -4.24579961e-03,  0.00000000e+00, -0.00000000e+00, -7.85125904e-04,
        4.51325933e-03,  1.48918468e-02, -0.00000000e+00,  8.55247347e-03,
       -3.07792847e-03,  3.39085163e-04,  1.69125974e-02,  8.81919328e-03,
       -0.00000000e+00,  4.16849704e-03,  0.00000000e+00,  1.82368474e-02,
        2.68739740e-03,  2.14670930e-03, -3.30842226e-03,  2.51524392e-03,
       -0.00000000e+00,  3.65212351e-03, -0.00000000e+00,  3.10382103e-03,
        6.30666272e-03, -5.45605712e-02, -7.32245732e-03, -1.18807075e-03,
        2.01110031e-03, -3.06314809e-03, -0.00000000e+00, -1.42597850e-02,
       -8.31067913e-03,  2.13559150e-03, -0.00000000e+00, -1.24631703e-03,
        0.00000000e+00, -2.62385689e-04, -1.88157483e-04, -0.00000000e+00,
       -2.06346305e-02, -9.07824123e-03, -1.84434361e-02, -8.98222167e-03,
       -0.00000000e+00,  1.81006523e-02,  3.64372362e-02,  3.32718277e-02,
        1.60521042e-02, -0.00000000e+00, -2.39709956e-02, -1.38794299e-02,
       -2.29020474e-02,  0.00000000e+00,  1.00861690e-02,  1.00880293e-02,
        1.16170730e-02, -1.18305350e-03,  4.36957128e-04,  0.00000000e+00,
        6.20531088e-03,  1.01174643e-02,  2.49690438e-01,  5.40036266e-02,
        5.30412904e-02,  5.00634644e-02,  1.68176206e-01,  1.03697938e-01,
        1.24946419e-01, -0.00000000e+00, -6.55025573e-03,  1.35180129e-02,
        0.00000000e+00,  0.00000000e+00, -1.35036425e-03,  4.58852340e-04,
        0.00000000e+00, -0.00000000e+00,  1.19124066e-03, -0.00000000e+00,
        3.61125483e-03, -3.98191655e-03,  7.83037574e-04, -0.00000000e+00,
       -0.00000000e+00, -0.00000000e+00,  0.00000000e+00,  3.61401779e-03,
       -0.00000000e+00, -0.00000000e+00,  6.49561039e-04,  0.00000000e+00,
       -0.00000000e+00, -1.68550531e-04, -1.05294499e-03,  3.68684185e-03,
       -0.00000000e+00, -3.63278892e-03,  0.00000000e+00, -1.82677155e-03,
        8.92155047e-04, -9.07675747e-03,  0.00000000e+00, -3.41269118e-03,
       -3.21451510e-03, -0.00000000e+00,  0.00000000e+00,  4.25353491e-03,
        0.00000000e+00,  6.49498703e-03, -6.40112837e-04,  4.46504406e-03,
       -1.69334246e-03,  0.00000000e+00, -1.46711153e-02, -1.40164477e-02,
        5.43186635e-03, -5.70824767e-03,  8.07733301e-03,  1.57833269e-02,
```

```
-0.00000000e+00, -2.86422057e-03, -0.00000000e+00,  7.80053298e-03,
-6.96296476e-04, -3.82494656e-03, -1.04495430e-02, -6.70179553e-03,
 2.88257305e-03, -0.00000000e+00, -0.00000000e+00,  2.19132010e-03,
 0.00000000e+00,  2.34199939e-03, -7.75748305e-03, -1.51874875e-03,
 0.00000000e+00, -0.00000000e+00, -1.98152852e-03, -1.75587994e-03,
-7.06708689e-03,  1.55860346e-02,  5.53479361e-03, -2.79738471e-03,
 0.00000000e+00, -0.00000000e+00, -1.42407243e-04, -1.31657836e-02,
-1.07536166e-02, -6.82264165e-03,  0.00000000e+00,  1.98333856e-03,
-0.00000000e+00, -9.42499372e-03,  1.28696897e-02,  6.82511826e-03,
-2.02714679e-03,  0.00000000e+00,  0.00000000e+00, -0.00000000e+00,
-0.00000000e+00, -0.00000000e+00, -0.00000000e+00, -4.58668455e-03,
-8.36368371e-03,  4.54835313e-03, -2.68753418e-04, -0.00000000e+00,
 0.00000000e+00, -1.64574470e-03, -4.62011943e-04, -0.00000000e+00,
 4.69604908e-03,  5.24547186e-03,  1.97651445e-03,  2.20770795e-03,
 2.65920340e-03,  1.84067530e-03,  1.21490703e-02, -0.00000000e+00,
 1.00145994e-03,  2.48336896e-02,  2.04680472e-03,  6.10812987e-05,
 4.46232755e-03,  1.02875254e-03,  1.62498497e-03,  1.78161037e-02,
 0.00000000e+00])
```

[409]:

```python
# Lets calculate some metrics such as R2 score, RSS and RMSE

y_pred_train = lasso.predict(X_train)
y_pred_test = lasso.predict(X_test)

metric3 = []
r2_train_lr = r2_score(y_train, y_pred_train)
print(r2_train_lr)
metric3.append(r2_train_lr)

r2_test_lr = r2_score(y_test, y_pred_test)
print(r2_test_lr)
metric3.append(r2_test_lr)

rss1_lr = np.sum(np.square(y_train - y_pred_train))
print(rss1_lr)
metric3.append(rss1_lr)

rss2_lr = np.sum(np.square(y_test - y_pred_test))
print(rss2_lr)
metric3.append(rss2_lr)

mse_train_lr = mean_squared_error(y_train, y_pred_train)
print(mse_train_lr)
metric3.append(mse_train_lr**0.5)

mse_test_lr = mean_squared_error(y_test, y_pred_test)
print(mse_test_lr)
```

```
metric3.append(mse_test_lr**0.5)
```

```
0.9523603255989929
0.865325505540838
7.645798167030675
9.70579837110768
0.0074885388511563905
0.022159357011661367
```

[410]:
```
# Creating a table which contain all the metrics

lr_table = {'Metric': ['R2 Score (Train)','R2 Score (Test)','RSS (Train)','RSS␣
 ↪(Test)',
                       'MSE (Train)','MSE (Test)'],
        'Linear Regression': metric
        }

lr_metric = pd.DataFrame(lr_table ,columns = ['Metric', 'Linear Regression'] )

rg_metric = pd.Series(metric2, name = 'Ridge Regression')
ls_metric = pd.Series(metric3, name = 'Lasso Regression')

final_metric = pd.concat([lr_metric, rg_metric, ls_metric], axis = 1)

final_metric
```

[410]:

	Metric	Linear Regression	Ridge Regression	Lasso Regression
0	R2 Score (Train)	9.594928e-01	0.957152	0.952360
1	R2 Score (Test)	-1.550167e+22	0.866066	0.865326
2	RSS (Train)	6.501093e+00	6.876814	7.645798
3	RSS (Test)	1.117183e+24	9.652399	9.705798
4	MSE (Train)	7.979585e-02	0.082069	0.086536
5	MSE (Test)	5.050393e+10	0.148450	0.148860

2 Changes in the coefficients after regularization

[411]:
```
betas = pd.DataFrame(index=X.columns)
```

[412]:
```
betas.rows = X.columns
```

[413]:
```
betas['Linear'] = lm.coef_
betas['Ridge'] = ridge.coef_
betas['Lasso'] = lasso.coef_
```

[414]:
```
pd.set_option('display.max_rows', None)
betas.head(68)
```

[414]:

	Linear	Ridge	Lasso
LotFrontage	-3.219921e-04	-0.003416	-0.000000
LotArea	3.169473e-02	0.029844	0.023475
Age	-6.831708e-02	-0.056087	-0.056733
YearRemodAdd	2.383094e-02	0.024057	0.025393
MasVnrArea	1.313672e-03	0.001287	0.001468
BsmtFinSF1	-4.400231e+10	0.020407	0.011586
BsmtFinSF2	-1.556337e+10	0.010621	0.001360
BsmtUnfSF	-4.262936e+10	0.008275	-0.000000
TotalBsmtSF	4.232434e+10	0.033456	0.040081
1stFlrSF	-9.618543e+10	0.041648	0.000000
2ndFlrSF	-1.086110e+11	0.050830	0.000000
LowQualFinSF	-1.209772e+10	-0.000874	-0.006438
GrLivArea	1.307428e+11	0.072785	0.140652
BsmtFullBath	1.254344e-02	0.016639	0.018876
BsmtHalfBath	7.305145e-04	0.001502	0.000000
FullBath	5.434513e-03	0.009960	0.007982
HalfBath	1.474857e-03	0.004048	0.003426
BedroomAbvGr	5.180359e-03	0.006089	0.000000
KitchenAbvGr	-1.891732e-02	-0.019887	-0.020046
TotRmsAbvGrd	4.136086e-03	0.006501	0.006578
Fireplaces	1.011086e-02	0.010906	0.013145
GarageYrBlt	-3.176689e-03	-0.005194	-0.000000
GarageCars	1.646423e-02	0.026245	0.026798
GarageArea	2.102470e-02	0.012749	0.008020
WoodDeckSF	1.034236e-02	0.010605	0.009350
OpenPorchSF	5.126595e-03	0.003738	0.002757
EnclosedPorch	5.578041e-03	0.006231	0.002486
3SsnPorch	4.771709e-03	0.004954	0.003713
ScreenPorch	9.697914e-03	0.010580	0.008869
PoolArea	4.233837e-03	-0.001939	-0.003337
MiscVal	1.765251e-03	0.001434	0.000062
MoSold	7.033348e-06	0.000049	-0.000000
MSZoning_FV	8.511829e-02	0.057248	0.024486
MSZoning_RH	4.283714e-02	0.029440	0.012955
MSZoning_RL	1.642227e-01	0.111530	0.051258
MSZoning_RM	1.236534e-01	0.082750	0.027605
Street_Pave	6.183386e-03	0.006933	0.004841
LotShape_IR2	2.280235e-03	0.002606	0.001290
LotShape_IR3	1.832962e-03	-0.002058	-0.001648
LotShape_Reg	3.325462e-03	0.003709	0.001244
LandContour_HLS	-1.678467e-03	0.000721	0.000000
LandContour_Low	-2.950668e-03	-0.001257	-0.000000
LandContour_Lvl	2.437115e-03	0.005328	0.002587
Utilities_NoSeWa	-1.314640e-03	-0.001952	-0.001343
LotConfig_CulDSac	6.642342e-03	0.007158	0.007781
LotConfig_FR2	-5.674601e-03	-0.006645	-0.002651

```
LotConfig_FR3        -3.466606e-04 -0.001061 -0.000000
LotConfig_Inside     -3.306389e-03 -0.003532 -0.000354
LandSlope_Mod         3.943920e-03  0.003501  0.000000
LandSlope_Sev        -2.119493e-02 -0.017981 -0.011882
Neighborhood_Blueste  2.293587e-03  0.001083  0.000000
Neighborhood_BrDale  -2.456665e-03 -0.004360 -0.004006
Neighborhood_BrkSide  1.764822e-02  0.010475  0.007394
Neighborhood_ClearCr  9.578705e-03  0.008512  0.008660
Neighborhood_CollgCr  4.768372e-03  0.002607  0.000000
Neighborhood_Crawfor  3.156567e-02  0.028670  0.026654
Neighborhood_Edwards -5.568504e-03 -0.011451 -0.012394
Neighborhood_Gilbert  6.357193e-03  0.003679 -0.000000
Neighborhood_IDOTRR   8.115768e-03 -0.000671 -0.004278
Neighborhood_MeadowV -1.544142e-02 -0.017986 -0.016581
Neighborhood_Mitchel -2.345562e-03 -0.004665 -0.004246
Neighborhood_NAmes    8.779526e-03  0.003662  0.000000
Neighborhood_NPkVill -5.569458e-04 -0.001763 -0.000000
Neighborhood_NWAmes  -5.817413e-05 -0.002273 -0.000785
Neighborhood_NoRidge  6.590366e-03  0.008306  0.004513
Neighborhood_NridgHt  1.292229e-02  0.016482  0.014892
Neighborhood_OldTown  1.287079e-02  0.002977 -0.000000
Neighborhood_SWISU    1.227760e-02  0.009898  0.008552
```

2.1 View the Top Variables

[415]:
```
## View the top 10 coefficients of Ridge regression in descending order
betas['Ridge'].sort_values(ascending=False)[:10]
```

[415]:
```
RoofMatl_CompShg    0.259595
RoofMatl_Tar&Grv    0.173271
RoofMatl_WdShngl    0.129206
MSZoning_RL         0.111530
RoofMatl_WdShake    0.103893
MSZoning_RM         0.082750
GrLivArea           0.072785
RoofMatl_Membran    0.058344
MSZoning_FV         0.057248
RoofMatl_Metal      0.056569
Name: Ridge, dtype: float64
```

[416]:
```
## View the top 10 coefficients of Lasso in descending order
betas['Lasso'].sort_values(ascending=False)[:10]
```

[416]:
```
RoofMatl_CompShg    0.249690
RoofMatl_Tar&Grv    0.168176
GrLivArea           0.140652
RoofMatl_WdShngl    0.124946
```

```
RoofMatl_WdShake    0.103698
RoofMatl_Membran    0.054004
RoofMatl_Metal      0.053041
MSZoning_RL         0.051258
RoofMatl_Roll       0.050063
TotalBsmtSF         0.040081
Name: Lasso, dtype: float64
```

[ ]:

Question 1: What is the optimal value of alpha for ridge and lasso regression? What will be the changes in the model if you choose double the value of alpha for both ridge and lasso?

[421]:

```
alpha = 5
ridge1 = Ridge(alpha=alpha)

ridge1.fit(X_train, y_train)
```

[421]:

```
Ridge(alpha=5)
```

[424]:

```
y_pred_train = ridge1.predict(X_train)
y_pred_test = ridge1.predict(X_test)

metric3 = []
r2_train_lr = r2_score(y_train, y_pred_train)
print(r2_train_lr)
metric3.append(r2_train_lr)

r2_test_lr = r2_score(y_test, y_pred_test)
print(r2_test_lr)
metric3.append(r2_test_lr)

rss1_lr = np.sum(np.square(y_train - y_pred_train))
print(rss1_lr)
metric3.append(rss1_lr)

rss2_lr = np.sum(np.square(y_test - y_pred_test))
print(rss2_lr)
metric3.append(rss2_lr)

mse_train_lr = mean_squared_error(y_train, y_pred_train)
print(mse_train_lr)
metric3.append(mse_train_lr**0.5)

mse_test_lr = mean_squared_error(y_test, y_pred_test)
print(mse_test_lr)
metric3.append(mse_test_lr**0.5)
```

```
0.9571517367529878
0.8660664550646793
6.876813847147698
9.652399197712207
0.006735371054992848
0.02203744109066714
```

[425]:
```
#Now double the alpha for Lasso
alpha = 0.002

lasso = Lasso(alpha=alpha)

lasso.fit(X_train, y_train)
```

[425]:
```
Lasso(alpha=0.002)
```

[426]:
```
y_pred_train = lasso.predict(X_train)
y_pred_test = lasso.predict(X_test)

metric3 = []
r2_train_lr = r2_score(y_train, y_pred_train)
print(r2_train_lr)
metric3.append(r2_train_lr)

r2_test_lr = r2_score(y_test, y_pred_test)
print(r2_test_lr)
metric3.append(r2_test_lr)

rss1_lr = np.sum(np.square(y_train - y_pred_train))
print(rss1_lr)
metric3.append(rss1_lr)

rss2_lr = np.sum(np.square(y_test - y_pred_test))
print(rss2_lr)
metric3.append(rss2_lr)

mse_train_lr = mean_squared_error(y_train, y_pred_train)
print(mse_train_lr)
metric3.append(mse_train_lr**0.5)

mse_test_lr = mean_squared_error(y_test, y_pred_test)
print(mse_test_lr)
metric3.append(mse_test_lr**0.5)
```

```
0.9412756545141392
0.8645896828911631
9.424801884590085
9.758828058008072
```

```
0.0092309518947993
0.022280429356182813
```

[427]:
```
# Creating a table which contain all the metrics

lr_table = {'Metric': ['R2 Score (Train)','R2 Score (Test)','RSS (Train)','RSS␣
↪(Test)',
                       'MSE (Train)','MSE (Test)'],
        'Linear Regression': metric
        }

lr_metric = pd.DataFrame(lr_table ,columns = ['Metric', 'Linear Regression'] )

rg_metric = pd.Series(metric2, name = 'Ridge Regression')
ls_metric = pd.Series(metric3, name = 'Lasso Regression')

final_metric = pd.concat([lr_metric, rg_metric, ls_metric], axis = 1)

final_metric
```

[427]:

	Metric	Linear Regression	Ridge Regression	Lasso Regression
0	R2 Score (Train)	9.594928e-01	0.957152	0.941276
1	R2 Score (Test)	-1.550167e+22	0.866066	0.864590
2	RSS (Train)	6.501093e+00	6.876814	9.424802
3	RSS (Test)	1.117183e+24	9.652399	9.758828
4	MSE (Train)	7.979585e-02	0.082069	0.096078
5	MSE (Test)	5.050393e+10	0.148450	0.149266

Question 3: After building the model, you realised that the five most important predictor variables in the lasso model are not available in the incoming data. You will now have to create another model excluding the five most important predictor variables. Which are the five most important predictor variables now?

[428]:
```
#Here, we will drop the top 5 features in Lasso model and build the model again.
top_var = ['RoofMatl_CompShg', 'RoofMatl_Tar&Grv', 'GrLivArea',␣
↪'RoofMatl_WdShngl', 'RoofMatl_WdShake']
## drop them from train and test data
X_train_dropped = X_train.drop(top_var, axis=1)
X_test_dropped = X_test.drop(top_var, axis=1)
```

[429]:
```
lasso = Lasso()

# cross validation
model_cv = GridSearchCV(estimator = lasso,
                        param_grid = params,
                        scoring= 'neg_mean_absolute_error',
                        cv = folds,
                        return_train_score=True,
```

```
                        verbose = 1)

model_cv.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 28 candidates, totalling 140 fits
```

[429]:
```
GridSearchCV(cv=5, estimator=Lasso(),
             param_grid={'alpha': [0.0001, 0.001, 0.01, 0.05, 0.1, 0.2, 0.3,
                                   0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 2.0, 3.0,
                                   4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 20, 50,
                                   100, 500, 1000]},
             return_train_score=True, scoring='neg_mean_absolute_error',
             verbose=1)
```

[430]:
```
# Printing the best hyperparameter alpha
print(model_cv.best_params_)
```

```
{'alpha': 0.001}
```

[431]:
```
#Fitting Ridge model for alpha = 0.001 and printing coefficients which have␣
↪been penalised

alpha = 0.001

lasso = Lasso(alpha=alpha)

lasso.fit(X_train, y_train)
```

[431]:
```
Lasso(alpha=0.001)
```

[433]:
```
# Lets calculate some metrics such as R2 score, RSS and RMSE

y_pred_train = lasso.predict(X_train)
y_pred_test = lasso.predict(X_test)

metric3 = []
r2_train_lr = r2_score(y_train, y_pred_train)
print(r2_train_lr)
metric3.append(r2_train_lr)

r2_test_lr = r2_score(y_test, y_pred_test)
print(r2_test_lr)
metric3.append(r2_test_lr)

rss1_lr = np.sum(np.square(y_train - y_pred_train))
print(rss1_lr)
metric3.append(rss1_lr)
```

```
rss2_lr = np.sum(np.square(y_test - y_pred_test))
print(rss2_lr)
metric3.append(rss2_lr)

mse_train_lr = mean_squared_error(y_train, y_pred_train)
print(mse_train_lr)
metric3.append(mse_train_lr**0.5)

mse_test_lr = mean_squared_error(y_test, y_pred_test)
print(mse_test_lr)
metric3.append(mse_test_lr**0.5)
```

```
0.9523603255989929
0.865325505540838
7.645798167030675
9.70579837110768
0.0074885388511563905
0.022159357011661367
```

[434]:
```
ls_metric = pd.Series(metric3, name = 'Lasso Regression')
ls_metric
```

[434]:
```
0    0.952360
1    0.865326
2    7.645798
3    9.705798
4    0.086536
5    0.148860
Name: Lasso Regression, dtype: float64
```

[435]:
```
#Changes in the coefficients after regularization

betas = pd.DataFrame(index=X.columns)
betas.rows = X.columns
betas['Lasso'] = lasso.coef_
pd.set_option('display.max_rows', None)
betas.head(68)
```

[435]:

	Lasso
LotFrontage	-0.000000
LotArea	0.023475
Age	-0.056733
YearRemodAdd	0.025393
MasVnrArea	0.001468
BsmtFinSF1	0.011586
BsmtFinSF2	0.001360
BsmtUnfSF	-0.000000
TotalBsmtSF	0.040081

1stFlrSF	0.000000
2ndFlrSF	0.000000
LowQualFinSF	-0.006438
GrLivArea	0.140652
BsmtFullBath	0.018876
BsmtHalfBath	0.000000
FullBath	0.007982
HalfBath	0.003426
BedroomAbvGr	0.000000
KitchenAbvGr	-0.020046
TotRmsAbvGrd	0.006578
Fireplaces	0.013145
GarageYrBlt	-0.000000
GarageCars	0.026798
GarageArea	0.008020
WoodDeckSF	0.009350
OpenPorchSF	0.002757
EnclosedPorch	0.002486
3SsnPorch	0.003713
ScreenPorch	0.008869
PoolArea	-0.003337
MiscVal	0.000062
MoSold	-0.000000
MSZoning_FV	0.024486
MSZoning_RH	0.012955
MSZoning_RL	0.051258
MSZoning_RM	0.027605
Street_Pave	0.004841
LotShape_IR2	0.001290
LotShape_IR3	-0.001648
LotShape_Reg	0.001244
LandContour_HLS	0.000000
LandContour_Low	-0.000000
LandContour_Lvl	0.002587
Utilities_NoSeWa	-0.001343
LotConfig_CulDSac	0.007781
LotConfig_FR2	-0.002651
LotConfig_FR3	-0.000000
LotConfig_Inside	-0.000354
LandSlope_Mod	0.000000
LandSlope_Sev	-0.011882
Neighborhood_Blueste	0.000000
Neighborhood_BrDale	-0.004006
Neighborhood_BrkSide	0.007394
Neighborhood_ClearCr	0.008660
Neighborhood_CollgCr	0.000000
Neighborhood_Crawfor	0.026654

```
Neighborhood_Edwards -0.012394
Neighborhood_Gilbert -0.000000
Neighborhood_IDOTRR  -0.004278
Neighborhood_MeadowV -0.016581
Neighborhood_Mitchel -0.004246
Neighborhood_NAmes    0.000000
Neighborhood_NPkVill -0.000000
Neighborhood_NWAmes  -0.000785
Neighborhood_NoRidge  0.004513
Neighborhood_NridgHt  0.014892
Neighborhood_OldTown -0.000000
Neighborhood_SWISU    0.008552
```

[436]:
```
## View the top 5 coefficients of Lasso in descending order
betas['Lasso'].sort_values(ascending=False)[:5]
```

[436]:
```
RoofMatl_CompShg    0.249690
RoofMatl_Tar&Grv    0.168176
GrLivArea           0.140652
RoofMatl_WdShngl    0.124946
RoofMatl_WdShake    0.103698
Name: Lasso, dtype: float64
```

[ ]: