house-price-prediction

June 6, 2023

1 Importing Data and Libraries

[46]:
```
# Importing pre-installed Libraries

import seaborn as sns  # For advanced data visualization and plotting,␣
↪enhancing the visual representation of data.
import numpy as np  # For numerical computations, efficient handling of arrays,␣
↪and mathematical operations.
import pandas as pd  # For data manipulation, analysis, and working with␣
↪structured data in tabular form.
import matplotlib.pyplot as plt  # For creating plots, charts, and␣
↪visualizations of data.
import seaborn as sns  # Additional functionality for data visualization,␣
↪complementing matplotlib.
import os  # For interacting with the operating system, such as file and␣
↪directory manipulation.

from scipy.stats import (
    boxcox,
)  # For performing the Box-Cox transformation, a data transformation technique␣
↪that helps normalize skewed data.

import statsmodels.api as sm  # For advanced statistical models and analysis,␣
↪providing a wide range of statistical tools.
from statsmodels.stats.outliers_influence import (
    variance_inflation_factor,
)  # For detecting multicollinearity, a measure of correlation between␣
↪predictor variables.

from sklearn.preprocessing import (
    StandardScaler,
)  # For standardizing feature scaling, ensuring each feature has a mean of 0␣
↪and standard deviation of 1.
from sklearn.model_selection import (
    train_test_split,
)  # For splitting data into training and testing sets, evaluating model␣
↪performance.
```

```
from sklearn.linear_model import (
    LinearRegression,
)  # For implementing linear regression models, fitting a linear equation to␣
 ↪the observed data.
from sklearn.ensemble import (
    RandomForestRegressor,
)  # For implementing random forest regression models, using an ensemble of␣
 ↪decision trees.
from sklearn.metrics import (
    r2_score,
    explained_variance_score,
)  # For evaluating regression model performance, measuring the amount of␣
 ↪variance explained by the model.
from sklearn.impute import (
    KNNImputer,
)  # For imputing missing values using the K-Nearest Neighbors algorithm.
from sklearn import (
    preprocessing,
)  # For preprocessing and data transformation, including scaling, encoding,␣
 ↪and normalization.
from sklearn import (
    metrics,
)  # For evaluating model performance and metrics, such as accuracy, precision,␣
 ↪and recall.

import missingno as msno  # For visualizing missing data patterns in the␣
 ↪dataset, identifying missing data and their distribution.
```

[47]:
```
# Importing self-build libraries and functions
from manipulation import (
    feature_select,
    box_cox_transformer,
    outlayer_remover,
    calculate_z_scores,
)
from eda import data_overview, visualization, prediction_scatter
from manipulation import preprocesing
from machine_learning import (
    feature_target_splitter,
    machine_learning,
    multireg_statmodels,
)
```

[48]:
```
# Getting current directory path
current_dir = os.getcwd()
```

[49]:
```
# Importing Data
df = pd.read_csv(current_dir + "/dataset.csv")
dscr = open(current_dir + "/data_description.txt")
dscr = dscr.read()
```

2 Data Overview

[50]:
```
overview = data_overview(df, dscr)
```

2.0.1 Looking at first and last rows of the dataframe

[51]:
```
overview.head_tail()
```

```
Data Head:

   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]
```

```
Data Tail:

        Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape  \
1455  1456          60       RL         62.0     7917   Pave   NaN      Reg
1456  1457          20       RL         85.0    13175   Pave   NaN      Reg
1457  1458          70       RL         66.0     9042   Pave   NaN      Reg
1458  1459          20       RL         68.0     9717   Pave   NaN      Reg
1459  1460          20       RL         75.0     9937   Pave   NaN      Reg

     LandContour Utilities  … PoolArea PoolQC  Fence MiscFeature MiscVal  \
1455         Lvl    AllPub  …        0    NaN    NaN         NaN       0
1456         Lvl    AllPub  …        0    NaN  MnPrv         NaN       0
1457         Lvl    AllPub  …        0    NaN  GdPrv        Shed    2500
1458         Lvl    AllPub  …        0    NaN    NaN         NaN       0
1459         Lvl    AllPub  …        0    NaN    NaN         NaN       0

     MoSold YrSold  SaleType  SaleCondition  SalePrice
1455      8   2007        WD         Normal     175000
1456      2   2010        WD         Normal     210000
1457      5   2010        WD         Normal     266500
1458      4   2010        WD         Normal     142125
1459      6   2008        WD         Normal     147500

[5 rows x 81 columns]
```

2.0.2 Looking at dataset documentation and feature description

[52]:
```
overview.features_description()
```

```
The Description for Features:


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)
```

2.0.3 Looking into the data information

[53]:
```
overview.data_information()
```

```
<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     588 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
None
```

2.0.4 Descriptive statistics of the columns with numeric objecs

[54]:
```
overview.descriptive_statistics()
```

	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

	BsmtFinSF2	BsmtUnfSF	TotalBsmtSF	1stFlrSF	2ndFlrSF
count	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000
mean	46.549315	567.240411	1057.429452	1162.626712	346.992466
std	161.319273	441.866955	438.705324	386.587738	436.528436
min	0.000000	0.000000	0.000000	334.000000	0.000000
25%	0.000000	223.000000	795.750000	882.000000	0.000000
50%	0.000000	477.500000	991.500000	1087.000000	0.000000
75%	0.000000	808.000000	1298.250000	1391.250000	728.000000
max	1474.000000	2336.000000	6110.000000	4692.000000	2065.000000

	LowQualFinSF	GrLivArea	BsmtFullBath	BsmtHalfBath	FullBath
count	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000
mean	5.844521	1515.463699	0.425342	0.057534	1.565068
std	48.623081	525.480383	0.518911	0.238753	0.550916
min	0.000000	334.000000	0.000000	0.000000	0.000000
25%	0.000000	1129.500000	0.000000	0.000000	1.000000
50%	0.000000	1464.000000	0.000000	0.000000	2.000000
75%	0.000000	1776.750000	1.000000	0.000000	2.000000
max	572.000000	5642.000000	3.000000	2.000000	3.000000

	HalfBath	BedroomAbvGr	KitchenAbvGr	TotRmsAbvGrd	Fireplaces
count	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000
mean	0.382877	2.866438	1.046575	6.517808	0.613014
std	0.502885	0.815778	0.220338	1.625393	0.644666

min	0.000000	0.000000	0.000000	2.000000	0.000000
25%	0.000000	2.000000	1.000000	5.000000	0.000000
50%	0.000000	3.000000	1.000000	6.000000	1.000000
75%	1.000000	3.000000	1.000000	7.000000	1.000000
max	2.000000	8.000000	3.000000	14.000000	3.000000

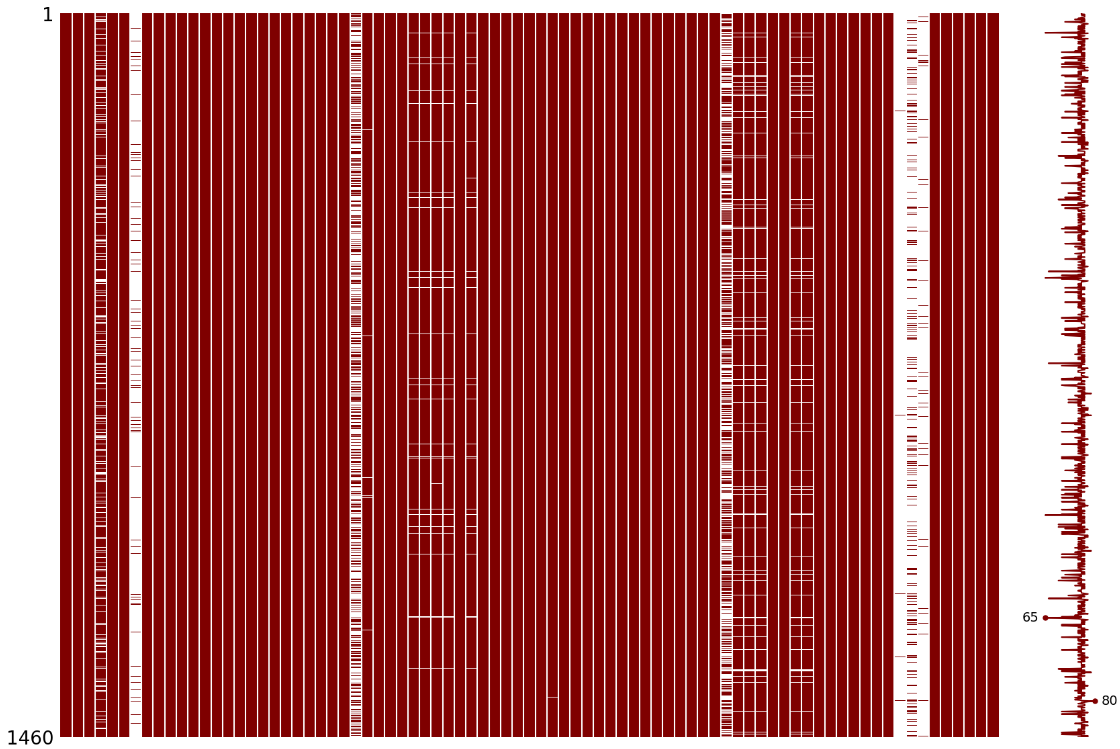
	GarageYrBlt	GarageCars	GarageArea	WoodDeckSF	OpenPorchSF
count	1379.000000	1460.000000	1460.000000	1460.000000	1460.000000
mean	1978.506164	1.767123	472.980137	94.244521	46.660274
std	24.689725	0.747315	213.804841	125.338794	66.256028
min	1900.000000	0.000000	0.000000	0.000000	0.000000
25%	1961.000000	1.000000	334.500000	0.000000	0.000000
50%	1980.000000	2.000000	480.000000	0.000000	25.000000
75%	2002.000000	2.000000	576.000000	168.000000	68.000000
max	2010.000000	4.000000	1418.000000	857.000000	547.000000

	EnclosedPorch	3SsnPorch	ScreenPorch	PoolArea	MiscVal
count	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000
mean	21.954110	3.409589	15.060959	2.758904	43.489041
std	61.119149	29.317331	55.757415	40.177307	496.123024
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	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000
max	552.000000	508.000000	480.000000	738.000000	15500.000000

	MoSold	YrSold	SalePrice
count	1460.000000	1460.000000	1460.000000
mean	6.321918	2007.815753	180921.195890
std	2.703626	1.328095	79442.502883
min	1.000000	2006.000000	34900.000000
25%	5.000000	2007.000000	129975.000000
50%	6.000000	2008.000000	163000.000000
75%	8.000000	2009.000000	214000.000000
max	12.000000	2010.000000	755000.000000

2.0.5 Visualize Missing Values

[55]:
```
overview.visualize_missing()
```



3 Data Manipulation

3.1 Check for duplicates

[56]:
```
sum(df.duplicated())
```

[56]: 0

If this line of code returns any number rather than 0 , it means we have duplicated rows in our dataset. But as can be seen, it returns 0 which means we have no duplicated row here.

- Usually after this step, we search for structural errors such as string inconsistencies. However, regarding the documentation, we don't have these type of issues for this dataset.

3.2 Checking for null values

[57]:
```
overview.features_with_null_values()
```

```
The features containing null values :

Feature     number of null

LotFrontage      259
```

```
Alley           1369
MasVnrType       872
MasVnrArea         8
BsmtQual          37
BsmtCond          37
BsmtExposure      38
BsmtFinType1      37
BsmtFinType2      38
Electrical         1
FireplaceQu      690
GarageType        81
GarageYrBlt       81
GarageFinish      81
GarageQual        81
GarageCond        81
PoolQC          1453
Fence           1179
MiscFeature     1406
dtype: int64
```

3.3 Handling missing values

As seen in the documentation, two kinds of null values exist in our dataset. One kind is meaningful, and we have to replace them with real values. On the other hand, real null values are truly missed during the data-collecting process. In this section, we replace the meaningful null values with the true values for them.

features mentioned in documentation

- **Alley** Considering data description, NA means "no alley access"

[58]:
```
df["Alley"] = df["Alley"].fillna("None")
```

- **PoolQC** According to the description, NA stands for "No Pool." That makes sense, considering the enormous missing value ratio (+99%) and the fact that the majority of houses don't generally have pools.

[59]:
```
df["PoolQC"] = df["PoolQC"].fillna("None")
```

- **FireplaceQu** data description says NA means "no fireplace"

[60]:
```
df["FireplaceQu"] = df["FireplaceQu"].fillna("None")
```

- **Fence** Considering data description, NA means “no fence”

[61]:
```
df["Fence"] = df["Fence"].fillna("None")
```

- **MiscFeature** Considering data description, NA means “no misc feature”

[62]:
```
df["MiscFeature"] = df["MiscFeature"].fillna("None")
```

- **BsmtQual, BsmtCond, BsmtExposure, BsmtFinType1 and BsmtFinType2** For all basement-related features, NaN means that there is no basement.

[63]:
```
for cname in ("BsmtQual", "BsmtCond", "BsmtExposure", "BsmtFinType1",␣
 ↪"BsmtFinType2"):
    df[cname] = df[cname].fillna("None")
```

- **GarageType, GarageFinish, GarageQual and GarageCond** Regrarding documentation, NA for garange features means ‘feature is not available’. so we replace missing data with ‘None’

[64]:
```
for cname in ("GarageType", "GarageFinish", "GarageQual", "GarageCond"):
    df[cname] = df[cname].fillna("None")
```

- **GarageYrBlt, GarageArea and GarageCars** Although it is not mentioned in the documentation, we replce non with 0 (Since No garage = no cars in such garage)

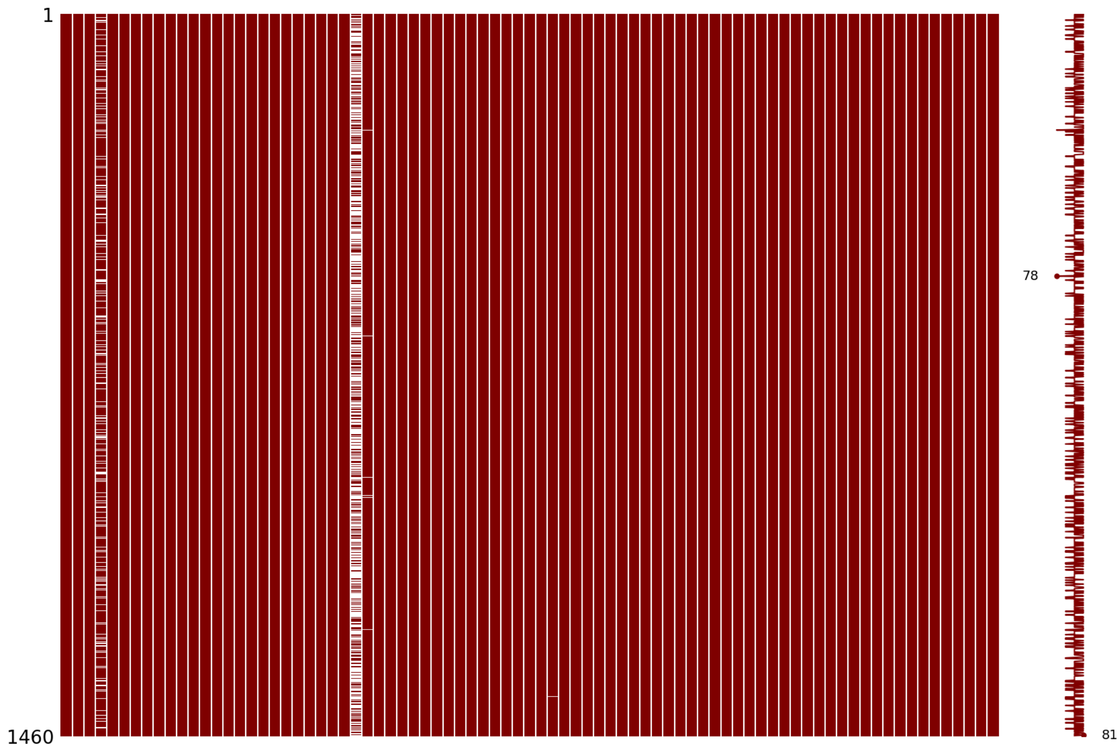
[65]:
```
for cname in ("GarageYrBlt", "GarageArea", "GarageCars"):
    df[cname] = df[cname].fillna(0)
```

[66]:
```
# Check for any null value
overview.features_with_null_values()
```

```
The features containing null values :

Feature     number of null

LotFrontage    259
MasVnrType     872
MasVnrArea       8
```

```
Electrical       1
dtype: int64
```

[67]:
```python
# Visualize missing values
overview.visualize_missing()
```



features not mentioned in the documentation As can be seen, after feature engineering, almost all of the features contain no null values anymore, except four features. MasVnrType and MasVnrArea have only 8 null values. The electrical feature has only one missing value. But LotFrontage has the most missing values, which is 259. For features with a few missing values, these null values have almost no effect on the feature selection process based on correlation. But for a feature like 'LotFrontage' with almost %20 missing values, the missing values might affect the feature selection process. So basically, it is better to impute these values by making some assumptions before feature selection.

- **MasVnrArea and MasVnrType** It is not mentioned in the description file that NA means none, But it most likely means no masonry veneer for these houses. So with this assumption we can fill 0 for the area and None for the type.

[68]:
```python
df["MasVnrType"] = df["MasVnrType"].fillna("None")
df["MasVnrArea"] = df["MasVnrArea"].fillna(0)
```

- **Electrical** This feature has only one null value. In this case we can just drop this single item. However, our dataset contains limited rows. So, it is more efficient to fill this one missing value with mode.

[69]:
```
df["Electrical"] = df["Electrical"].fillna(df["Electrical"].mode()[0])
```

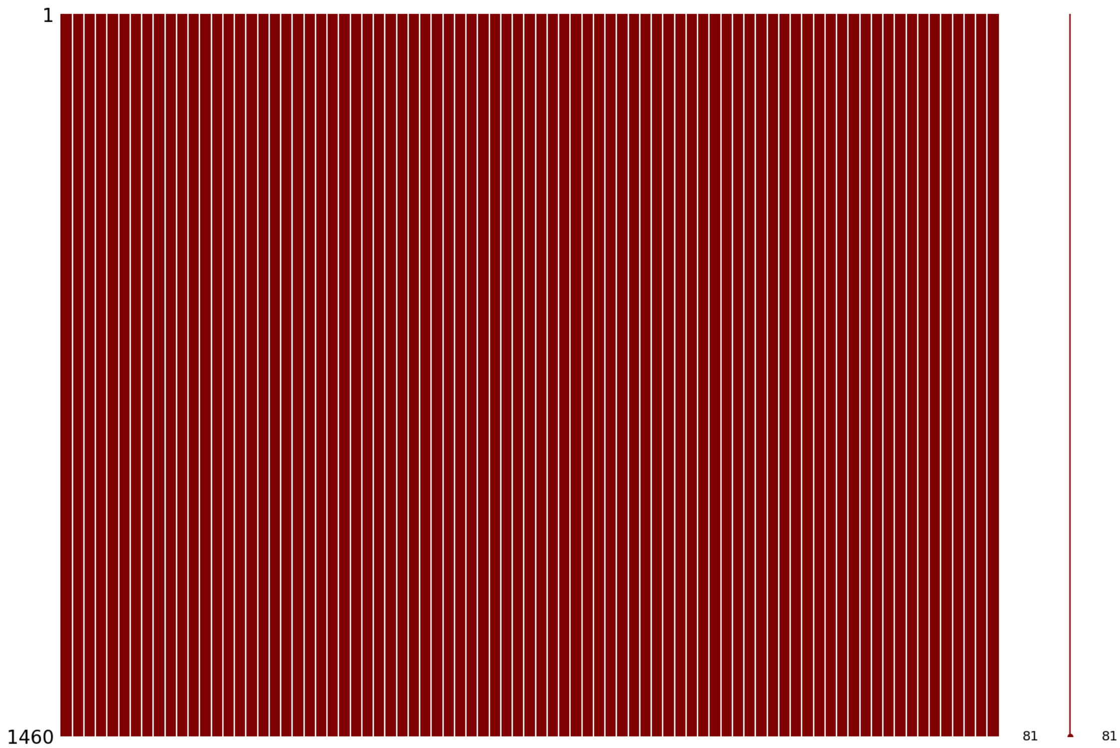
- **LotFrontage** There is no clue in documntation to impute the LotFrontage null values. But, since the area of each street connected to the house property most likely have a similar area to other houses in its neighborhood , we can fill in missing values by average of its four neighbour values. To do so, we use KNNImputer which is a scikit-learn module.

[70]:
```
# Imputing null values with KNNImputer
imputer = KNNImputer(n_neighbors=4)
df["LotFrontage"] = imputer.fit_transform(
    imputer.fit_transform(np.array(df["LotFrontage"]).reshape(-1, 1))
)
```

[71]:
```
# Check for any null value
overview.features_with_null_values()
```

```
No feature contains null value
```

[72]:
```
# Visualize missing values
overview.visualize_missing()
```




```
[73]: all_data = df.copy()
```

3.4 String to Numeric transformation

3.4.1 Converting Data to Categorical

First we need to check if there is any string inconsistencies in the columns with string data type. To do so, we check for unique values of columns.

```
[74]: # Check out if there is any string inconsistencies
for cname in df.columns:
    if df[cname].dtype == "object":
        print(f"{cname} : ", df[cname].unique(), "\n")
```

```
MSZoning :  ['RL' 'RM' 'C (all)' 'FV' 'RH']

Street :  ['Pave' 'Grvl']

Alley :  ['None' 'Grvl' 'Pave']

LotShape :  ['Reg' 'IR1' 'IR2' 'IR3']

LandContour :  ['Lvl' 'Bnk' 'Low' 'HLS']
```

```
Utilities :  ['AllPub' 'NoSeWa']

LotConfig :  ['Inside' 'FR2' 'Corner' 'CulDSac' 'FR3']

LandSlope :  ['Gtl' 'Mod' 'Sev']

Neighborhood :  ['CollgCr' 'Veenker' 'Crawfor' 'NoRidge' 'Mitchel' 'Somerst'
'NWAmes'
 'OldTown' 'BrkSide' 'Sawyer' 'NridgHt' 'NAmes' 'SawyerW' 'IDOTRR'
 'MeadowV' 'Edwards' 'Timber' 'Gilbert' 'StoneBr' 'ClearCr' 'NPkVill'
 'Blmngtn' 'BrDale' 'SWISU' 'Blueste']

Condition1 :  ['Norm' 'Feedr' 'PosN' 'Artery' 'RRAe' 'RRNn' 'RRAn' 'PosA'
'RRNe']

Condition2 :  ['Norm' 'Artery' 'RRNn' 'Feedr' 'PosN' 'PosA' 'RRAn' 'RRAe']

BldgType :  ['1Fam' '2fmCon' 'Duplex' 'TwnhsE' 'Twnhs']

HouseStyle :  ['2Story' '1Story' '1.5Fin' '1.5Unf' 'SFoyer' 'SLvl' '2.5Unf'
'2.5Fin']

RoofStyle :  ['Gable' 'Hip' 'Gambrel' 'Mansard' 'Flat' 'Shed']

RoofMatl :  ['CompShg' 'WdShngl' 'Metal' 'WdShake' 'Membran' 'Tar&Grv' 'Roll'
 'ClyTile']

Exterior1st :  ['VinylSd' 'MetalSd' 'Wd Sdng' 'HdBoard' 'BrkFace' 'WdShing'
'CemntBd'
 'Plywood' 'AsbShng' 'Stucco' 'BrkComm' 'AsphShn' 'Stone' 'ImStucc'
 'CBlock']

Exterior2nd :  ['VinylSd' 'MetalSd' 'Wd Shng' 'HdBoard' 'Plywood' 'Wd Sdng'
'CmentBd'
 'BrkFace' 'Stucco' 'AsbShng' 'Brk Cmn' 'ImStucc' 'AsphShn' 'Stone'
 'Other' 'CBlock']

MasVnrType :  ['BrkFace' 'None' 'Stone' 'BrkCmn']

ExterQual :  ['Gd' 'TA' 'Ex' 'Fa']

ExterCond :  ['TA' 'Gd' 'Fa' 'Po' 'Ex']

Foundation :  ['PConc' 'CBlock' 'BrkTil' 'Wood' 'Slab' 'Stone']

BsmtQual :  ['Gd' 'TA' 'Ex' 'None' 'Fa']

BsmtCond :  ['TA' 'Gd' 'None' 'Fa' 'Po']
```

```
BsmtExposure :  ['No' 'Gd' 'Mn' 'Av' 'None']

BsmtFinType1 :  ['GLQ' 'ALQ' 'Unf' 'Rec' 'BLQ' 'None' 'LwQ']

BsmtFinType2 :  ['Unf' 'BLQ' 'None' 'ALQ' 'Rec' 'LwQ' 'GLQ']

Heating :  ['GasA' 'GasW' 'Grav' 'Wall' 'OthW' 'Floor']

HeatingQC :  ['Ex' 'Gd' 'TA' 'Fa' 'Po']

CentralAir :  ['Y' 'N']

Electrical :  ['SBrkr' 'FuseF' 'FuseA' 'FuseP' 'Mix']

KitchenQual :  ['Gd' 'TA' 'Ex' 'Fa']

Functional :  ['Typ' 'Min1' 'Maj1' 'Min2' 'Mod' 'Maj2' 'Sev']

FireplaceQu :  ['None' 'TA' 'Gd' 'Fa' 'Ex' 'Po']

GarageType :  ['Attchd' 'Detchd' 'BuiltIn' 'CarPort' 'None' 'Basment' '2Types']

GarageFinish :  ['RFn' 'Unf' 'Fin' 'None']

GarageQual :  ['TA' 'Fa' 'Gd' 'None' 'Ex' 'Po']

GarageCond :  ['TA' 'Fa' 'None' 'Gd' 'Po' 'Ex']

PavedDrive :  ['Y' 'N' 'P']

PoolQC :  ['None' 'Ex' 'Fa' 'Gd']

Fence :  ['None' 'MnPrv' 'GdWo' 'GdPrv' 'MnWw']

MiscFeature :  ['None' 'Shed' 'Gar2' 'Othr' 'TenC']

SaleType :  ['WD' 'New' 'COD' 'ConLD' 'ConLI' 'CWD' 'ConLw' 'Con' 'Oth']

SaleCondition :  ['Normal' 'Abnorml' 'Partial' 'AdjLand' 'Alloca' 'Family']
```

As can be seen, there is no string inconsistency in the columns. Also, all the values are the same as the values in the documentation. The problem here is that values are strings. To do Machine Learning techniques on them, we need to convert them to numeric data. To do so, initially, we need to convert them to categoricals data which are pandas data types corresponding to categorical variables in statistics. Then we need to convert them to numerics.

To accomplish the first step, we define a function which takes a data frame as the input and returns another data frame with the same shape and values. The only difference is that in the returned data frame string values are converted to categorical data.

[75]:
```python
# Convert string data to categorical data
preprocessing = preprocesing()
df = preprocessing.string_to_categorical(df)
```

[76]:
```python
df.head()
```

[76]:
```
   Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape
0   1          60       RL         65.0     8450   Pave  None      Reg  \
1   2          20       RL         80.0     9600   Pave  None      Reg
2   3          60       RL         68.0    11250   Pave  None      IR1
3   4          70       RL         60.0     9550   Pave  None      IR1
4   5          60       RL         84.0    14260   Pave  None      IR1

  LandContour Utilities  … PoolArea PoolQC Fence MiscFeature MiscVal MoSold
0         Lvl    AllPub  …        0   None  None        None       0      2  \
1         Lvl    AllPub  …        0   None  None        None       0      5
2         Lvl    AllPub  …        0   None  None        None       0      9
3         Lvl    AllPub  …        0   None  None        None       0      2
4         Lvl    AllPub  …        0   None  None        None       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]
```

3.4.2 Converting Categorical Data to Numeric

Now that non-numeric data is categorical, we need to convert them to numerics. There are different methods to accomplish it. We prefer Scikit Learn Label Encoder tool that can satisfy our needs.

It iterates through the columns. For each column, it starts to label instances of that column from zero to a number of different instances in that column minus one. The problem with this approach is that the label encoder labels instances with numbers regardless of the hierarchy of instances. For example, if we have three different instances, such as 'negative', 'neutral', and 'positive, it doesn't necessarily label them with 0, 1, and 2. Instead, it labels them randomly, considering the order in these instances is sorted in the column. This issue might affect our accuracy in the machine learning process; however, it could not impact the correlation much. So, we must handle it after feature selection.

Our function (categorical_to_numeric) takes the data frame containing categorical data and converts it to a new data frame consisting of all numerics. Then it returns the new data frame, also a dictionary to map the numerics to the original categories. It could be helpful in data visualisation because, in that case, we need to represent the data with original labels to convey insight to the consumer rather than meaningless numerics.

[77]:
```
# Convert Categorical Data to Numeric
df, col_dic = preprocessing.categorical_to_numeric(df)
```

[78]:
```
df.head()
```

[78]:
```
   Id  MSSubClass  MSZoning  LotFrontage  LotArea  Street  Alley  LotShape  \
0   1          60         3         65.0     8450       1      1         3
1   2          20         3         80.0     9600       1      1         3
2   3          60         3         68.0    11250       1      1         0
3   4          70         3         60.0     9550       1      1         0
4   5          60         3         84.0    14260       1      1         0

   LandContour  Utilities  …  PoolArea  PoolQC  Fence  MiscFeature  MiscVal  \
0            3          0  …         0       3      4            1        0
1            3          0  …         0       3      4            1        0
2            3          0  …         0       3      4            1        0
3            3          0  …         0       3      4            1        0
4            3          0  …         0       3      4            1        0

   MoSold  YrSold  SaleType  SaleCondition  SalePrice
0       2    2008         8              4     208500
1       5    2007         8              4     181500
2       9    2008         8              4     223500
3       2    2006         8              0     140000
4      12    2008         8              4     250000

[5 rows x 81 columns]
```

[79]:
```
# Preview the 5 first item of the mapping dictionary
dict(list(col_dic.items())[0:5])
```

[79]:
```
{'MSZoning': {0: 'C (all)', 1: 'FV', 2: 'RH', 3: 'RL', 4: 'RM'},
 'Street': {0: 'Grvl', 1: 'Pave'},
 'Alley': {0: 'Grvl', 1: 'None', 2: 'Pave'},
 'LotShape': {0: 'IR1', 1: 'IR2', 2: 'IR3', 3: 'Reg'},
 'LandContour': {0: 'Bnk', 1: 'HLS', 2: 'Low', 3: 'Lvl'}}
```

[80]:
```
rawdf = df.copy()
```

4 Feature Selection

Now that all our data is numeric, it is time to select the features that impact our target 'SalePrice' most. There are two main methods for feature selection. Supervised and Unsupervised. As our data set is supervised and we have access to our targets, the correlation method, a supervised approach, is suitable for our dataset.

[81]:
```python
# Select features by highest correlation
corr_dic = feature_select(df, "SalePrice")
```

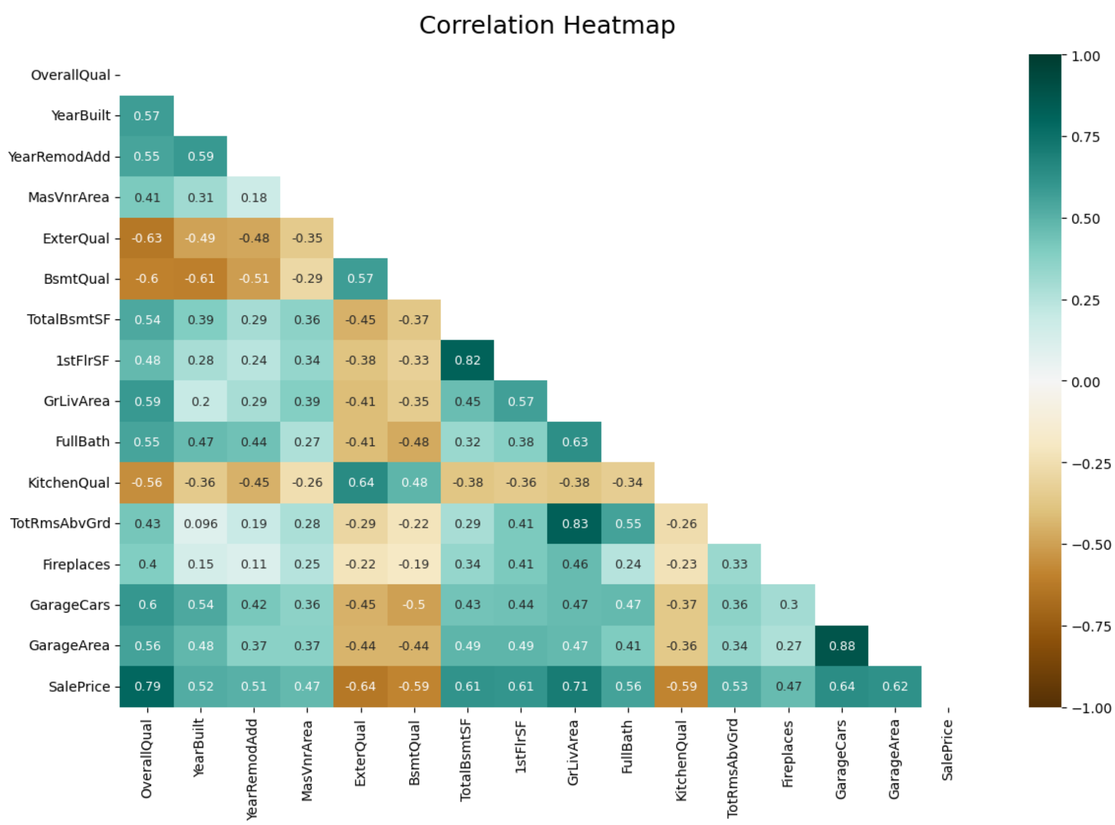
[82]:
```python
corr_dic
```

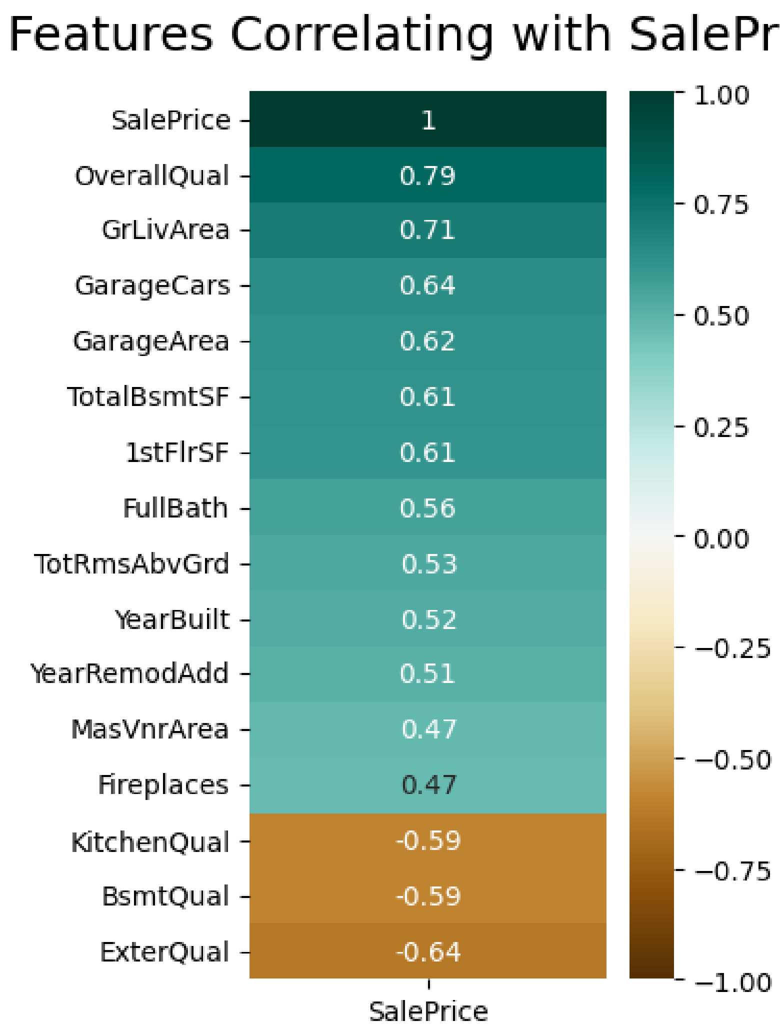
[82]:
```
{'OverallQual': 0.7909816005838053,
 'YearBuilt': 0.5228973328794968,
 'YearRemodAdd': 0.5071009671113862,
 'MasVnrArea': 0.4726144990045739,
 'ExterQual': -0.6368836943991126,
 'BsmtQual': -0.5937339191038187,
 'TotalBsmtSF': 0.6135805515591956,
 '1stFlrSF': 0.6058521846919145,
 'GrLivArea': 0.7086244776126521,
 'FullBath': 0.5606637627484456,
 'KitchenQual': -0.5891887782994207,
 'TotRmsAbvGrd': 0.5337231555820281,
 'Fireplaces': 0.46692883675152796,
 'GarageCars': 0.6404091972583522,
 'GarageArea': 0.6234314389183617,
 'SalePrice': 0.9999999999999998}
```

[83]:
```python
# Create new dataframe with selected features and target
df = df[corr_dic.keys()]

# Print the shape of new dataframe
df.shape
```

[83]:
```
(1460, 16)
```

[84]:
```python
grapher = visualization(df)
grapher.correlation_plotter()
```


Correlation Heatmap
OverallQual
YearBuilt 0.57
YearRemodAdd 0.55 0.59
MasVnrArea 0.41 0.31 0.18
ExterQual -0.63 -0.49 -0.48 -0.35
BsmtQual -0.6 -0.61 -0.51 -0.29 0.57
TotalBsmtSF 0.54 0.39 0.29 0.36 -0.45 -0.37
1stFlrSF 0.48 0.28 0.24 0.34 -0.38 -0.33 0.82
GrLivArea 0.59 0.2 0.29 0.39 -0.41 -0.35 0.45 0.57
FullBath 0.55 0.47 0.44 0.27 -0.41 -0.48 0.32 0.38 0.63
KitchenQual -0.56 -0.36 -0.45 -0.26 0.64 0.48 -0.38 -0.36 -0.38 -0.34
TotRmsAbvGrd 0.43 0.096 0.19 0.28 -0.29 -0.22 0.29 0.41 0.83 0.55 -0.26
Fireplaces 0.4 0.15 0.11 0.25 -0.22 -0.19 0.34 0.41 0.46 0.24 -0.23 0.33
GarageCars 0.6 0.54 0.42 0.36 -0.45 -0.5 0.43 0.44 0.47 0.47 -0.37 0.36 0.3
GarageArea 0.56 0.48 0.37 0.37 -0.44 -0.44 0.49 0.49 0.47 0.41 -0.36 0.34 0.27 0.88
SalePrice 0.79 0.52 0.51 0.47 -0.64 -0.59 0.61 0.61 0.71 0.56 -0.59 0.53 0.47 0.64 0.62
OverallQual YearBuilt YearRemodAdd MasVnrArea ExterQual BsmtQual TotalBsmtSF 1stFlrSF GrLivArea FullBath KitchenQual TotRmsAbvGrd Fireplaces GarageCars GarageArea SalePrice
1.00 0.75 0.50 0.25 0.00 −0.25 −0.50 −0.75 −1.00



[85]:
```
df.head()
```

[85]:

	OverallQual	YearBuilt	YearRemodAdd	MasVnrArea	ExterQual	BsmtQual	
0	7	2003	2003	196.0	2	2	\
1	6	1976	1976	0.0	3	2	
2	7	2001	2002	162.0	2	2	
3	7	1915	1970	0.0	3	4	
4	8	2000	2000	350.0	2	2	

	TotalBsmtSF	1stFlrSF	GrLivArea	FullBath	KitchenQual	TotRmsAbvGrd	
0	856	856	1710	2	2	8	\
1	1262	1262	1262	2	3	6	
2	920	920	1786	2	2	6	
3	756	961	1717	1	2	7	
4	1145	1145	2198	2	2	9	

```
   Fireplaces  GarageCars  GarageArea  SalePrice
0           0           2         548     208500
1           1           2         460     181500
2           1           2         608     223500
3           1           3         642     140000
4           1           3         836     250000
```

[86]:
```
# Previewing which what is the encoding of the categories
[col_dic[i] for i in ["ExterQual", "BsmtQual", "KitchenQual", "GarageFinish"]]
```

[86]:
```
[{0: 'Ex', 1: 'Fa', 2: 'Gd', 3: 'TA'},
 {0: 'Ex', 1: 'Fa', 2: 'Gd', 3: 'None', 4: 'TA'},
 {0: 'Ex', 1: 'Fa', 2: 'Gd', 3: 'TA'},
 {0: 'Fin', 1: 'None', 2: 'RFn', 3: 'Unf'}]
```

4.0.1 Relabelling the categorical data

Label encoders don't consider the trend of categories. They start from index zero and label the categories depending on the sort of data in the dataset. It can reduce the accuracy of the model when we do Machine Learning on the dataset. We should relabel the categorical features considering their trend to solve this issue.

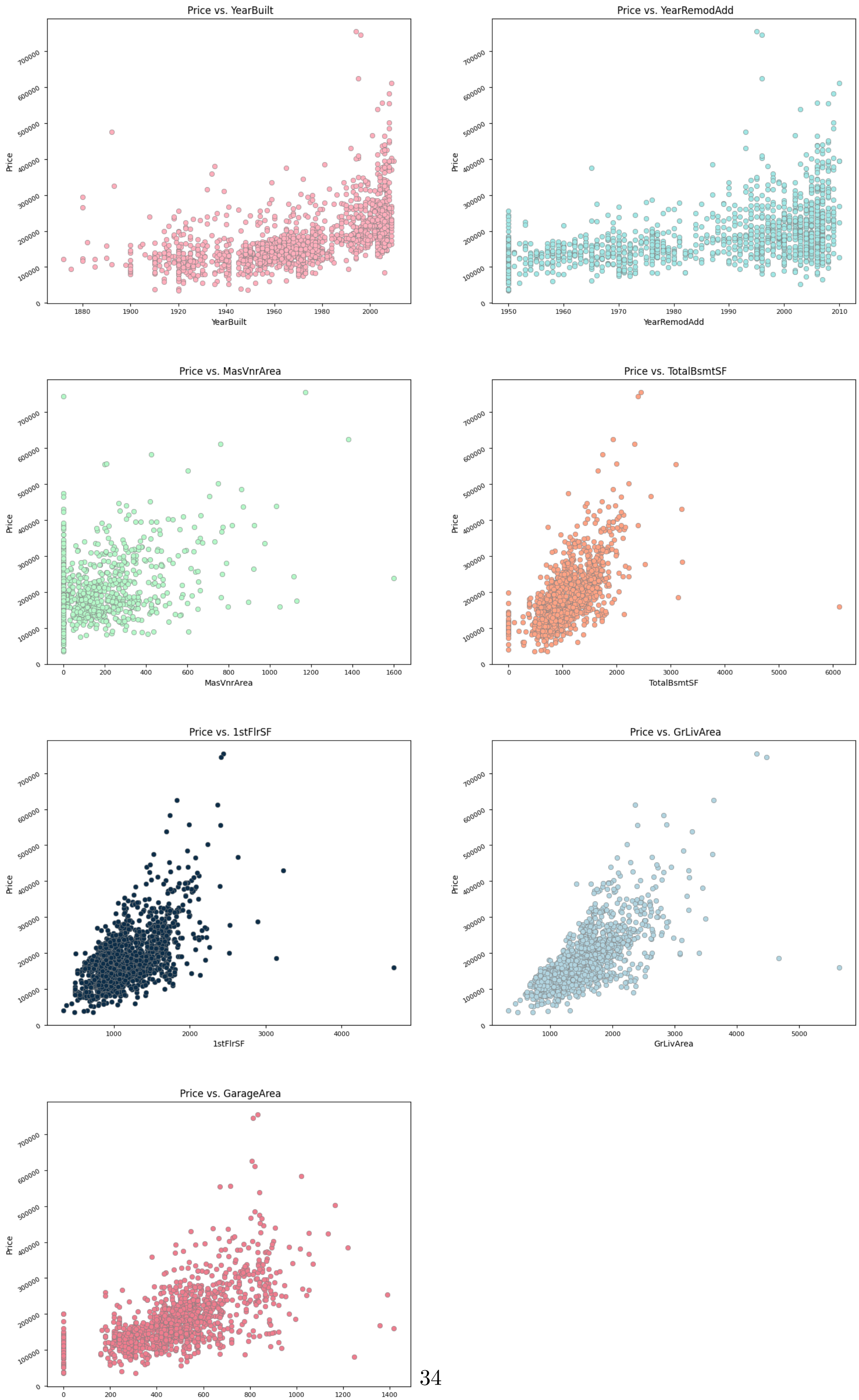
[87]:
```
# Rearrange the categorical values by their real value in a ascending format

df = df.copy(deep=True)
col_dic["ExterQual"] = {0: 4, 1: 1, 2: 3, 3: 2}
df.replace({"ExterQual": col_dic["ExterQual"]}, inplace=True)
col_dic["BsmtQual"] = {0: 4, 1: 1, 2: 3, 3: 0, 4: 2}
df.replace({"BsmtQual": col_dic["BsmtQual"]}, inplace=True)
col_dic["KitchenQual"] = {0: 4, 1: 1, 2: 3, 3: 2}
df.replace({"KitchenQual": col_dic["KitchenQual"]}, inplace=True)
col_dic["GarageFinish"] = {0: 2, 1: 0, 2: 1, 3: 0}
df.replace({"GarageFinish": col_dic["GarageFinish"]}, inplace=True)
```

4.1 Data Visualization

[88]:
```
# Create an object of visualization class for our dataframe
plotter = visualization(df)
```

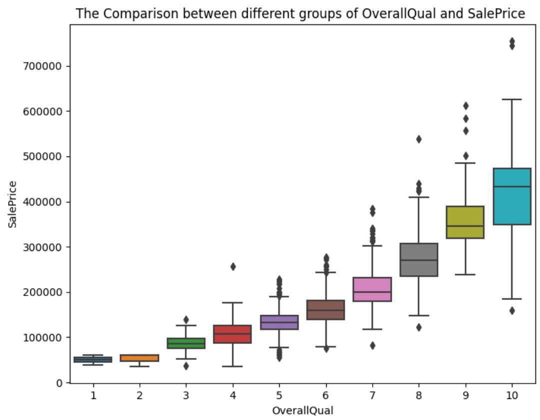
Scatter Plots

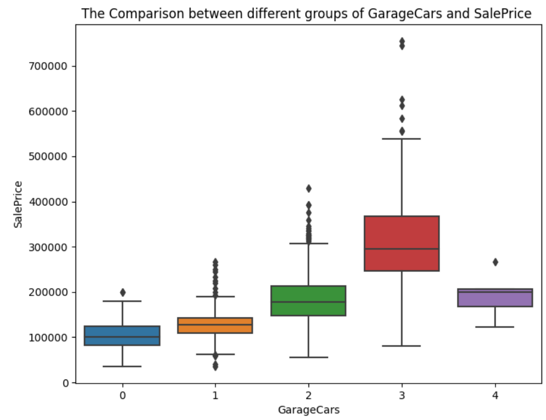
[89]:
```
# Plotting the Scatter plots of price vs different features
plotter.features_price_scatter()
```

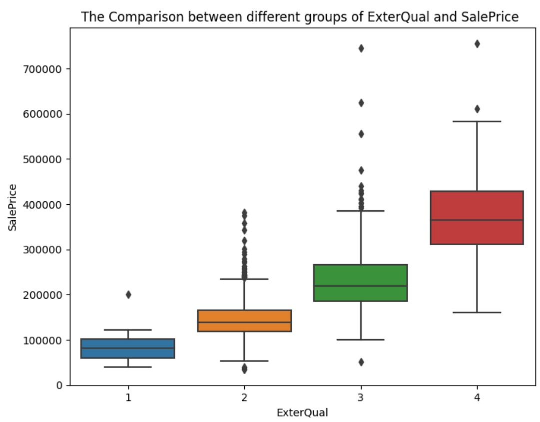

Price vs. YearBuilt
Price
YearBuilt
Price vs. YearRemodAdd
Price
YearRemodAdd
Price vs. MasVnrArea
Price
MasVnrArea
Price vs. TotalBsmtSF
Price
TotalBsmtSF
Price vs. 1stFlrSF
Price
1stFlrSF
Price vs. GrLivArea
Price
GrLivArea
Price vs. GarageArea
Price
GarageArea

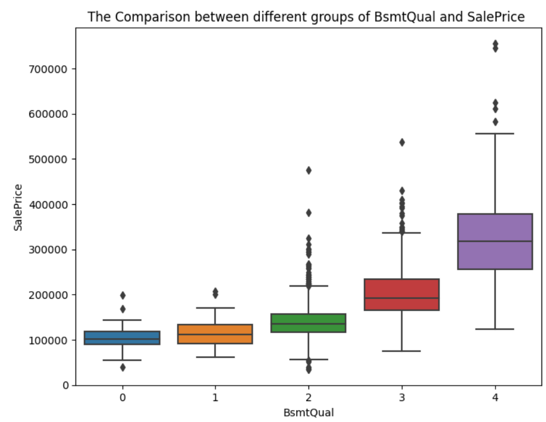
Categorical features boxplots regarding the price

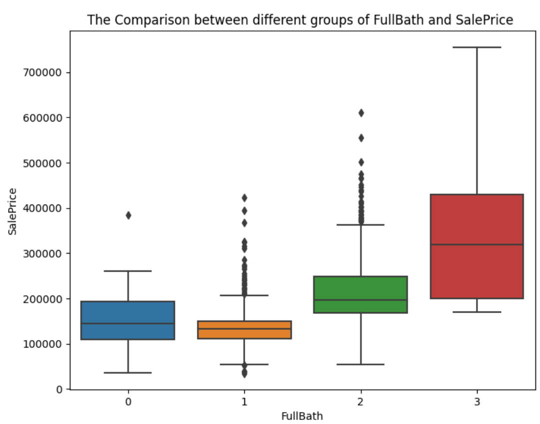
[90]:
```
# Plotting independent categorical variables vs target
plotter.feature_price_boxplot()
```

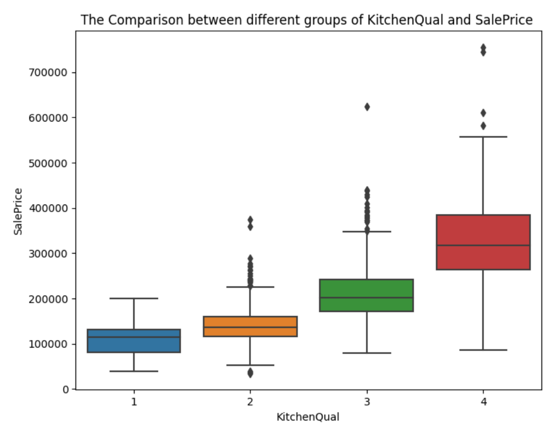

The Comparison between different groups of OverallQual and SalePrice
SalePrice
OverallQual

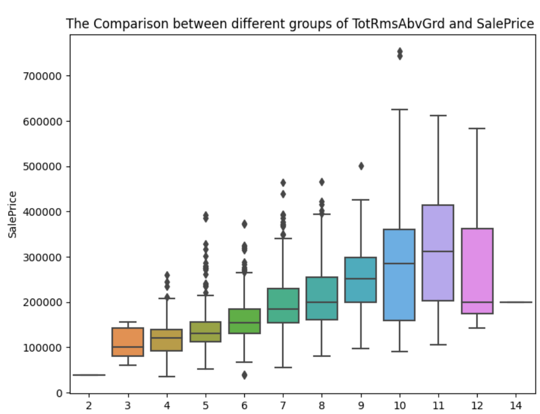

The Comparison between different groups of GarageCars and SalePrice
SalePrice
GarageCars

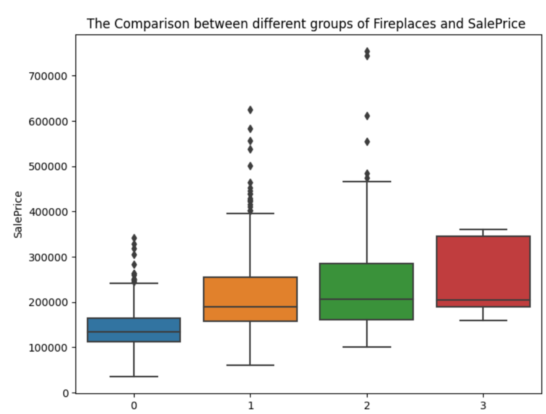

The Comparison between different groups of ExterQual and SalePrice
SalePrice
ExterQual


The Comparison between different groups of BsmtQual and SalePrice
SalePrice
BsmtQual


The Comparison between different groups of FullBath and SalePrice
SalePrice
FullBath


The Comparison between different groups of KitchenQual and SalePrice
SalePrice
KitchenQual


The Comparison between different groups of TotRmsAbvGrd and SalePrice
SalePrice
TotRmsAbvGrd


The Comparison between different groups of Fireplaces and SalePrice
SalePrice
Fireplaces

Result: Considering GarageCars feature, by increasing the number of these features, we have an upward trend for price. However, we can see a reduction when parking spaces increase from 3 to 4. Now we have to check why this is the case. To do so, first, we need to extract two dataframe from our main dataframe. The first one consists of the instances with 3 parking spaces, and the second one includes houses with 4 parking spaces.

[91]:
```python
# Cropping the data frame to two new dataframe
# df_parking_3 : houses with 3 parking
# df_parking_4 : houses with 4 parking
df_parking_4 = df[df["GarageCars"] == 4]
df_parking_3 = df[df["GarageCars"] == 3]

# Get the average of the feature values of cropped data frames
mean_features_4 = df_parking_4.describe().loc["mean"]
mean_features_3 = df_parking_3.describe().loc["mean"]

# Create a dataframe to compare the difference between the average values for␣
↪each group
comparison = pd.DataFrame(
    index=mean_features_4.index,
    data={
        "Average feature values (GarageCars == 3)": mean_features_3.values,
        "Average feature values (GarageCars == 4)": mean_features_4.values,
    },
)
comparison
```

[91]:
```
              Average feature values (GarageCars == 3)
OverallQual                                   7.950276  \
YearBuilt                                  1997.287293
YearRemodAdd                               2000.850829
MasVnrArea                                  285.806630
ExterQual                                     3.099448
BsmtQual                                      3.359116
TotalBsmtSF                                1546.657459
1stFlrSF                                   1562.055249
GrLivArea                                  2084.607735
FullBath                                      2.016575
KitchenQual                                   3.276243
TotRmsAbvGrd                                  8.082873
Fireplaces                                    0.972376
GarageCars                                    3.000000
GarageArea                                  811.574586
SalePrice                                309636.121547
```

```
               Average feature values (GarageCars == 4)
OverallQual                                         5.4
YearBuilt                                        1955.6
YearRemodAdd                                     1983.2
MasVnrArea                                        143.4
ExterQual                                           2.2
BsmtQual                                            2.2
TotalBsmtSF                                      1187.8
1stFlrSF                                         1299.2
GrLivArea                                        1822.4
FullBath                                            1.4
KitchenQual                                         2.2
TotRmsAbvGrd                                        8.0
Fireplaces                                          0.4
GarageCars                                          4.0
GarageArea                                        890.4
SalePrice                                      192655.8
```

regarding the above data, houses with 3 parking have better Overal Quality, External Quality, Basement Quality, and Kitchen Quality rather than houses with 4 parking on average. Furthermore, overall, the former group's houses are bigger than the latter. Also, there is a big gap between the year they've been built. The houses with 4 parking usually had been built almost 40 years before the other group. So basically, the reason why we observe a disorder in price trend by increasing the number of parking is has been found.

This pattern again has been observed in the 'TotRmsAbvGrd' feature. In fact, the houses which has 12 total room above the average has less prices than the ones with 11 above the average. To analyse why this is the case, we repeat the above procedure.

[92]:
```
# Croppiing the data frame to two new dataframe
# df_rooms_12 : houses with 12 rooms above the average
# df_rooms_11 : houses with 11 rooms above the average
df_rooms_12 = df[df["TotRmsAbvGrd"] == 12]
df_rooms_11 = df[df["TotRmsAbvGrd"] == 11]

# Get the average of the feature values of cropped data frames
mean_features_12 = df_rooms_12.describe().loc["mean"]
mean_features_11 = df_rooms_11.describe().loc["mean"]

# Create a dataframe to compare the difference between the average values for␣
 ↪each group
comparison = pd.DataFrame(
    index=mean_features_12.index,
    data={
        "Average feature values (TotRmsAbvGrd == 11)": mean_features_11.values,
        "Average feature values (TotRmsAbvGrd == 12)": mean_features_12.values,
    },
```

```
)
comparison
```

[92]:

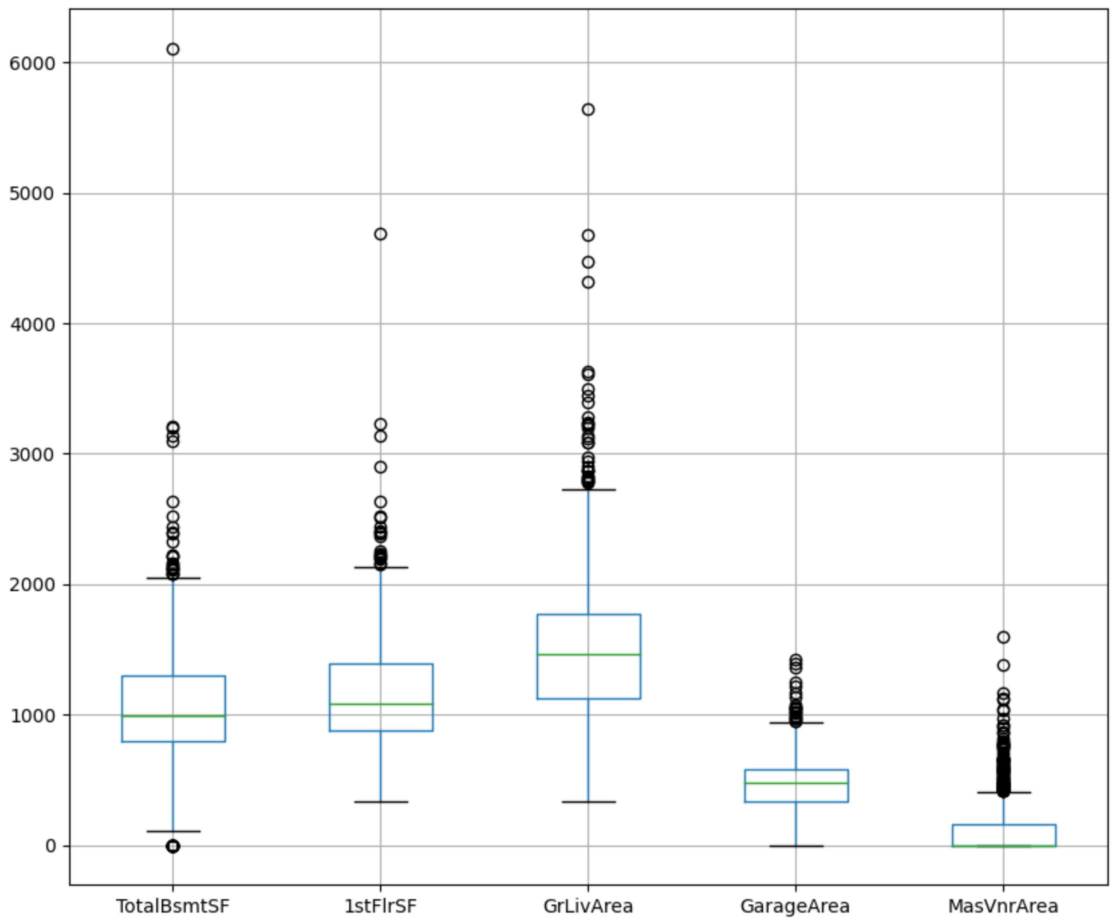
```
                 Average feature values (TotRmsAbvGrd == 11)
OverallQual                                          7.555556  \
YearBuilt                                         1972.333333
YearRemodAdd                                      1995.944444
MasVnrArea                                         298.555556
ExterQual                                            2.944444
BsmtQual                                             3.166667
TotalBsmtSF                                       1365.333333
1stFlrSF                                          1628.111111
GrLivArea                                         2812.000000
FullBath                                             2.444444
KitchenQual                                          3.166667
TotRmsAbvGrd                                        11.000000
Fireplaces                                           1.166667
GarageCars                                           2.500000
GarageArea                                         707.666667
SalePrice                                       318022.000000

                 Average feature values (TotRmsAbvGrd == 12)
OverallQual                                          6.909091
YearBuilt                                         1967.272727
YearRemodAdd                                      1985.454545
MasVnrArea                                         248.818182
ExterQual                                            2.636364
BsmtQual                                             2.454545
TotalBsmtSF                                       1648.090909
1stFlrSF                                          1752.090909
GrLivArea                                         3097.363636
FullBath                                             2.363636
KitchenQual                                          3.000000
TotRmsAbvGrd                                        12.000000
Fireplaces                                           1.181818
GarageCars                                           2.272727
GarageArea                                         721.000000
SalePrice                                       280971.454545
```

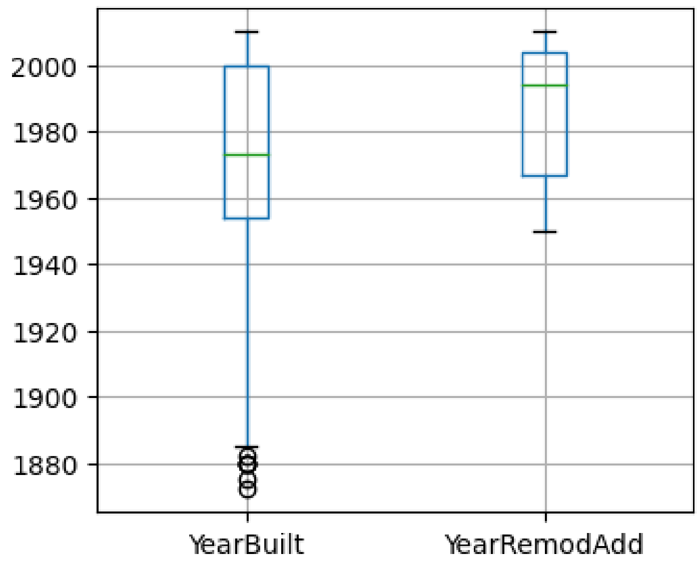
Comparing the columns of the table above, although the houses with one extra room are bigger than the other group on average, their quality in diferrent indexes are inferior to the other houses. Also, these group are usually older than the other buildings. So , the impact of quality and year they are built outweight the size.

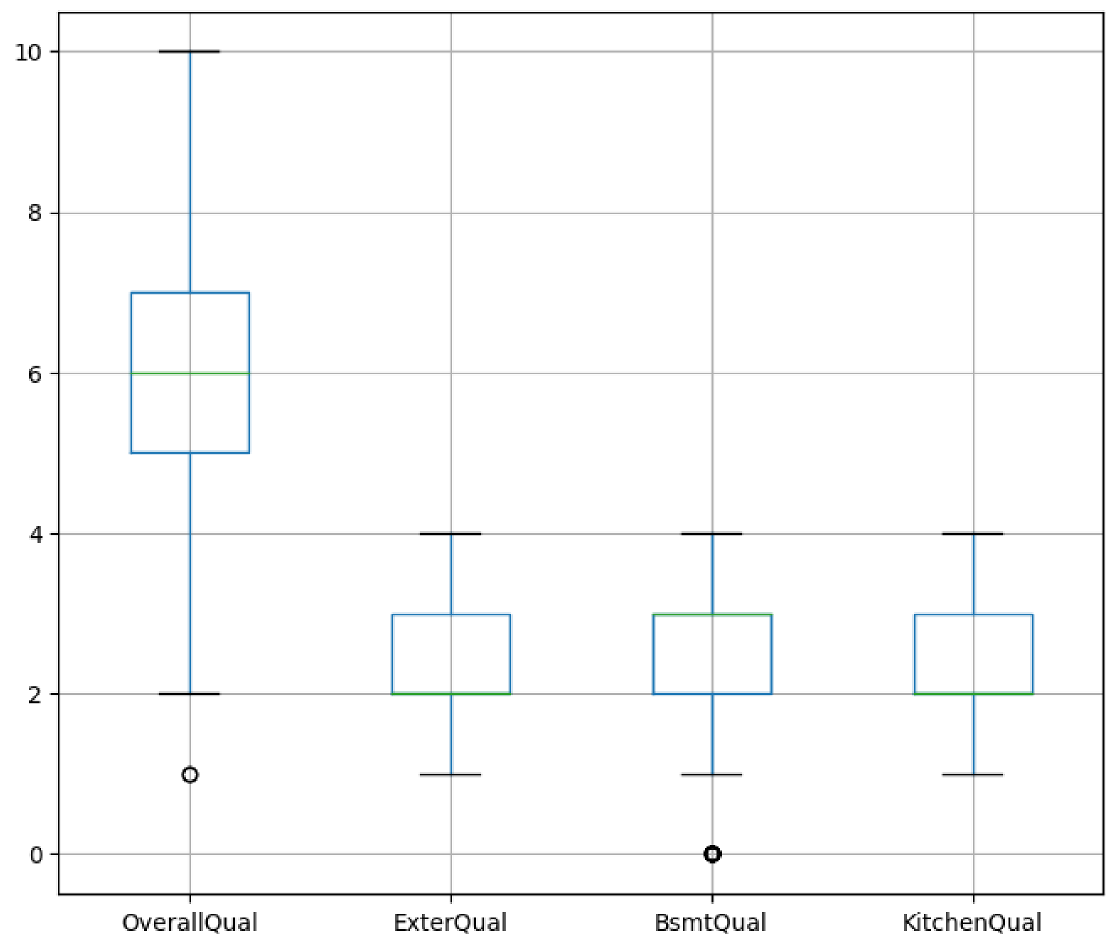
Box-Plot for each feature

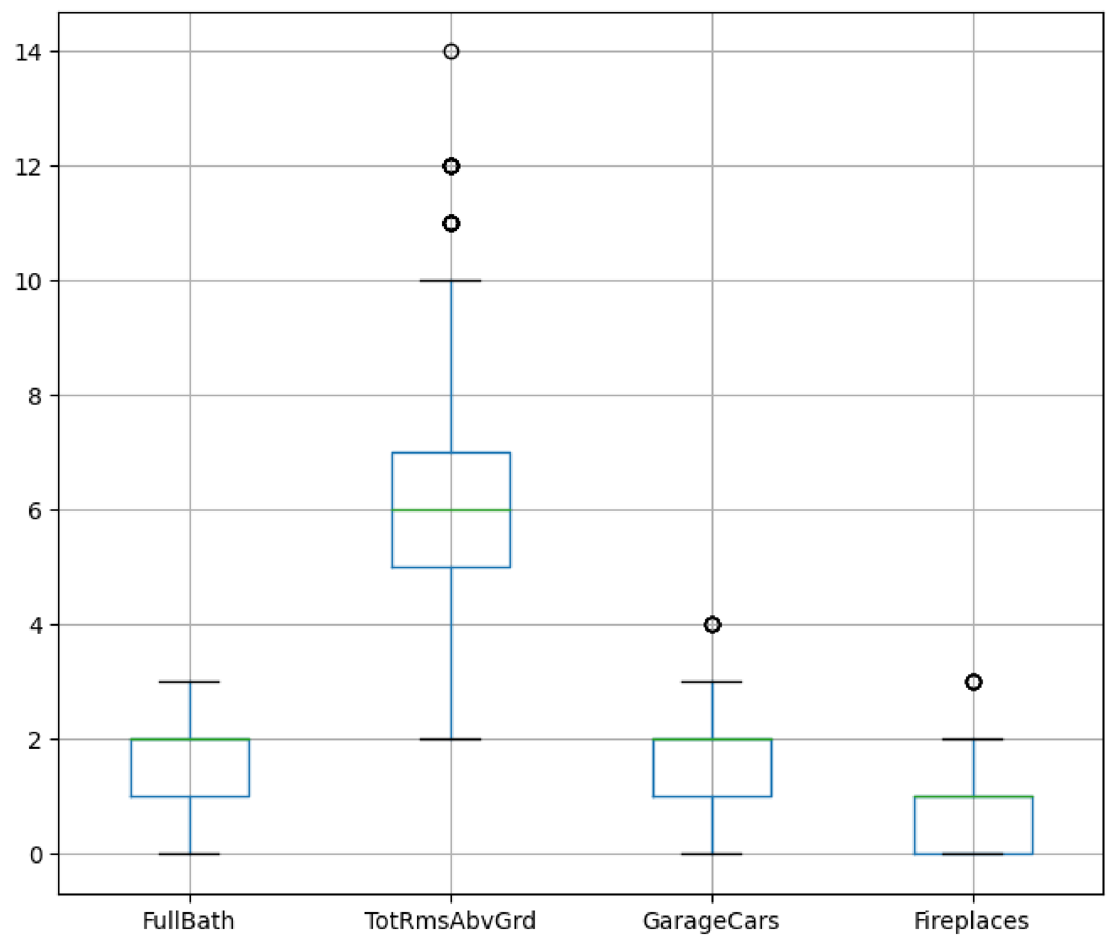
[93]:

```
# Plotting the box plot for all the features to understand their distributions
plotter.features_boxplots()
```

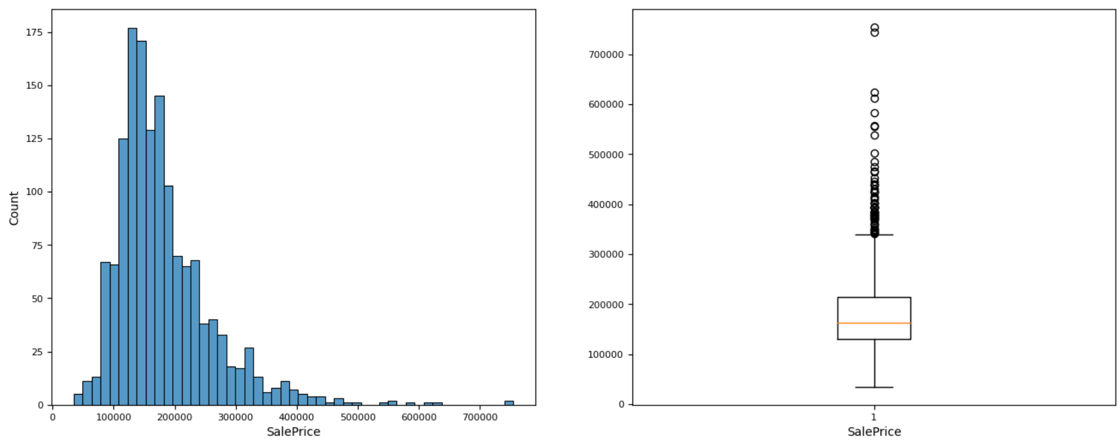

6000
5000
4000
3000
2000
1000
0
TotalBsmtSF
1stFlrSF
GrLivArea
GarageArea
MasVnrArea


2000
1980
1960
1940
1920
1900
1880
YearBuilt
YearRemodAdd


10
8
6
4
2
0
OverallQual
ExterQual
BsmtQual
KitchenQual



[94]:
```
# plotting the price's boxplot and historam

plotter.price_distribution()
```



4.2 Checking for the Outliers

Hypothesis: Considering the SalePrice distribution above, it can be that there are some suspicious instances which can be categorised as outliers. But the chance of being our layer for two is very high. They might be fancy houses with enormous prices. However, we need solid reasons to consider an instance as an outlier and remove it from the database. So, it could be a better approach to create another dataframe with removed potential outliers. Then, we do ML on the original and modified dataframes. Finally, if we see any improvement in the accuracy of our modified model, it can be understood that the removed instances were out layers. Otherwise, they were not.

Method: Z-scores Usually, we calculate the Z-score for detecting outliers, which is the deviation of an instance divided by the standard deviation of the data for all instances. Then those instances with Z-score above 3 standard deviations or below -3 standard deviation will be dropped from the dataset.

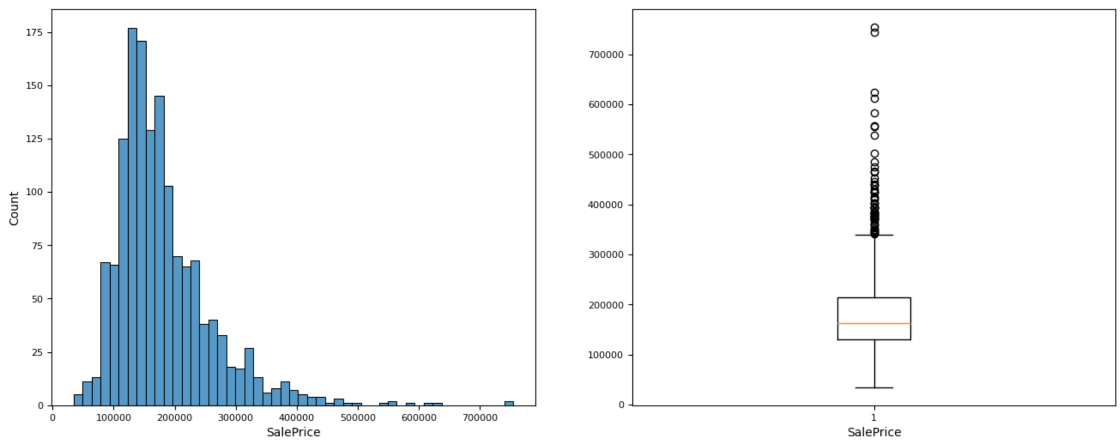
[95]:
```python
# Remove the outlayers and print out the number of outlayers
df_without_outlayers = outlayer_remover(df, calculate_z_scores(df["SalePrice"]))
```

```
22 Houses are detected as outlayers
```

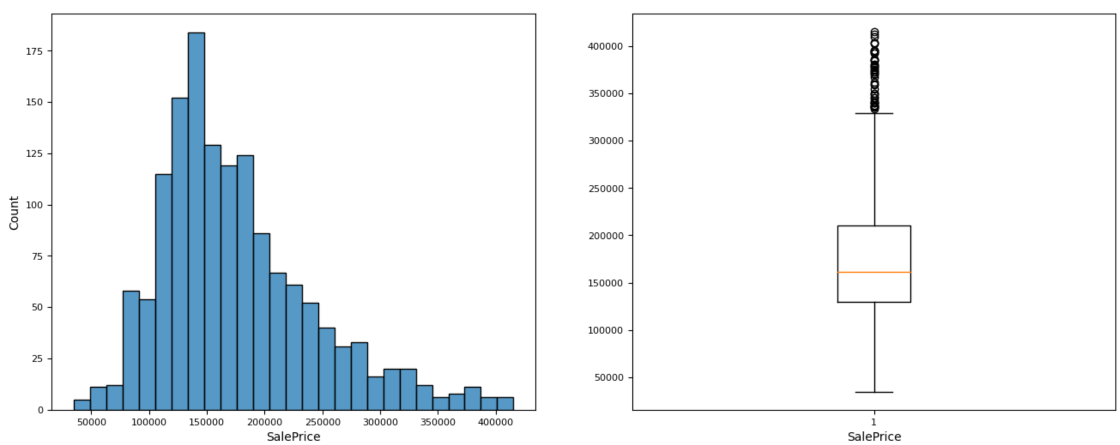
Showing the distributions after and before removing outlayers:

[96]:
```python
new_plotter = visualization(df_without_outlayers)

print("The old Price distribution:")
plotter.price_distribution()
plt.show()
print("\n\n\nNew Price distribution:")
new_plotter.price_distribution()
```

```
The old Price distribution:
```



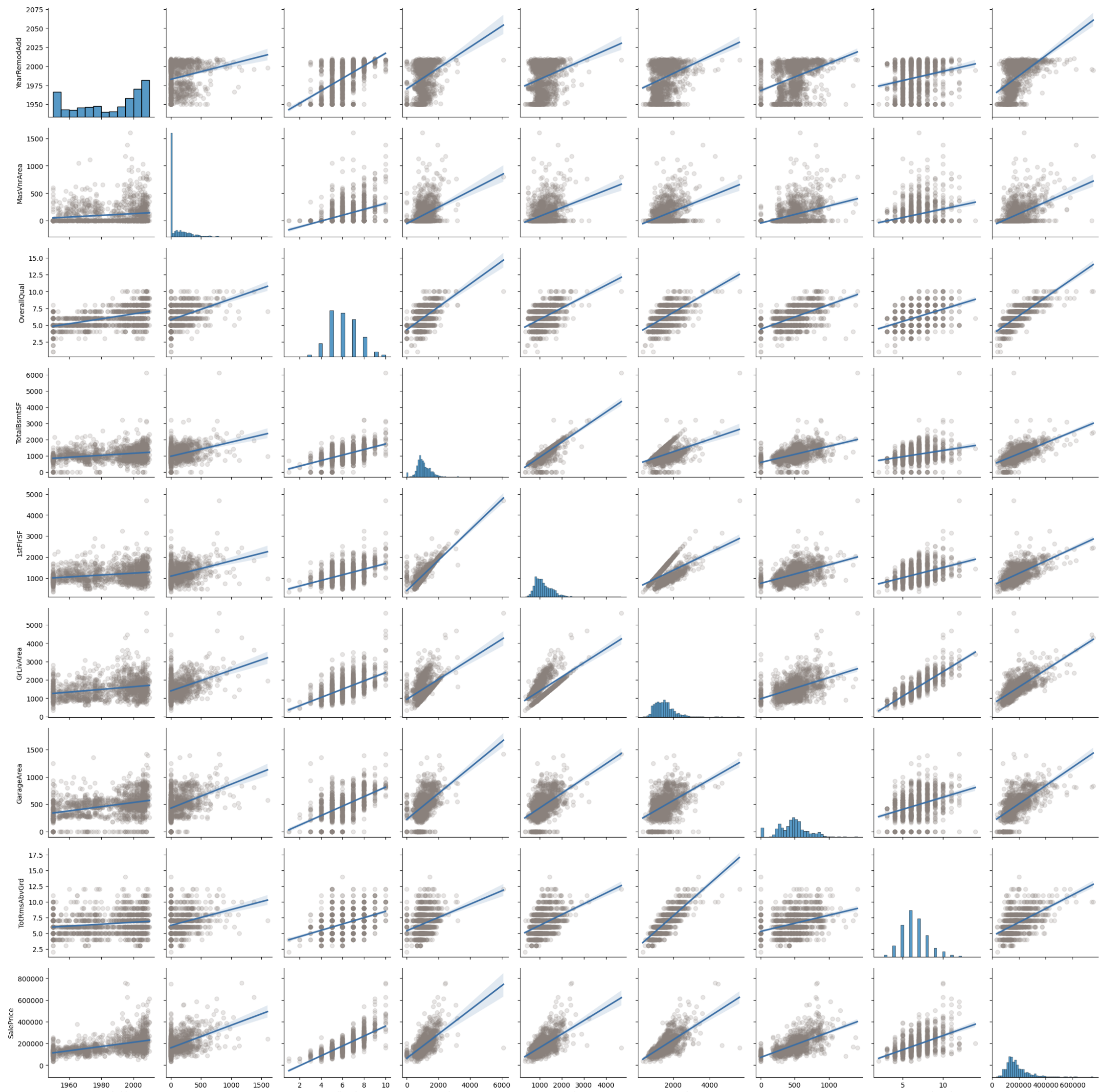
New Price distribution:



Result: Considering visualizations we can not judge these houses are outlayers or not. Because they might be fancy houses with high prices. To investigate it, we can do ML and discuss if these are outlayers or not.

Pair-Plots The discription of this plot can be found within the function.

[97]:
```
# Plotting the pairplots
plotter.pair_plots()
```



4.2.1 Box-Cox Data Transformation

Regarding the distribution of features in the PairPlots graph, some non-categorical features have a normal distribution. However, all of them are skewed to the left. these features are 'OverallQual', 'TotalBsmtSF', '1stFlrSF', 'GrLivArea', 'GarageArea', 'SalePrice', and 'MasVnrArea'. We have to transform their distribution to remove this skewness in these features. Our options are Log-Transformation and Box-Cox Transformation. Although log transformation is an easy method to use in comparison with the box-cox method, we choose box-cox as it completely solves this issue, while log can't.

The problem with this method is that box-cox can be used just for positive data, whereas we have many zeroes in our dataset. So, we shift our data by adding an epsilon that is an extremely small number to all of our data, then we implement the box-cox transformation.

[98]:
```python
# Deploying Box-Cox transformation
transformed_df, skew_dataframe, lambda_dic = box_cox_transformer(df)
```

[99]:
```python
transformed_df.head()
```

[99]:

	OverallQual	YearBuilt	YearRemodAdd	MasVnrArea	ExterQual	BsmtQual
0	4.470077	2003.0	2003.0	4.695494	3.0	3.0 \
1	3.829085	1976.0	1976.0	-34.333996	2.0	3.0
2	4.470077	2001.0	2002.0	4.544780	3.0	3.0
3	4.470077	1915.0	1970.0	-34.333996	2.0	2.0
4	5.089582	2000.0	2000.0	5.146264	3.0	3.0

	TotalBsmtSF	1stFlrSF	GrLivArea	FullBath	KitchenQual	TotRmsAbvGrd
0	63.443149	5.235744	7.621712	2.0	3.0	8.0 \
1	78.155763	5.460253	7.303623	2.0	2.0	6.0
2	65.954972	5.277966	7.667293	2.0	3.0	6.0
3	59.331570	5.303382	7.625994	1.0	3.0	7.0
4	74.183155	5.404626	7.885039	2.0	3.0	9.0

	Fireplaces	GarageCars	GarageArea	SalePrice
0	1.000000e-09	2.0	33.850183	7.932606
1	1.000000e+00	2.0	31.183954	7.878259
2	1.000000e+00	2.0	35.532111	7.959614
3	1.000000e+00	3.0	36.443966	7.774951
4	1.000000e+00	3.0	41.190778	8.002870

[100]:
```python
lambda_dic
```

[100]:
```
{'OverallQual': 0.7622455698532536,
 'TotalBsmtSF': 0.5230460974863727,
 '1stFlrSF': -0.07883214245105231,
 'GrLivArea': 0.006304877505247834,
 'GarageArea': 0.43788711651301193,
 'SalePrice': -0.07692401520136988,
 'MasVnrArea': -0.045219976590401896}
```

Result The table belew, compares the original skewness of our data with the new skewness. Skewness = 0 is the ideal. Overall we can see almost all of the skewnesses have improvements.

[101]:
```python
skew_dataframe
```

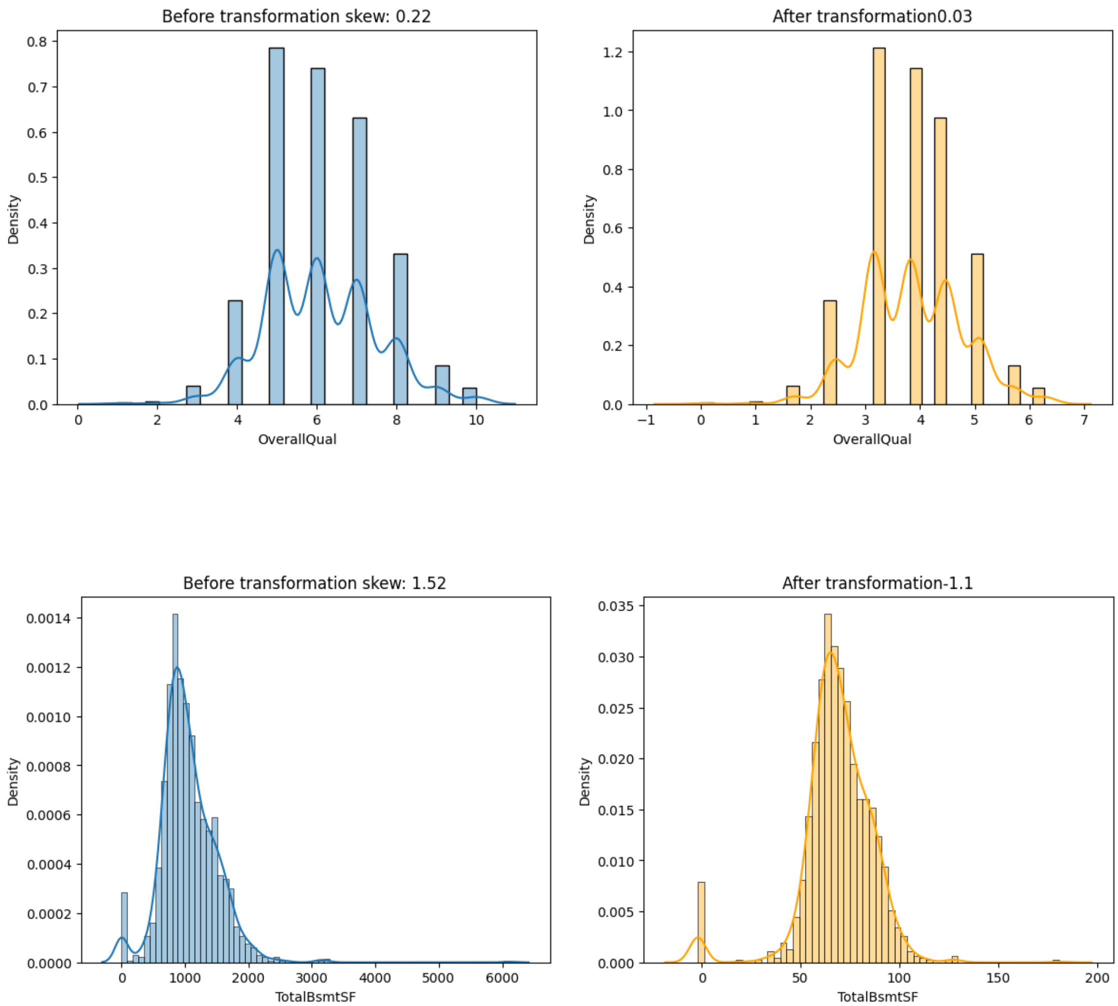
[101]:

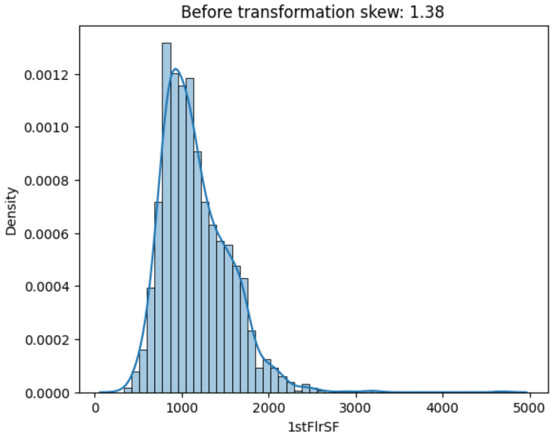
	Original Data Skew	Transformed Data Skew
OverallQual	0.216944	0.028409
TotalBsmtSF	1.524255	-1.098511
1stFlrSF	1.376757	-0.001098
GrLivArea	1.366560	0.000195
GarageArea	0.179981	-1.932475

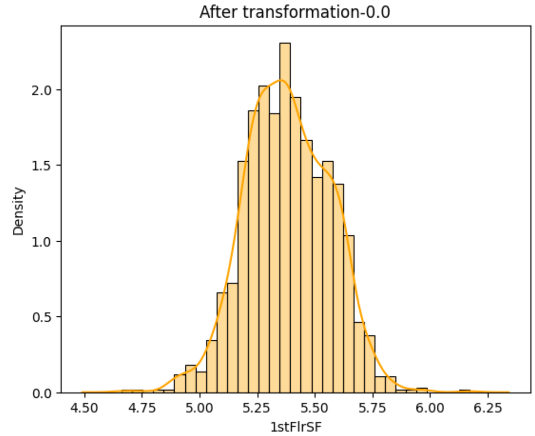
```
SalePrice              1.882876            -0.008653
MasVnrArea             2.677616             0.389984
```

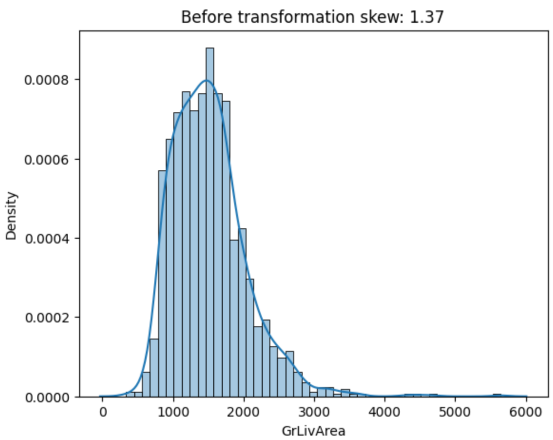
Comparing the distributions before and after transformation

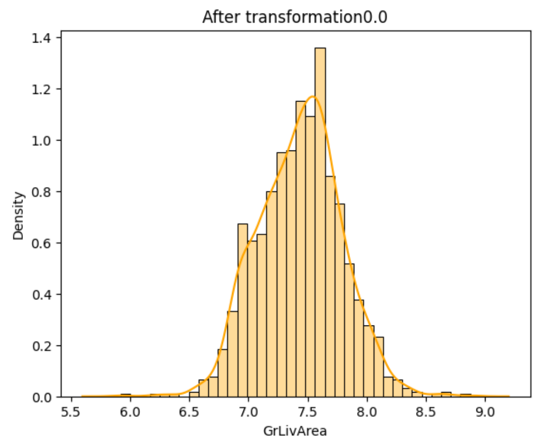
[102]:
```
plotter.distribution_comparison(transformed_df, skew_dataframe)
```

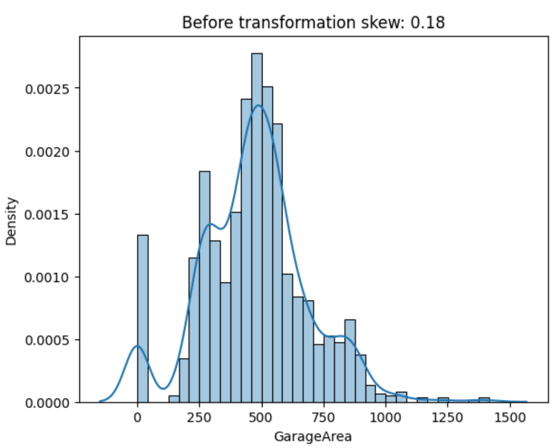


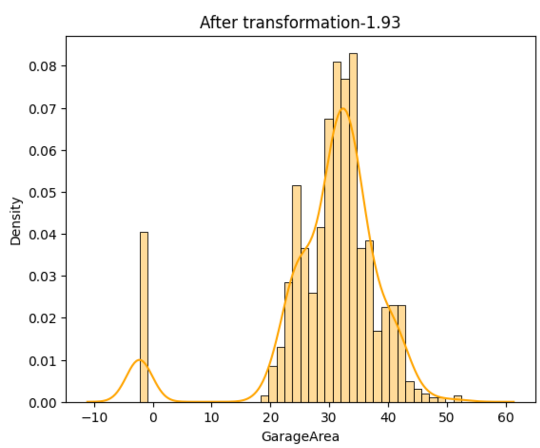

Before transformation skew: 1.38
Density
1stFlrSF

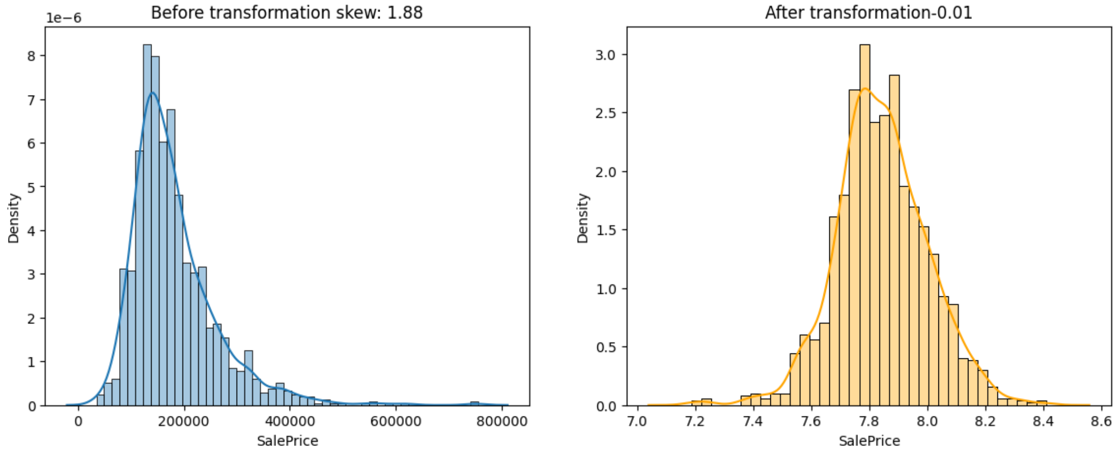

After transformation-0.0
Density
1stFlrSF


Before transformation skew: 1.37
Density
GrLivArea


After transformation0.0
Density
GrLivArea


Before transformation skew: 0.18
Density
GarageArea


After transformation-1.93
Density
GarageArea



5 Machine Learning

[103]:
```
# Creating an object for machine learning process
ml = machine_learning(df, transformed_df)
```

5.0.1 Scikit-Learn Multivariate Liniear Regression model

[104]:
```
# Scikit-Learn Multivariate Liniear Regression model
ml.multi_regr_with_scikit()
```

```
Normal model score is:  0.7642659946885764
Transformed model score is:  0.8388684160659758
```

[104]:

	coef	average	Effect on price
OverallQual	0.0391	6.099315	0.238483
YearBuilt	0.0006	1971.267808	1.182761
YearRemodAdd	0.0008	1984.865753	1.587893
MasVnrArea	-0.0001	103.117123	-0.010312
ExterQual	0.0025	2.395890	0.005990
BsmtQual	0.0135	2.514384	0.033944
TotalBsmtSF	0.0006	1057.429452	0.634458
1stFlrSF	0.0895	1162.626712	104.055091
GrLivArea	0.1298	1515.463699	196.707188
FullBath	-0.0041	1.565068	-0.006417
KitchenQual	0.0203	2.511644	0.050986
TotRmsAbvGrd	0.0010	6.517808	0.006518
Fireplaces	0.0227	0.613014	0.013915
GarageCars	0.0164	1.767123	0.028981

```
GarageArea    0.0009   472.980137         0.425682
```

Result: considering the models' scores, the linear regression model obviously performed better on the transformed dataset. the accuracy for the original dataset is %76.43, while after box-cox transformation, our accuracy has improved up to %83.89. Consequently, we can use the transformed dataset rather than the original one to deploy further improvements to our model.

Also, regarding the "effect on price" index, we can see that the parameters which refer to the area of the house, such as "GrLivArea" and "1stFlrSF", have the most impact on the price. And the second important variable group that significantly affects the price is the date of construction and reconstruction. Features such as "YearBuilt" and "YearRemodAdd" moderately impact price.

5.0.2 Scikit-Learn Random Forest Ensemble learning method

[105]:
```
ml.multi_regr_with_randomforest()
```

```
Normal model score is:  0.8065805549434145
Transformed model score is:  0.81960352953889
```

5.0.3 statsmodels library Multivariate linear model via least squares

statmodels library offers more statistical metrics compared with scikit-learn that are useful for model selection. We shortly describe these metrics here and how they help the process of feature selection.

BIC: Bayesian Information Criterion (BIC) is a statistical criterion for model selection that balances the trade-off between model fit and complexity. It is calculated by adding a penalty term to the log-likelihood of the model, with the penalty term increasing as the number of parameters in the model increases. BIC is commonly used to select the best model among competing models in various statistical and machine-learning applications. The Smaller BIC, the simpler model.

AIC : Akaike Information Criterion (AIC) is a statistical criterion for model selection that measures the quality of a model based on how well it fits the data and how many parameters it has. It is calculated by adding a penalty term to the log-likelihood of the model, where the penalty term is proportional to the number of parameters in the model. AIC is commonly used to select the best model among competing models in various statistical and machine-learning applications. The Smaller BIC, the better model.

R-squared: R-squared is a statistical measure representing the proportion of variation in a dependent variable explained by an independent variable or variables in a linear regression model. It ranges between 0 and 1, where 0 indicates that the model doesn't explain any variation, and 1 indicates a perfect fit between the model and the data. We can consider it as the accuracy of our model.

p-values: P-values are a statistical measure used to determine the probability of observing a result as extreme or more extreme than the one obtained, assuming that the null hypothesis is true. The p-value is typically compared to a significance level (e.g., 0.05), and if it is less than the significance level, the null hypothesis is rejected.

VIF: Variance Inflation Factor (VIF) is a statistical measure used to detect the presence of multicollinearity in regression analysis. It measures how much the variance of the estimated regression coefficients is inflated due to multicollinearity among the independent variables. VIF values greater than 3 indicate the presence of multicollinearity, with higher values indicating more severe multicollinearity.

5.0.4 Model Selection Approach:

Initially, we are looking for models with a higher r-squared. Models with bigger r-squared are better fitted on data points. Once we measure the r-squared of our model, we evaluate the p-values of our features. p-values higher than 0.05 indicate that the coefficient of our independent variables is significant. Therefore, we should drop features with p-values over 0.05. Once we've done it, we should check BIC and AIC factors, they can be a good measure to analyse is our new model is simpler than our previous model or not. If removing a column doesn't affect the r-squared that much and also reduces the BIC and AIC, we choose the new model as the best model.

5.0.5 Note:

In all the models below, we expect a high multicollinearity dues to a weak feature selection process, for instance, the features such as YearBuilt and YearRemodAdd, or features which refer to the size of the house such as TotalBsmtSF, 1stFlrSF, and GrLivArea have compelling multicollinearity effect on each other.

[106]:
```
ml.multi_regr_with_lstat()
```

```
BIC is:  -2866.5939459025276
AIC is:  -2946.5701001785374
r-squared for original dataframe is:  0.8036786061691842
r-squared for modified dataframe is:  0.8497640074088291
```

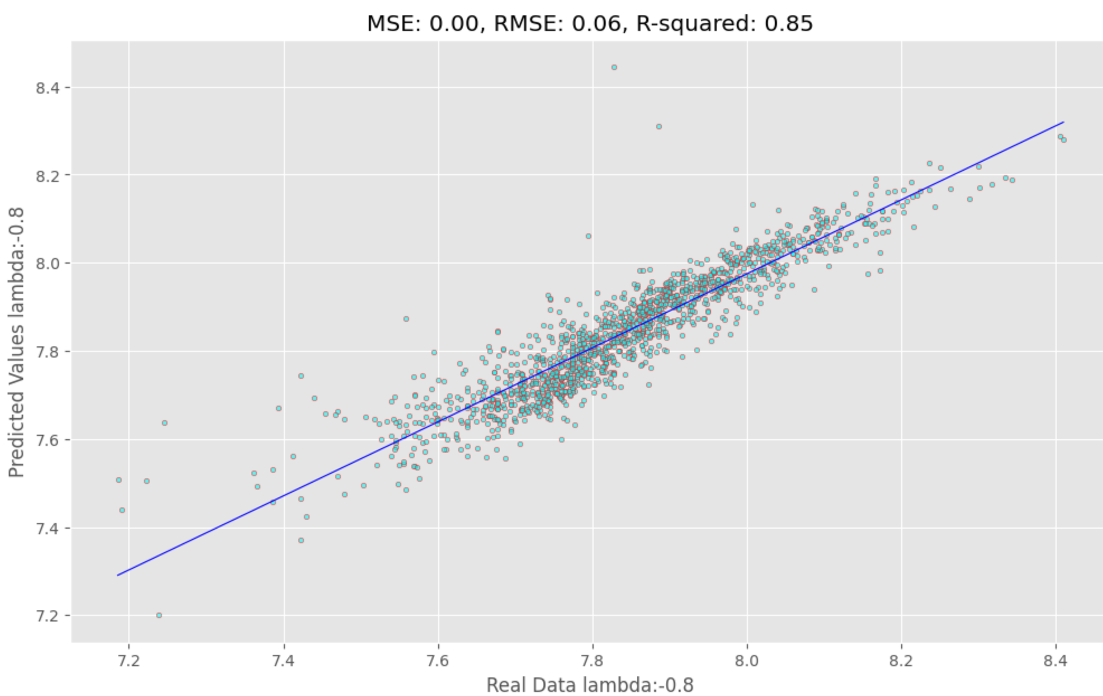
[106]:
```
(                   coef  p-values  VIF Factor
 const          3.345330     0.000       0.000
 OverallQual    0.039077     0.000      73.749
 YearBuilt      0.000576     0.000    8885.052
 YearRemodAdd   0.000788     0.000    8740.769
 MasVnrArea    -0.000078     0.496       1.824
 ExterQual      0.002484     0.657      53.278
 BsmtQual       0.013489     0.002      28.861
 TotalBsmtSF    0.000555     0.004      32.237
 1stFlrSF       0.089462     0.000      45.348
 GrLivArea      0.129825     0.000      47.400
 FullBath      -0.004131     0.428      18.941
 KitchenQual    0.020311     0.000      37.191
 TotRmsAbvGrd   0.001034     0.630      56.702
 Fireplaces     0.022721     0.000       2.700
 GarageCars     0.016385     0.005      35.402
 GarageArea     0.000857     0.036      31.574,
 <class 'statsmodels.iolib.summary.Summary'>
 """
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              SalePrice   R-squared:                       0.847
Model:                            OLS   Adj. R-squared:                  0.845
Method:                 Least Squares   F-statistic:                     398.2
Date:                Tue, 06 Jun 2023   Prob (F-statistic):               0.00
Time:                        19:18:53   Log-Likelihood:                 1489.3
No. Observations:                1095   AIC:                            -2947.
Df Residuals:                    1079   BIC:                            -2867.
Df Model:                          15
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
const            3.3453      0.312     10.710      0.000       2.732
3.958
OverallQual      0.0391      0.004      9.853      0.000       0.031
0.047
YearBuilt        0.0006      0.000      5.439      0.000       0.000
0.001
YearRemodAdd     0.0008      0.000      5.897      0.000       0.001
0.001
MasVnrArea   -7.759e-05      0.000     -0.681      0.496      -0.000
0.000
ExterQual        0.0025      0.006      0.444      0.657      -0.009
0.013
BsmtQual         0.0135      0.004      3.073      0.002       0.005
0.022
TotalBsmtSF      0.0006      0.000      2.897      0.004       0.000
0.001
1stFlrSF         0.0895      0.017      5.226      0.000       0.056
0.123
GrLivArea        0.1298      0.013     10.248      0.000       0.105
0.155
FullBath        -0.0041      0.005     -0.792      0.428      -0.014
0.006
KitchenQual      0.0203      0.005      4.451      0.000       0.011
0.029
TotRmsAbvGrd     0.0010      0.002      0.481      0.630      -0.003
0.005
Fireplaces       0.0227      0.003      6.584      0.000       0.016
0.029
GarageCars       0.0164      0.006      2.814      0.005       0.005
0.028
GarageArea       0.0009      0.000      2.102      0.036    5.72e-05
0.002
```

```
==============================================================================
Omnibus:                      537.787   Durbin-Watson:                   2.011
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             7654.169
Skew:                          -1.897   Prob(JB):                         0.00
Kurtosis:                      15.384   Cond. No.                     4.63e+05
==============================================================================

Notes:
 [1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
 [2] The condition number is large, 4.63e+05. This might indicate that there are
 strong multicollinearity or other numerical problems.
 """)
```

[107]:

```
# Plotting the prediction vs real prices
targets, predictions = multireg_statmodels(transformed_df)
prediction_scatter(targets, predictions)
```



[108]:

```
ml = machine_learning(
    df.drop("FullBath", axis=1), transformed_df.drop("FullBath", axis=1)
)
ml.multi_regr_with_lstat()
```

```
BIC is:  -2872.9555006839364
AIC is:  -2947.9331453176956
```

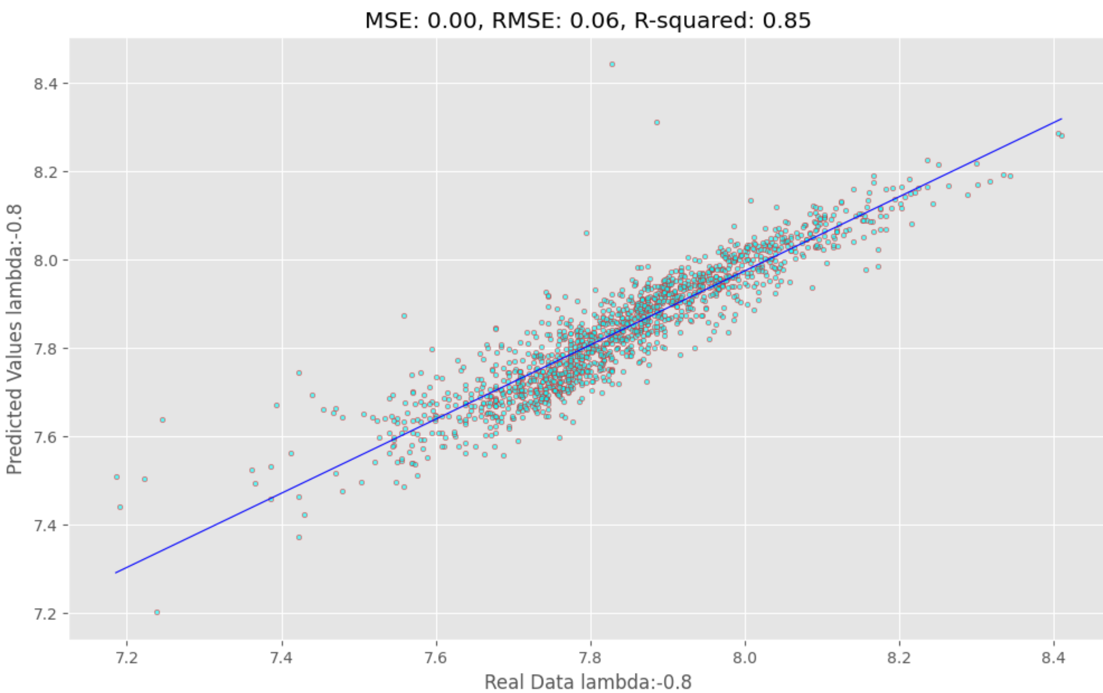
```
r-squared for original dataframe is:  0.8030503588376833
r-squared for modified dataframe is:  0.8495191462177774
```

[108]:
```
(                   coef  p-values  VIF Factor
 const          3.422742     0.000       0.000
 OverallQual    0.038926     0.000      72.939
 YearBuilt      0.000556     0.000    8684.845
 YearRemodAdd   0.000778     0.000    8527.005
 MasVnrArea    -0.000076     0.505       1.819
 ExterQual      0.002363     0.673      52.942
 BsmtQual       0.013530     0.002      28.701
 TotalBsmtSF    0.000568     0.003      31.897
 1stFlrSF       0.089390     0.000      45.231
 GrLivArea      0.126694     0.000      44.279
 KitchenQual    0.020256     0.000      36.972
 TotRmsAbvGrd   0.000842     0.693      55.796
 Fireplaces     0.022991     0.000       2.654
 GarageCars     0.015911     0.006      34.881
 GarageArea     0.000899     0.026      31.202,
 <class 'statsmodels.iolib.summary.Summary'>
 """
                             OLS Regression Results
 ==============================================================================
 Dep. Variable:              SalePrice   R-squared:                       0.847
 Model:                            OLS   Adj. R-squared:                  0.845
 Method:                 Least Squares   F-statistic:                     426.8
 Date:                Tue, 06 Jun 2023   Prob (F-statistic):               0.00
 Time:                        19:18:55   Log-Likelihood:                 1489.0
 No. Observations:                1095   AIC:                            -2948.
 Df Residuals:                    1080   BIC:                            -2873.
 Df Model:                          14
 Covariance Type:            nonrobust
================================================================================
                    coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
 const            3.4227      0.297     11.538      0.000       2.841
4.005
 OverallQual      0.0389      0.004      9.828      0.000       0.031
0.047
 YearBuilt        0.0006      0.000      5.409      0.000       0.000
0.001
 YearRemodAdd     0.0008      0.000      5.850      0.000       0.001
0.001
 MasVnrArea   -7.588e-05      0.000     -0.667      0.505      -0.000
0.000
 ExterQual        0.0024      0.006      0.422      0.673      -0.009
```

```
0.013
 BsmtQual          0.0135      0.004      3.083      0.002       0.005
0.022
 TotalBsmtSF       0.0006      0.000      2.973      0.003       0.000
0.001
 1stFlrSF          0.0894      0.017      5.223      0.000       0.056
0.123
 GrLivArea         0.1267      0.012     10.528      0.000       0.103
0.150
 KitchenQual       0.0203      0.005      4.440      0.000       0.011
0.029
 TotRmsAbvGrd      0.0008      0.002      0.394      0.693      -0.003
0.005
 Fireplaces        0.0230      0.003      6.696      0.000       0.016
0.030
 GarageCars        0.0159      0.006      2.747      0.006       0.005
0.027
 GarageArea        0.0009      0.000      2.223      0.026       0.000
0.002
 ==============================================================================
 Omnibus:                      535.398   Durbin-Watson:                   2.015
 Prob(Omnibus):                  0.000   Jarque-Bera (JB):             7512.127
 Skew:                          -1.890   Prob(JB):                         0.00
 Kurtosis:                      15.262   Cond. No.                     4.39e+05
 ==============================================================================

 Notes:
 [1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
 [2] The condition number is large, 4.39e+05. This might indicate that there are
 strong multicollinearity or other numerical problems.
 """)
```

[109]:

```
# Plotting the prediction vs real prices
targets, predictions = multireg_statmodels(transformed_df.drop(["FullBath"],
 ↪axis=1))
prediction_scatter(targets, predictions)
```



[110]:
```python
ml = machine_learning(
    df.drop(["FullBath", "ExterQual"], axis=1),
    transformed_df.drop(["FullBath", "ExterQual"], axis=1),
)
ml.multi_regr_with_lstat()
```

```
BIC is:  -2879.773330685338
AIC is:  -2949.7524656768464
r-squared for original dataframe is:  0.8004283611714987
r-squared for modified dataframe is:  0.8493866329058775
```

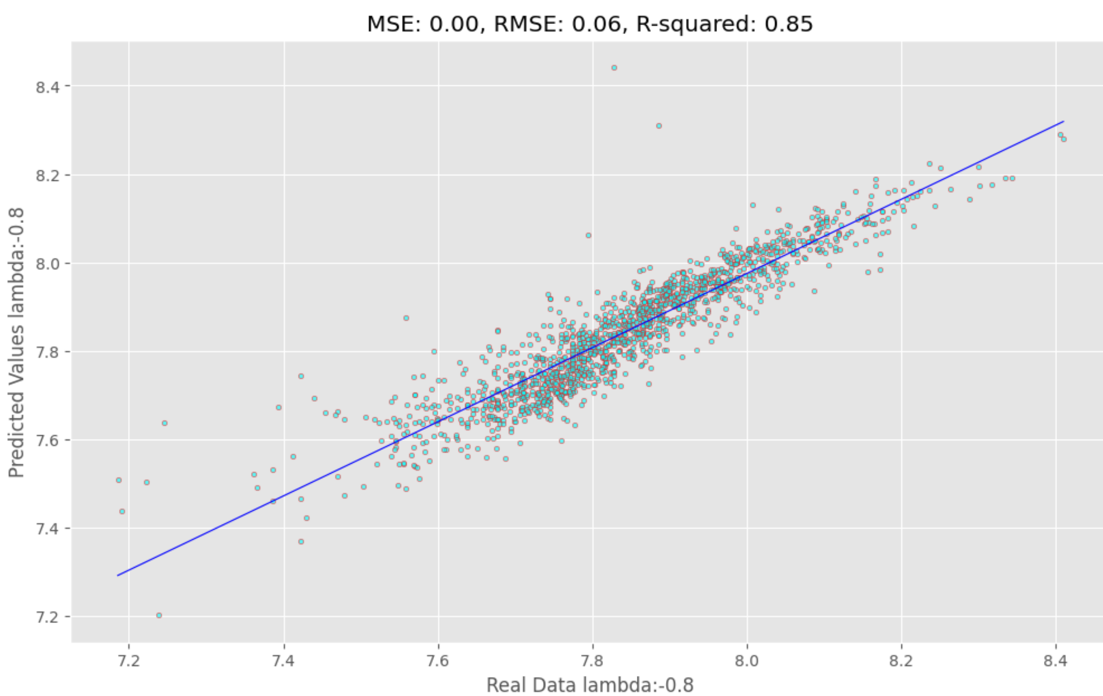
[110]: (

	coef	p-values	VIF Factor
const	3.402262	0.000	0.000
OverallQual	0.039393	0.000	65.447
YearBuilt	0.000562	0.000	8635.902
YearRemodAdd	0.000782	0.000	8494.169
MasVnrArea	-0.000076	0.505	1.816
BsmtQual	0.013636	0.002	28.324
TotalBsmtSF	0.000569	0.003	31.876
1stFlrSF	0.089556	0.000	45.228
GrLivArea	0.126819	0.000	44.276
KitchenQual	0.020940	0.000	30.838
TotRmsAbvGrd	0.000828	0.698	55.757
Fireplaces	0.022907	0.000	2.635
GarageCars	0.015993	0.006	34.868

```
 GarageArea    0.000896     0.027      31.194,
 <class 'statsmodels.iolib.summary.Summary'>
 """
                             OLS Regression Results
 ==============================================================================
 Dep. Variable:              SalePrice   R-squared:                       0.847
 Model:                            OLS   Adj. R-squared:                  0.845
 Method:                 Least Squares   F-statistic:                     459.9
 Date:                Tue, 06 Jun 2023   Prob (F-statistic):               0.00
 Time:                        19:18:58   Log-Likelihood:                 1488.9
 No. Observations:                1095   AIC:                            -2950.
 Df Residuals:                    1081   BIC:                            -2880.
 Df Model:                          13
 Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
 const           3.4023      0.293     11.630      0.000       2.828
3.976
 OverallQual     0.0394      0.004     10.362      0.000       0.032
0.047
 YearBuilt       0.0006      0.000      5.530      0.000       0.000
0.001
 YearRemodAdd    0.0008      0.000      5.899      0.000       0.001
0.001
 MasVnrArea   -7.592e-05      0.000     -0.667      0.505      -0.000
0.000
 BsmtQual        0.0136      0.004      3.114      0.002       0.005
0.022
 TotalBsmtSF     0.0006      0.000      2.982      0.003       0.000
0.001
 1stFlrSF        0.0896      0.017      5.236      0.000       0.056
0.123
 GrLivArea       0.1268      0.012     10.546      0.000       0.103
0.150
 KitchenQual     0.0209      0.004      4.911      0.000       0.013
0.029
 TotRmsAbvGrd    0.0008      0.002      0.388      0.698      -0.003
0.005
 Fireplaces      0.0229      0.003      6.685      0.000       0.016
0.030
 GarageCars      0.0160      0.006      2.764      0.006       0.005
0.027
 GarageArea      0.0009      0.000      2.217      0.027       0.000
0.002
 ==============================================================================
```

```
 Omnibus:                      533.171   Durbin-Watson:                   2.017
 Prob(Omnibus):                  0.000   Jarque-Bera (JB):             7428.296
 Skew:                          -1.882   Prob(JB):                         0.00
 Kurtosis:                      15.192   Cond. No.                     4.33e+05
 ==============================================================================

 Notes:
 [1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
 [2] The condition number is large, 4.33e+05. This might indicate that there are
 strong multicollinearity or other numerical problems.
 """)
```

[111]:
```
# Plotting the prediction vs real prices
targets, predictions = multireg_statmodels(
    transformed_df.drop(["FullBath", "ExterQual"], axis=1)
)
prediction_scatter(targets, predictions)
```



[112]:
```
ml = machine_learning(
    df.drop(["FullBath", "ExterQual", "MasVnrArea"], axis=1),
    transformed_df.drop(["FullBath", "ExterQual", "MasVnrArea"], axis=1),
)
ml.multi_regr_with_lstat()
```

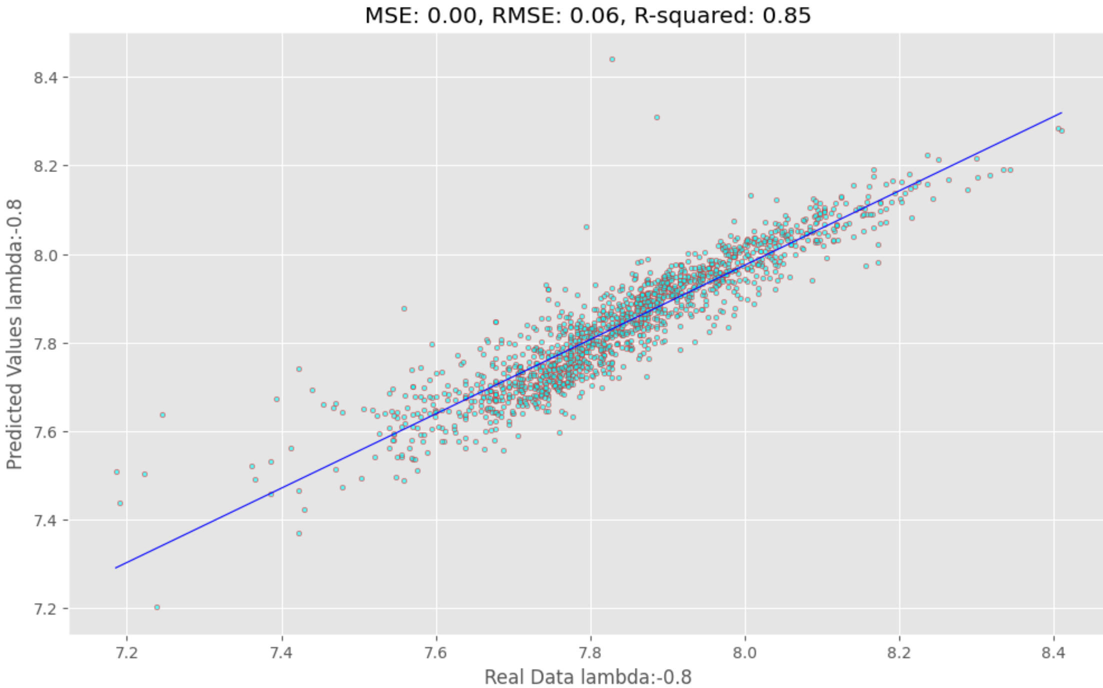
```
BIC is:  -2886.320958727809
AIC is:  -2951.3015840770668
r-squared for original dataframe is:  0.795556012923421
r-squared for modified dataframe is:  0.8491895496551417
```

[112]:

```
(                  coef  p-values  VIF Factor
 const         3.426023     0.000       0.000
 OverallQual   0.039127     0.000      64.746
 YearBuilt     0.000547     0.000    8474.927
 YearRemodAdd  0.000788     0.000    8302.511
 BsmtQual      0.013819     0.002      28.281
 TotalBsmtSF   0.000560     0.003      31.630
 1stFlrSF      0.089084     0.000      45.220
 GrLivArea     0.127023     0.000      43.409
 KitchenQual   0.020977     0.000      30.831
 TotRmsAbvGrd  0.000767     0.719      55.658
 Fireplaces    0.022781     0.000       2.635
 GarageCars    0.015712     0.007      34.867
 GarageArea    0.000904     0.025      31.090,
 <class 'statsmodels.iolib.summary.Summary'>
 """
                             OLS Regression Results
 ==============================================================================
 Dep. Variable:              SalePrice   R-squared:                       0.847
 Model:                            OLS   Adj. R-squared:                  0.845
 Method:                 Least Squares   F-statistic:                     498.5
 Date:                Tue, 06 Jun 2023   Prob (F-statistic):               0.00
 Time:                        19:18:59   Log-Likelihood:                 1488.7
 No. Observations:                1095   AIC:                            -2951.
 Df Residuals:                    1082   BIC:                            -2886.
 Df Model:                          12
 Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
 const           3.4260      0.290     11.802      0.000       2.856
3.996
 OverallQual     0.0391      0.004     10.352      0.000       0.032
0.047
 YearBuilt       0.0005    9.9e-05      5.523      0.000       0.000
0.001
 YearRemodAdd    0.0008      0.000      5.951      0.000       0.001
0.001
 BsmtQual        0.0138      0.004      3.163      0.002       0.005
0.022
 TotalBsmtSF     0.0006      0.000      2.944      0.003       0.000
```

```
0.001
 1stFlrSF         0.0891      0.017      5.214      0.000       0.056
0.123
 GrLivArea        0.1270      0.012     10.569      0.000       0.103
0.151
 KitchenQual      0.0210      0.004      4.922      0.000       0.013
0.029
 TotRmsAbvGrd     0.0008      0.002      0.360      0.719      -0.003
0.005
 Fireplaces       0.0228      0.003      6.660      0.000       0.016
0.029
 GarageCars       0.0157      0.006      2.724      0.007       0.004
0.027
 GarageArea       0.0009      0.000      2.239      0.025       0.000
0.002
 ==============================================================================
 Omnibus:                      532.598   Durbin-Watson:                   2.020
 Prob(Omnibus):                  0.000   Jarque-Bera (JB):             7407.298
 Skew:                          -1.880   Prob(JB):                         0.00
 Kurtosis:                      15.174   Cond. No.                     4.30e+05
 ==============================================================================

 Notes:
 [1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
 [2] The condition number is large, 4.3e+05. This might indicate that there are
 strong multicollinearity or other numerical problems.
 """)
```

[113]:
```python
# Plotting the prediction vs real prices
targets, predictions = multireg_statmodels(
    transformed_df.drop(["FullBath", "ExterQual", "MasVnrArea"], axis=1)
)
prediction_scatter(targets, predictions)
```



[114]:
```
ml = machine_learning(
    df.drop(["FullBath", "ExterQual", "MasVnrArea", "TotRmsAbvGrd"], axis=1),
    transformed_df.drop(
        ["FullBath", "ExterQual", "MasVnrArea", "TotRmsAbvGrd"], axis=1
    ),
)
ml.multi_regr_with_lstat()
```

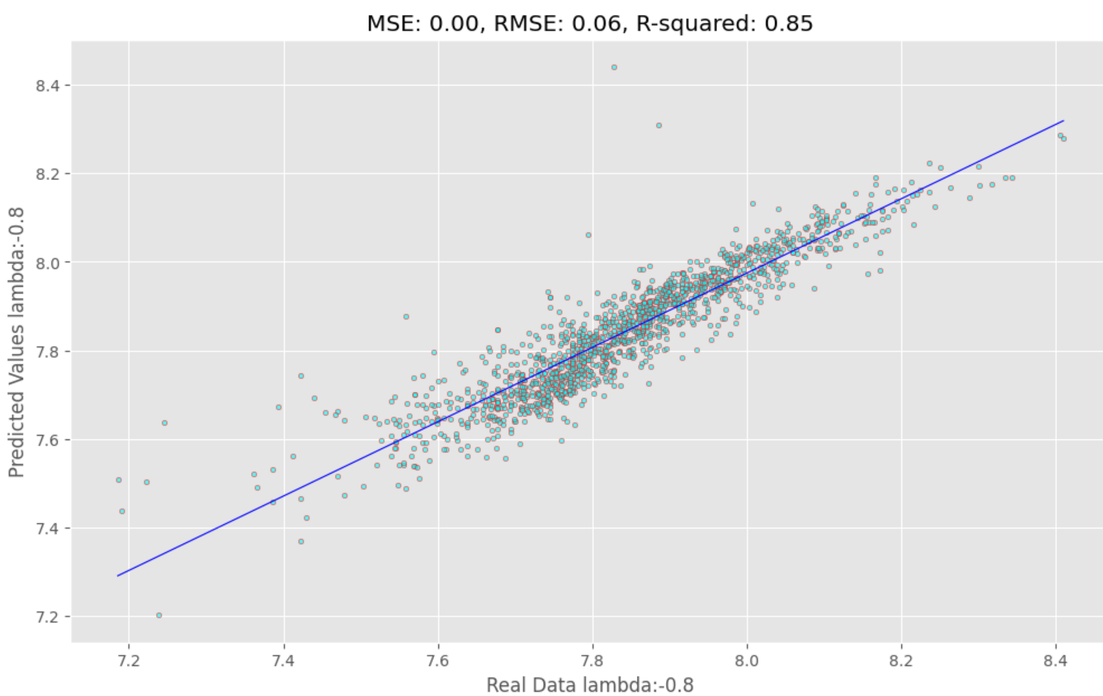
```
BIC is:  -2893.1883289724483
AIC is:  -2953.1704446794556
r-squared for original dataframe is:  0.7950242103826386
r-squared for modified dataframe is:  0.8491191136086412
```

[114]:
```
(                 coef  p-values  VIF Factor
 const        3.416856     0.000       0.000
 OverallQual  0.039082     0.000      64.742
 YearBuilt    0.000545     0.000    8474.278
 YearRemodAdd 0.000786     0.000    8283.997
 BsmtQual     0.013758     0.002      28.193
 TotalBsmtSF  0.000560     0.003      31.581
 1stFlrSF     0.088880     0.000      45.206
 GrLivArea    0.130263     0.000      19.026
 KitchenQual  0.020994     0.000      30.772
 Fireplaces   0.022677     0.000       2.621
 GarageCars   0.015965     0.005      34.603
```

```
 GarageArea    0.000884     0.027      30.891,
 <class 'statsmodels.iolib.summary.Summary'>
 """
                             OLS Regression Results
 ==============================================================================
 Dep. Variable:              SalePrice   R-squared:                       0.847
 Model:                            OLS   Adj. R-squared:                  0.845
 Method:                 Least Squares   F-statistic:                     544.2
 Date:                Tue, 06 Jun 2023   Prob (F-statistic):               0.00
 Time:                        19:19:01   Log-Likelihood:                 1488.6
 No. Observations:                1095   AIC:                            -2953.
 Df Residuals:                    1083   BIC:                            -2893.
 Df Model:                          11
 Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
 const           3.4169      0.289     11.821      0.000       2.850
3.984
 OverallQual     0.0391      0.004     10.350      0.000       0.032
0.046
 YearBuilt       0.0005   9.88e-05      5.513      0.000       0.000
0.001
 YearRemodAdd    0.0008      0.000      5.943      0.000       0.001
0.001
 BsmtQual        0.0138      0.004      3.152      0.002       0.005
0.022
 TotalBsmtSF     0.0006      0.000      2.945      0.003       0.000
0.001
 1stFlrSF        0.0889      0.017      5.207      0.000       0.055
0.122
 GrLivArea       0.1303      0.008     16.358      0.000       0.115
0.146
 KitchenQual     0.0210      0.004      4.928      0.000       0.013
0.029
 Fireplaces      0.0227      0.003      6.656      0.000       0.016
0.029
 GarageCars      0.0160      0.006      2.789      0.005       0.005
0.027
 GarageArea      0.0009      0.000      2.211      0.027    9.96e-05
0.002
 ==============================================================================
 Omnibus:                      532.155   Durbin-Watson:                   2.020
 Prob(Omnibus):                  0.000   Jarque-Bera (JB):             7377.431
 Skew:                          -1.879   Prob(JB):                         0.00
 Kurtosis:                      15.148   Cond. No.                     4.29e+05
```

```
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 4.29e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
""")
```

[115]:
```
# Plotting the prediction vs real prices
targets, predictions = multireg_statmodels(
    transformed_df.drop(["FullBath", "ExterQual", "MasVnrArea",␣
 ↪"TotRmsAbvGrd"], axis=1)
)
prediction_scatter(targets, predictions)
```



[116]:
```
ml = machine_learning(
    df.drop(
        ["FullBath", "ExterQual", "MasVnrArea", "TotRmsAbvGrd", "GarageArea"],␣
 ↪axis=1
    ),
    transformed_df.drop(
        ["FullBath", "ExterQual", "MasVnrArea", "TotRmsAbvGrd", "GarageArea"],␣
 ↪axis=1
```

```
    ),
)
ml.multi_regr_with_lstat()
```

```
BIC is:  -2895.254309765134
AIC is:  -2950.237915829891
r-squared for original dataframe is:  0.7939248952801108
r-squared for modified dataframe is:  0.8473809468842513
```

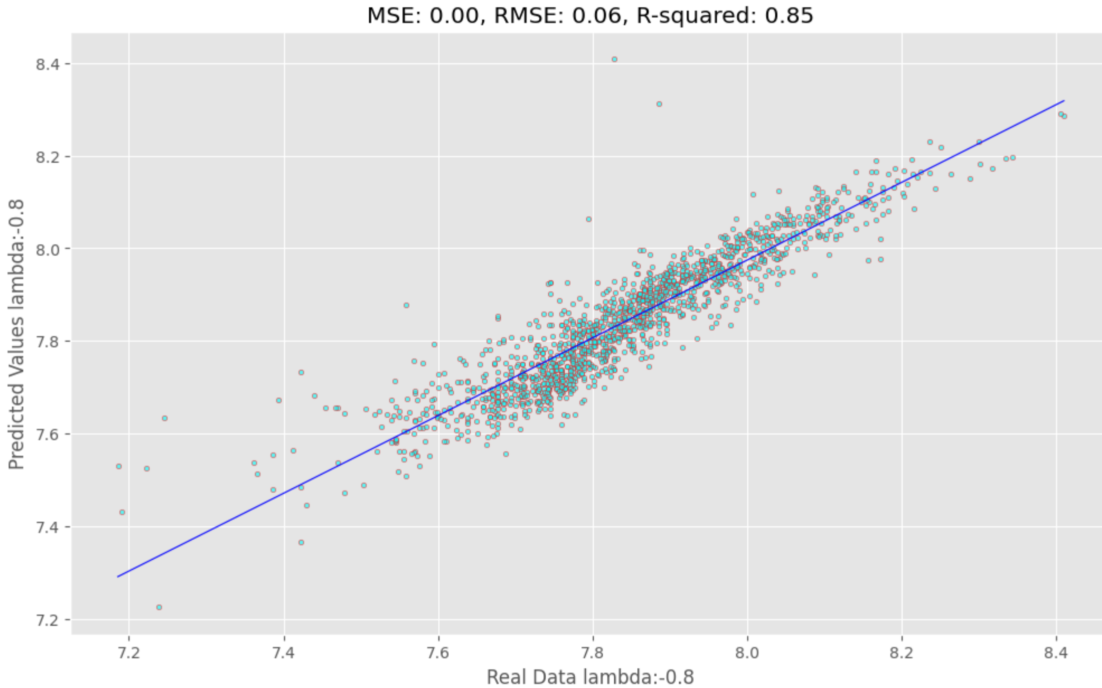
[116]:

```
(                  coef  p-values  VIF Factor
 const         3.463554     0.000       0.000
 OverallQual   0.039197     0.000      64.740
 YearBuilt     0.000555     0.000    8471.827
 YearRemodAdd  0.000758     0.000    8280.054
 BsmtQual      0.012511     0.004      27.995
 TotalBsmtSF   0.000597     0.002      31.188
 1stFlrSF      0.089159     0.000      45.029
 GrLivArea     0.129350     0.000      18.955
 KitchenQual   0.021469     0.000      30.663
 Fireplaces    0.022786     0.000       2.598
 GarageCars    0.026169     0.000      11.482,
 <class 'statsmodels.iolib.summary.Summary'>
 """
                            OLS Regression Results
 ==============================================================================
 Dep. Variable:              SalePrice   R-squared:                       0.846
 Model:                            OLS   Adj. R-squared:                  0.845
 Method:                 Least Squares   F-statistic:                     596.0
 Date:                Tue, 06 Jun 2023   Prob (F-statistic):               0.00
 Time:                        19:19:02   Log-Likelihood:                 1486.1
 No. Observations:                1095   AIC:                            -2950.
 Df Residuals:                    1084   BIC:                            -2895.
 Df Model:                          10
 Covariance Type:            nonrobust
================================================================================
                    coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
 const            3.4636      0.289     11.993      0.000       2.897
4.030
 OverallQual      0.0392      0.004     10.363      0.000       0.032
0.047
 YearBuilt        0.0006   9.88e-05      5.616      0.000       0.000
0.001
 YearRemodAdd     0.0008      0.000      5.750      0.000       0.000
0.001
 BsmtQual         0.0125      0.004      2.886      0.004       0.004
```

```
0.021
 TotalBsmtSF      0.0006      0.000      3.143      0.002      0.000
0.001
 1stFlrSF         0.0892      0.017      5.214      0.000      0.056
0.123
 GrLivArea        0.1293      0.008     16.236      0.000      0.114
0.145
 KitchenQual      0.0215      0.004      5.037      0.000      0.013
0.030
 Fireplaces       0.0228      0.003      6.677      0.000      0.016
0.029
 GarageCars       0.0262      0.003      7.715      0.000      0.020
0.033
 ==============================================================================
 Omnibus:                      514.725   Durbin-Watson:                   2.024
 Prob(Omnibus):                  0.000   Jarque-Bera (JB):             6497.375
 Skew:                          -1.829   Prob(JB):                         0.00
 Kurtosis:                      14.359   Cond. No.                     4.27e+05
 ==============================================================================

 Notes:
 [1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
 [2] The condition number is large, 4.27e+05. This might indicate that there are
 strong multicollinearity or other numerical problems.
 """)
```

[117]:

```
# Plotting the prediction vs real prices
targets, predictions = multireg_statmodels(
    transformed_df.drop(
        ["FullBath", "ExterQual", "MasVnrArea", "TotRmsAbvGrd", "GarageArea"],␣
 ↪axis=1
    )
)
prediction_scatter(targets, predictions)
```



Result: Among All the above regressor models, The statsmodel linear regression model outperforms, and also, among all the statsmodel models, the least model is the best model considering the lowest BIC and AIC, also the significance of the coefficients regarding p-values. However, with respect to the weak feature engineering of the dataset, we can see that the variance inflation factor for the features is too high, which illustrates a massive multicollinearity amongst the features. all in all, the best regression model here is the statsmodel with 11 features that are:

OverallQual, YearBuilt, YearRemodAdd, MasVnrArea, BsmtQual, TotalBsmtSF, 1stFlrSF, GrLivArea, KitchenQua, Fireplaces, and GarageCars.

5.0.6 Comparing Model with and without outlayers:

[118]:
```
(
    transformed_df_without_outlayers,
    skew_dataframe_without_outlayers,
    lambda_dic_without_outlayers,
) = box_cox_transformer(df_without_outlayers)
transformed_df, skew_dataframe, lambda_dic = box_cox_transformer(df)
```

[119]:
```
comparison = machine_learning(
    transformed_df.drop(
        ["FullBath", "ExterQual", "MasVnrArea", "TotRmsAbvGrd", "GarageArea"],␣
 ↪axis=1
    ),
```

```
    transformed_df_without_outlayers.drop(
        ["FullBath", "ExterQual", "MasVnrArea", "TotRmsAbvGrd", "GarageArea"],␣
 ↪axis=1
    ),
)
comparison.multi_regr_with_lstat()
```

```
BIC is:  1652.8173140450403
AIC is:  1598.005823778882
r-squared for original dataframe is:  0.8473809468842513
r-squared for modified dataframe is:  0.8390410060048632
```

[119]:

```
(                   coef  p-values  VIF Factor
 const        -10.754153     0.000       0.000
 OverallQual    0.272156     0.000      59.512
 YearBuilt      0.005139     0.000    8516.907
 YearRemodAdd   0.006190     0.000    9419.520
 BsmtQual       0.092301     0.011      33.595
 TotalBsmtSF    0.005993     0.000      51.408
 1stFlrSF       0.496347     0.000    2319.841
 GrLivArea      0.624196     0.000     930.614
 KitchenQual    0.194016     0.000      30.442
 Fireplaces     0.252747     0.000       2.636
 GarageCars     0.196880     0.000      11.444,
 <class 'statsmodels.iolib.summary.Summary'>
 """
                             OLS Regression Results
 ==============================================================================
 Dep. Variable:              SalePrice   R-squared:                       0.853
 Model:                            OLS   Adj. R-squared:                  0.852
 Method:                 Least Squares   F-statistic:                     618.6
 Date:                Tue, 06 Jun 2023   Prob (F-statistic):               0.00
 Time:                        19:19:03   Log-Likelihood:                -788.00
 No. Observations:                1078   AIC:                             1598.
 Df Residuals:                    1067   BIC:                             1653.
 Df Model:                          10
 Covariance Type:            nonrobust
================================================================================
                    coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
 const          -10.7542      2.274     -4.728      0.000     -15.217
-6.291
 OverallQual      0.2722      0.026     10.659      0.000       0.222
0.322
 YearBuilt        0.0051      0.001      6.497      0.000       0.004
0.007
```

```
 YearRemodAdd     0.0062       0.001      5.800      0.000        0.004
0.008
 BsmtQual         0.0923       0.036      2.540      0.011        0.021
0.164
 TotalBsmtSF      0.0060       0.001      4.069      0.000        0.003
0.009
 1stFlrSF         0.4963       0.102      4.860      0.000        0.296
0.697
 GrLivArea        0.6242       0.036     17.382      0.000        0.554
0.695
 KitchenQual      0.1940       0.035      5.581      0.000        0.126
0.262
 Fireplaces       0.2527       0.028      9.120      0.000        0.198
0.307
 GarageCars       0.1969       0.029      6.897      0.000        0.141
0.253
 ==============================================================================
 Omnibus:                      260.098   Durbin-Watson:                   2.071
 Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1395.429
 Skew:                          -1.002   Prob(JB):                    9.69e-304
 Kurtosis:                       8.201   Cond. No.                     4.14e+05
 ==============================================================================

 Notes:
 [1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
 [2] The condition number is large, 4.14e+05. This might indicate that there are
 strong multicollinearity or other numerical problems.
 """)
```

5.0.7 Result:

As can be seen, removing outliers decreases the accuracy of the model, which means possibly they are not outliers. Also, removing them increases the AIC and BIC significantly, which means it adds up to the complexity of the model, which is not a good result. All in all, Obviously, the 22 houses which we used to consider as outlayer are not real outlayers. They are just houses with prices above the average, such as fancy houses.

6 Note:

Due to the macbooks' M series chipsets' architecture, installing TensorFlow on macbook is tricky. We don't provide any instruction for that in this project. ANN regressor only works properly if you install TensorFlow on your virtual environment, otherwise it won't. Nonetheless, if you want to run ANN regression, you can use the link below to run this notebook on the Kaggle platform. This notebook is public and you can access it with your account.

Link: https://www.kaggle.com/vahidsharifi76/house-price-python-uog

6.1 ANN Regressor

[65]:
```
import tensorflow as tf
from tensorflow.keras.layers import Dense, Input, Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
```

[66]:
```
features, target = feature_target_splitter(transformed_df)
```

[67]:
```
x_train, x_test, y_train, y_test = train_test_split(
    features, target, test_size=0.2, random_state=2200
)
```

[68]:
```
# normalizing the data
s_scaler = StandardScaler()
x_train = s_scaler.fit_transform(x_train)
x_test = s_scaler.transform(x_test)
```

[69]:
```
i = Input(shape=(15,))
x = Dense(90, activation="relu")(i)
x = Dropout(0.2)(x)
x = Dense(45, activation="relu")(x)
x = Dense(15, activation="relu")(x)
x = Dense(5, activation="relu")(x)
x = Dense(30, activation="relu")(x)
x = Dense(90, activation="relu")(x)
x = Dense(45, activation="relu")(x)
x = Dense(14, activation="relu")(x)
x = Dense(1)(x)

model = Model(i, x)
```

[70]:
```
results = []
for i in [1, 5, 10]:
    model.compile(loss="mse", optimizer=Adam(learning_rate=0.001 / i))
    r = model.fit(
        x=x_train,
        y=y_train,
        validation_data=(x_test, y_test),
        batch_size=256,
        epochs=200,
    )
    results.append(r.history["loss"])

model.summary()
```

```
Epoch 1/200
5/5 [==============================] - 1s 57ms/step - loss: 61.2627 - val_loss:
```

```
59.9970
Epoch 2/200
5/5 [==============================] - 0s 10ms/step - loss: 58.3598 - val_loss:
55.6625
Epoch 3/200
5/5 [==============================] - 0s 11ms/step - loss: 52.4328 - val_loss:
46.7008
Epoch 4/200
5/5 [==============================] - 0s 10ms/step - loss: 41.0678 - val_loss:
31.1483
Epoch 5/200
5/5 [==============================] - 0s 9ms/step - loss: 23.5902 - val_loss:
11.2383
Epoch 6/200
5/5 [==============================] - 0s 10ms/step - loss: 7.9898 - val_loss:
7.6896
Epoch 7/200
5/5 [==============================] - 0s 10ms/step - loss: 11.4204 - val_loss:
6.3549
Epoch 8/200
5/5 [==============================] - 0s 15ms/step - loss: 5.7233 - val_loss:
4.1766
Epoch 9/200
5/5 [==============================] - 0s 10ms/step - loss: 5.1391 - val_loss:
5.0518
Epoch 10/200
5/5 [==============================] - 0s 10ms/step - loss: 5.1079 - val_loss:
3.5874
Epoch 11/200
5/5 [==============================] - 0s 10ms/step - loss: 3.4600 - val_loss:
2.4558
Epoch 12/200
5/5 [==============================] - 0s 10ms/step - loss: 3.3432 - val_loss:
2.4104
Epoch 13/200
5/5 [==============================] - 0s 11ms/step - loss: 2.9213 - val_loss:
2.0251
Epoch 14/200
5/5 [==============================] - 0s 11ms/step - loss: 2.3917 - val_loss:
2.0317
Epoch 15/200
5/5 [==============================] - 0s 11ms/step - loss: 2.2530 - val_loss:
1.8069
Epoch 16/200
5/5 [==============================] - 0s 11ms/step - loss: 1.9966 - val_loss:
1.6122
Epoch 17/200
5/5 [==============================] - 0s 10ms/step - loss: 1.9662 - val_loss:
```

```
1.5161
Epoch 18/200
5/5 [==============================] - 0s 10ms/step - loss: 1.9037 - val_loss:
1.4334
Epoch 19/200
5/5 [==============================] - 0s 11ms/step - loss: 1.7559 - val_loss:
1.3821
Epoch 20/200
5/5 [==============================] - 0s 10ms/step - loss: 1.6712 - val_loss:
1.2933
Epoch 21/200
5/5 [==============================] - 0s 10ms/step - loss: 1.5394 - val_loss:
1.2131
Epoch 22/200
5/5 [==============================] - 0s 11ms/step - loss: 1.4000 - val_loss:
1.1572
Epoch 23/200
5/5 [==============================] - 0s 11ms/step - loss: 1.4855 - val_loss:
1.1265
Epoch 24/200
5/5 [==============================] - 0s 11ms/step - loss: 1.4412 - val_loss:
1.0688
Epoch 25/200
5/5 [==============================] - 0s 10ms/step - loss: 1.2675 - val_loss:
0.9927
Epoch 26/200
5/5 [==============================] - 0s 10ms/step - loss: 1.2185 - val_loss:
0.9469
Epoch 27/200
5/5 [==============================] - 0s 10ms/step - loss: 1.2565 - val_loss:
0.9462
Epoch 28/200
5/5 [==============================] - 0s 11ms/step - loss: 1.1214 - val_loss:
0.9181
Epoch 29/200
5/5 [==============================] - 0s 11ms/step - loss: 1.1294 - val_loss:
0.8378
Epoch 30/200
5/5 [==============================] - 0s 11ms/step - loss: 1.0068 - val_loss:
0.8206
Epoch 31/200
5/5 [==============================] - 0s 10ms/step - loss: 0.9698 - val_loss:
0.8128
Epoch 32/200
5/5 [==============================] - 0s 10ms/step - loss: 1.0308 - val_loss:
0.7518
Epoch 33/200
5/5 [==============================] - 0s 11ms/step - loss: 0.9712 - val_loss:
```

```
0.7398
Epoch 34/200
5/5 [==============================] - 0s 10ms/step - loss: 0.9176 - val_loss:
0.7495
Epoch 35/200
5/5 [==============================] - 0s 11ms/step - loss: 0.9520 - val_loss:
0.6851
Epoch 36/200
5/5 [==============================] - 0s 10ms/step - loss: 0.8948 - val_loss:
0.6476
Epoch 37/200
5/5 [==============================] - 0s 10ms/step - loss: 0.8318 - val_loss:
0.6651
Epoch 38/200
5/5 [==============================] - 0s 10ms/step - loss: 0.8344 - val_loss:
0.6046
Epoch 39/200
5/5 [==============================] - 0s 11ms/step - loss: 0.7577 - val_loss:
0.5682
Epoch 40/200
5/5 [==============================] - 0s 11ms/step - loss: 0.7117 - val_loss:
0.5696
Epoch 41/200
5/5 [==============================] - 0s 11ms/step - loss: 0.7941 - val_loss:
0.5435
Epoch 42/200
5/5 [==============================] - 0s 11ms/step - loss: 0.6970 - val_loss:
0.5194
Epoch 43/200
5/5 [==============================] - 0s 15ms/step - loss: 0.6776 - val_loss:
0.5148
Epoch 44/200
5/5 [==============================] - 0s 11ms/step - loss: 0.6314 - val_loss:
0.4712
Epoch 45/200
5/5 [==============================] - 0s 10ms/step - loss: 0.6236 - val_loss:
0.4617
Epoch 46/200
5/5 [==============================] - 0s 10ms/step - loss: 0.5975 - val_loss:
0.4343
Epoch 47/200
5/5 [==============================] - 0s 10ms/step - loss: 0.5949 - val_loss:
0.4337
Epoch 48/200
5/5 [==============================] - 0s 10ms/step - loss: 0.6142 - val_loss:
0.4018
Epoch 49/200
5/5 [==============================] - 0s 10ms/step - loss: 0.5437 - val_loss:
```

```
0.3999
Epoch 50/200
5/5 [==============================] - 0s 10ms/step - loss: 0.5337 - val_loss:
0.3930
Epoch 51/200
5/5 [==============================] - 0s 10ms/step - loss: 0.4988 - val_loss:
0.3603
Epoch 52/200
5/5 [==============================] - 0s 10ms/step - loss: 0.4757 - val_loss:
0.3506
Epoch 53/200
5/5 [==============================] - 0s 10ms/step - loss: 0.4555 - val_loss:
0.3491
Epoch 54/200
5/5 [==============================] - 0s 10ms/step - loss: 0.4563 - val_loss:
0.3355
Epoch 55/200
5/5 [==============================] - 0s 10ms/step - loss: 0.4564 - val_loss:
0.3337
Epoch 56/200
5/5 [==============================] - 0s 10ms/step - loss: 0.4394 - val_loss:
0.3018
Epoch 57/200
5/5 [==============================] - 0s 10ms/step - loss: 0.3933 - val_loss:
0.3103
Epoch 58/200
5/5 [==============================] - 0s 10ms/step - loss: 0.3692 - val_loss:
0.2813
Epoch 59/200
5/5 [==============================] - 0s 10ms/step - loss: 0.3932 - val_loss:
0.2785
Epoch 60/200
5/5 [==============================] - 0s 11ms/step - loss: 0.3744 - val_loss:
0.2584
Epoch 61/200
5/5 [==============================] - 0s 10ms/step - loss: 0.3502 - val_loss:
0.2437
Epoch 62/200
5/5 [==============================] - 0s 10ms/step - loss: 0.3352 - val_loss:
0.2354
Epoch 63/200
5/5 [==============================] - 0s 10ms/step - loss: 0.3180 - val_loss:
0.2287
Epoch 64/200
5/5 [==============================] - 0s 10ms/step - loss: 0.2916 - val_loss:
0.2368
Epoch 65/200
5/5 [==============================] - 0s 10ms/step - loss: 0.2924 - val_loss:
```

```
0.2053
Epoch 66/200
5/5 [==============================] - 0s 10ms/step - loss: 0.3030 - val_loss:
0.2033
Epoch 67/200
5/5 [==============================] - 0s 11ms/step - loss: 0.2825 - val_loss:
0.2077
Epoch 68/200
5/5 [==============================] - 0s 10ms/step - loss: 0.2498 - val_loss:
0.1984
Epoch 69/200
5/5 [==============================] - 0s 10ms/step - loss: 0.2528 - val_loss:
0.1786
Epoch 70/200
5/5 [==============================] - 0s 10ms/step - loss: 0.2405 - val_loss:
0.1805
Epoch 71/200
5/5 [==============================] - 0s 10ms/step - loss: 0.2302 - val_loss:
0.1584
Epoch 72/200
5/5 [==============================] - 0s 10ms/step - loss: 0.2281 - val_loss:
0.1944
Epoch 73/200
5/5 [==============================] - 0s 10ms/step - loss: 0.2193 - val_loss:
0.1467
Epoch 74/200
5/5 [==============================] - 0s 11ms/step - loss: 0.1928 - val_loss:
0.1646
Epoch 75/200
5/5 [==============================] - 0s 10ms/step - loss: 0.2003 - val_loss:
0.1457
Epoch 76/200
5/5 [==============================] - 0s 10ms/step - loss: 0.1716 - val_loss:
0.1537
Epoch 77/200
5/5 [==============================] - 0s 10ms/step - loss: 0.1856 - val_loss:
0.1298
Epoch 78/200
5/5 [==============================] - 0s 10ms/step - loss: 0.1825 - val_loss:
0.1547
Epoch 79/200
5/5 [==============================] - 0s 11ms/step - loss: 0.1648 - val_loss:
0.1219
Epoch 80/200
5/5 [==============================] - 0s 11ms/step - loss: 0.1630 - val_loss:
0.1341
Epoch 81/200
5/5 [==============================] - 0s 11ms/step - loss: 0.1533 - val_loss:
```

```
0.1228
Epoch 82/200
5/5 [==============================] - 0s 11ms/step - loss: 0.1419 - val_loss:
0.1148
Epoch 83/200
5/5 [==============================] - 0s 11ms/step - loss: 0.1491 - val_loss:
0.1302
Epoch 84/200
5/5 [==============================] - 0s 10ms/step - loss: 0.1275 - val_loss:
0.1090
Epoch 85/200
5/5 [==============================] - 0s 10ms/step - loss: 0.1254 - val_loss:
0.1131
Epoch 86/200
5/5 [==============================] - 0s 10ms/step - loss: 0.1160 - val_loss:
0.1068
Epoch 87/200
5/5 [==============================] - 0s 10ms/step - loss: 0.1093 - val_loss:
0.1025
Epoch 88/200
5/5 [==============================] - 0s 15ms/step - loss: 0.1089 - val_loss:
0.0914
Epoch 89/200
5/5 [==============================] - 0s 12ms/step - loss: 0.1059 - val_loss:
0.0984
Epoch 90/200
5/5 [==============================] - 0s 11ms/step - loss: 0.1040 - val_loss:
0.0941
Epoch 91/200
5/5 [==============================] - 0s 15ms/step - loss: 0.0969 - val_loss:
0.0893
Epoch 92/200
5/5 [==============================] - 0s 11ms/step - loss: 0.1043 - val_loss:
0.0901
Epoch 93/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0956 - val_loss:
0.0766
Epoch 94/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0811 - val_loss:
0.0807
Epoch 95/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0909 - val_loss:
0.0781
Epoch 96/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0904 - val_loss:
0.0721
Epoch 97/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0807 - val_loss:
```

```
0.0965
Epoch 98/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0729 - val_loss:
0.0681
Epoch 99/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0719 - val_loss:
0.0709
Epoch 100/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0683 - val_loss:
0.0700
Epoch 101/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0668 - val_loss:
0.0716
Epoch 102/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0638 - val_loss:
0.0628
Epoch 103/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0586 - val_loss:
0.0568
Epoch 104/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0591 - val_loss:
0.0629
Epoch 105/200
5/5 [==============================] - 0s 9ms/step - loss: 0.0566 - val_loss:
0.0585
Epoch 106/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0470 - val_loss:
0.0559
Epoch 107/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0485 - val_loss:
0.0633
Epoch 108/200
5/5 [==============================] - 0s 15ms/step - loss: 0.0471 - val_loss:
0.0538
Epoch 109/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0460 - val_loss:
0.0497
Epoch 110/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0415 - val_loss:
0.0466
Epoch 111/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0422 - val_loss:
0.0498
Epoch 112/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0400 - val_loss:
0.0446
Epoch 113/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0391 - val_loss:
```

```
0.0481
Epoch 114/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0372 - val_loss:
0.0484
Epoch 115/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0345 - val_loss:
0.0383
Epoch 116/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0330 - val_loss:
0.0463
Epoch 117/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0337 - val_loss:
0.0450
Epoch 118/200
5/5 [==============================] - 0s 13ms/step - loss: 0.0316 - val_loss:
0.0383
Epoch 119/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0300 - val_loss:
0.0365
Epoch 120/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0277 - val_loss:
0.0418
Epoch 121/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0310 - val_loss:
0.0371
Epoch 122/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0287 - val_loss:
0.0356
Epoch 123/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0247 - val_loss:
0.0346
Epoch 124/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0275 - val_loss:
0.0351
Epoch 125/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0260 - val_loss:
0.0308
Epoch 126/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0229 - val_loss:
0.0359
Epoch 127/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0235 - val_loss:
0.0379
Epoch 128/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0218 - val_loss:
0.0281
Epoch 129/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0213 - val_loss:
```

```
0.0279
Epoch 130/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0204 - val_loss:
0.0302
Epoch 131/200
5/5 [==============================] - 0s 14ms/step - loss: 0.0199 - val_loss:
0.0308
Epoch 132/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0192 - val_loss:
0.0261
Epoch 133/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0172 - val_loss:
0.0282
Epoch 134/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0174 - val_loss:
0.0248
Epoch 135/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0174 - val_loss:
0.0272
Epoch 136/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0156 - val_loss:
0.0246
Epoch 137/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0149 - val_loss:
0.0272
Epoch 138/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0161 - val_loss:
0.0231
Epoch 139/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0156 - val_loss:
0.0221
Epoch 140/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0152 - val_loss:
0.0278
Epoch 141/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0148 - val_loss:
0.0239
Epoch 142/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0139 - val_loss:
0.0224
Epoch 143/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0132 - val_loss:
0.0221
Epoch 144/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0128 - val_loss:
0.0219
Epoch 145/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0126 - val_loss:
```

```
0.0229
Epoch 146/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0120 - val_loss:
0.0224
Epoch 147/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0113 - val_loss:
0.0214
Epoch 148/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0115 - val_loss:
0.0206
Epoch 149/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0126 - val_loss:
0.0218
Epoch 150/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0114 - val_loss:
0.0210
Epoch 151/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0101 - val_loss:
0.0197
Epoch 152/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0106 - val_loss:
0.0182
Epoch 153/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0097 - val_loss:
0.0203
Epoch 154/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0103 - val_loss:
0.0186
Epoch 155/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0101 - val_loss:
0.0183
Epoch 156/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0102 - val_loss:
0.0173
Epoch 157/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0104 - val_loss:
0.0176
Epoch 158/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0094 - val_loss:
0.0177
Epoch 159/200
5/5 [==============================] - 0s 15ms/step - loss: 0.0097 - val_loss:
0.0172
Epoch 160/200
5/5 [==============================] - 0s 13ms/step - loss: 0.0093 - val_loss:
0.0170
Epoch 161/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0099 - val_loss:
```

```
0.0175
Epoch 162/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0094 - val_loss:
0.0178
Epoch 163/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0088 - val_loss:
0.0172
Epoch 164/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0088 - val_loss:
0.0175
Epoch 165/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0081 - val_loss:
0.0172
Epoch 166/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0085 - val_loss:
0.0158
Epoch 167/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0092 - val_loss:
0.0167
Epoch 168/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0104 - val_loss:
0.0159
Epoch 169/200
5/5 [==============================] - 0s 9ms/step - loss: 0.0098 - val_loss:
0.0176
Epoch 170/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0098 - val_loss:
0.0185
Epoch 171/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0088 - val_loss:
0.0168
Epoch 172/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0080 - val_loss:
0.0148
Epoch 173/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0083 - val_loss:
0.0150
Epoch 174/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0087 - val_loss:
0.0176
Epoch 175/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0081 - val_loss:
0.0160
Epoch 176/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0089 - val_loss:
0.0145
Epoch 177/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0073 - val_loss:
```

```
0.0144
Epoch 178/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0081 - val_loss:
0.0146
Epoch 179/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0072 - val_loss:
0.0153
Epoch 180/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0071 - val_loss:
0.0153
Epoch 181/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0081 - val_loss:
0.0144
Epoch 182/200
5/5 [==============================] - 0s 18ms/step - loss: 0.0070 - val_loss:
0.0148
Epoch 183/200
5/5 [==============================] - 0s 15ms/step - loss: 0.0081 - val_loss:
0.0143
Epoch 184/200
5/5 [==============================] - 0s 13ms/step - loss: 0.0080 - val_loss:
0.0144
Epoch 185/200
5/5 [==============================] - 0s 15ms/step - loss: 0.0076 - val_loss:
0.0149
Epoch 186/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0071 - val_loss:
0.0157
Epoch 187/200
5/5 [==============================] - 0s 13ms/step - loss: 0.0079 - val_loss:
0.0133
Epoch 188/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0074 - val_loss:
0.0131
Epoch 189/200
5/5 [==============================] - 0s 15ms/step - loss: 0.0072 - val_loss:
0.0136
Epoch 190/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0070 - val_loss:
0.0142
Epoch 191/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0069 - val_loss:
0.0134
Epoch 192/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0068 - val_loss:
0.0136
Epoch 193/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0072 - val_loss:
```

```
0.0134
Epoch 194/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0072 - val_loss:
0.0135
Epoch 195/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0071 - val_loss:
0.0140
Epoch 196/200
5/5 [==============================] - 0s 13ms/step - loss: 0.0071 - val_loss:
0.0130
Epoch 197/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0069 - val_loss:
0.0130
Epoch 198/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0069 - val_loss:
0.0130
Epoch 199/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0067 - val_loss:
0.0130
Epoch 200/200
5/5 [==============================] - 0s 15ms/step - loss: 0.0064 - val_loss:
0.0131
Epoch 1/200
5/5 [==============================] - 1s 48ms/step - loss: 0.0104 - val_loss:
0.0196
Epoch 2/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0087 - val_loss:
0.0125
Epoch 3/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0077 - val_loss:
0.0147
Epoch 4/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0066 - val_loss:
0.0105
Epoch 5/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0064 - val_loss:
0.0115
Epoch 6/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0069 - val_loss:
0.0101
Epoch 7/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0068 - val_loss:
0.0101
Epoch 8/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0061 - val_loss:
0.0101
Epoch 9/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0059 - val_loss:
```

```
0.0097
Epoch 10/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0064 - val_loss:
0.0099
Epoch 11/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0062 - val_loss:
0.0095
Epoch 12/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0063 - val_loss:
0.0098
Epoch 13/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0059 - val_loss:
0.0095
Epoch 14/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0056 - val_loss:
0.0100
Epoch 15/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0063 - val_loss:
0.0091
Epoch 16/200
5/5 [==============================] - 0s 14ms/step - loss: 0.0059 - val_loss:
0.0095
Epoch 17/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0055 - val_loss:
0.0089
Epoch 18/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0059 - val_loss:
0.0089
Epoch 19/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0060 - val_loss:
0.0087
Epoch 20/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0053 - val_loss:
0.0085
Epoch 21/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0057 - val_loss:
0.0089
Epoch 22/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0054 - val_loss:
0.0084
Epoch 23/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0056 - val_loss:
0.0093
Epoch 24/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0055 - val_loss:
0.0085
Epoch 25/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0052 - val_loss:
```

```
0.0090
Epoch 26/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0052 - val_loss:
0.0088
Epoch 27/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0056 - val_loss:
0.0087
Epoch 28/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0055 - val_loss:
0.0086
Epoch 29/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0049 - val_loss:
0.0083
Epoch 30/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0051 - val_loss:
0.0087
Epoch 31/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0051 - val_loss:
0.0079
Epoch 32/200
5/5 [==============================] - 0s 14ms/step - loss: 0.0053 - val_loss:
0.0077
Epoch 33/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0056 - val_loss:
0.0077
Epoch 34/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0055 - val_loss:
0.0076
Epoch 35/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0054 - val_loss:
0.0074
Epoch 36/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0048 - val_loss:
0.0075
Epoch 37/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0049 - val_loss:
0.0076
Epoch 38/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0050 - val_loss:
0.0073
Epoch 39/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0055 - val_loss:
0.0076
Epoch 40/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0048 - val_loss:
0.0081
Epoch 41/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0050 - val_loss:
```

```
0.0074
Epoch 42/200
5/5 [==============================] - 0s 15ms/step - loss: 0.0050 - val_loss:
0.0074
Epoch 43/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0050 - val_loss:
0.0075
Epoch 44/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0047 - val_loss:
0.0072
Epoch 45/200
5/5 [==============================] - 0s 14ms/step - loss: 0.0048 - val_loss:
0.0076
Epoch 46/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0053 - val_loss:
0.0072
Epoch 47/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0046 - val_loss:
0.0070
Epoch 48/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0051 - val_loss:
0.0072
Epoch 49/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0049 - val_loss:
0.0070
Epoch 50/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0050 - val_loss:
0.0066
Epoch 51/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0052 - val_loss:
0.0065
Epoch 52/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0055 - val_loss:
0.0068
Epoch 53/200
5/5 [==============================] - 0s 9ms/step - loss: 0.0049 - val_loss:
0.0067
Epoch 54/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0048 - val_loss:
0.0061
Epoch 55/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0046 - val_loss:
0.0062
Epoch 56/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0048 - val_loss:
0.0061
Epoch 57/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0043 - val_loss:
```

```
0.0059
Epoch 58/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0048 - val_loss:
0.0065
Epoch 59/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0051 - val_loss:
0.0063
Epoch 60/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0047 - val_loss:
0.0061
Epoch 61/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0047 - val_loss:
0.0063
Epoch 62/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0045 - val_loss:
0.0067
Epoch 63/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0044 - val_loss:
0.0061
Epoch 64/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0048 - val_loss:
0.0062
Epoch 65/200
5/5 [==============================] - 0s 9ms/step - loss: 0.0045 - val_loss:
0.0065
Epoch 66/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0044 - val_loss:
0.0064
Epoch 67/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0043 - val_loss:
0.0062
Epoch 68/200
5/5 [==============================] - 0s 15ms/step - loss: 0.0044 - val_loss:
0.0062
Epoch 69/200
5/5 [==============================] - 0s 15ms/step - loss: 0.0046 - val_loss:
0.0063
Epoch 70/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0043 - val_loss:
0.0062
Epoch 71/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0045 - val_loss:
0.0062
Epoch 72/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0042 - val_loss:
0.0063
Epoch 73/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0043 - val_loss:
```

```
0.0061
Epoch 74/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0045 - val_loss:
0.0069
Epoch 75/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0042 - val_loss:
0.0062
Epoch 76/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0047 - val_loss:
0.0061
Epoch 77/200
5/5 [==============================] - 0s 14ms/step - loss: 0.0045 - val_loss:
0.0063
Epoch 78/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0043 - val_loss:
0.0066
Epoch 79/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0046 - val_loss:
0.0061
Epoch 80/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0042 - val_loss:
0.0061
Epoch 81/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0042 - val_loss:
0.0062
Epoch 82/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0043 - val_loss:
0.0061
Epoch 83/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0043 - val_loss:
0.0060
Epoch 84/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0044 - val_loss:
0.0061
Epoch 85/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0045 - val_loss:
0.0061
Epoch 86/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0043 - val_loss:
0.0065
Epoch 87/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0041 - val_loss:
0.0060
Epoch 88/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0042 - val_loss:
0.0060
Epoch 89/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0042 - val_loss:
```

```
0.0060
Epoch 90/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0040 - val_loss:
0.0059
Epoch 91/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0043 - val_loss:
0.0057
Epoch 92/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0045 - val_loss:
0.0061
Epoch 93/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0043 - val_loss:
0.0060
Epoch 94/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0043 - val_loss:
0.0056
Epoch 95/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0042 - val_loss:
0.0053
Epoch 96/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0042 - val_loss:
0.0052
Epoch 97/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0044 - val_loss:
0.0057
Epoch 98/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0042 - val_loss:
0.0051
Epoch 99/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0042 - val_loss:
0.0053
Epoch 100/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0038 - val_loss:
0.0053
Epoch 101/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0042 - val_loss:
0.0052
Epoch 102/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0042 - val_loss:
0.0053
Epoch 103/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0044 - val_loss:
0.0055
Epoch 104/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0042 - val_loss:
0.0053
Epoch 105/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0041 - val_loss:
```

```
0.0052
Epoch 106/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0039 - val_loss:
0.0052
Epoch 107/200
5/5 [==============================] - 0s 14ms/step - loss: 0.0041 - val_loss:
0.0051
Epoch 108/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0041 - val_loss:
0.0050
Epoch 109/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0039 - val_loss:
0.0054
Epoch 110/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0039 - val_loss:
0.0052
Epoch 111/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0040 - val_loss:
0.0053
Epoch 112/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0042 - val_loss:
0.0052
Epoch 113/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0043 - val_loss:
0.0052
Epoch 114/200
5/5 [==============================] - 0s 14ms/step - loss: 0.0042 - val_loss:
0.0051
Epoch 115/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0042 - val_loss:
0.0050
Epoch 116/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0042 - val_loss:
0.0055
Epoch 117/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0041 - val_loss:
0.0052
Epoch 118/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0042 - val_loss:
0.0051
Epoch 119/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0038 - val_loss:
0.0054
Epoch 120/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0044 - val_loss:
0.0055
Epoch 121/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0040 - val_loss:
```

```
0.0053
Epoch 122/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0040 - val_loss:
0.0052
Epoch 123/200
5/5 [==============================] - 0s 12ms/step - loss: 0.0043 - val_loss:
0.0054
Epoch 124/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0044 - val_loss:
0.0053
Epoch 125/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0040 - val_loss:
0.0054
Epoch 126/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0041 - val_loss:
0.0051
Epoch 127/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0041 - val_loss:
0.0052
Epoch 128/200
5/5 [==============================] - 0s 15ms/step - loss: 0.0038 - val_loss:
0.0052
Epoch 129/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0040 - val_loss:
0.0051
Epoch 130/200
5/5 [==============================] - 0s 15ms/step - loss: 0.0038 - val_loss:
0.0052
Epoch 131/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0037 - val_loss:
0.0054
Epoch 132/200
5/5 [==============================] - 0s 15ms/step - loss: 0.0038 - val_loss:
0.0053
Epoch 133/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0039 - val_loss:
0.0053
Epoch 134/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0038 - val_loss:
0.0050
Epoch 135/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0037 - val_loss:
0.0048
Epoch 136/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0039 - val_loss:
0.0049
Epoch 137/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0039 - val_loss:
```

```
0.0047
Epoch 138/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0040 - val_loss:
0.0046
Epoch 139/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0042 - val_loss:
0.0048
Epoch 140/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0039 - val_loss:
0.0055
Epoch 141/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0045 - val_loss:
0.0049
Epoch 142/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0039 - val_loss:
0.0046
Epoch 143/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0039 - val_loss:
0.0046
Epoch 144/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0041 - val_loss:
0.0052
Epoch 145/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0042 - val_loss:
0.0051
Epoch 146/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0040 - val_loss:
0.0043
Epoch 147/200
5/5 [==============================] - 0s 12ms/step - loss: 0.0041 - val_loss:
0.0045
Epoch 148/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0041 - val_loss:
0.0046
Epoch 149/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0041 - val_loss:
0.0047
Epoch 150/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0041 - val_loss:
0.0044
Epoch 151/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0038 - val_loss:
0.0045
Epoch 152/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0039 - val_loss:
0.0046
Epoch 153/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0039 - val_loss:
```

```
0.0044
Epoch 154/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0038 - val_loss:
0.0044
Epoch 155/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0039 - val_loss:
0.0043
Epoch 156/200
5/5 [==============================] - 0s 12ms/step - loss: 0.0037 - val_loss:
0.0044
Epoch 157/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0039 - val_loss:
0.0043
Epoch 158/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0038 - val_loss:
0.0043
Epoch 159/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0040 - val_loss:
0.0043
Epoch 160/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0041 - val_loss:
0.0044
Epoch 161/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0041 - val_loss:
0.0042
Epoch 162/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0038 - val_loss:
0.0045
Epoch 163/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0039 - val_loss:
0.0045
Epoch 164/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0041 - val_loss:
0.0048
Epoch 165/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0039 - val_loss:
0.0043
Epoch 166/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0037 - val_loss:
0.0044
Epoch 167/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0040 - val_loss:
0.0043
Epoch 168/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0040 - val_loss:
0.0046
Epoch 169/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0038 - val_loss:
```

```
0.0042
Epoch 170/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0038 - val_loss:
0.0042
Epoch 171/200
5/5 [==============================] - 0s 15ms/step - loss: 0.0038 - val_loss:
0.0042
Epoch 172/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0040 - val_loss:
0.0043
Epoch 173/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0038 - val_loss:
0.0045
Epoch 174/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0040 - val_loss:
0.0045
Epoch 175/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0037 - val_loss:
0.0044
Epoch 176/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0036 - val_loss:
0.0041
Epoch 177/200
5/5 [==============================] - 0s 15ms/step - loss: 0.0038 - val_loss:
0.0040
Epoch 178/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0037 - val_loss:
0.0042
Epoch 179/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0038 - val_loss:
0.0048
Epoch 180/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0037 - val_loss:
0.0043
Epoch 181/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0037 - val_loss:
0.0043
Epoch 182/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0036 - val_loss:
0.0044
Epoch 183/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0035 - val_loss:
0.0045
Epoch 184/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0037 - val_loss:
0.0041
Epoch 185/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0035 - val_loss:
```

```
0.0041
Epoch 186/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0037 - val_loss:
0.0044
Epoch 187/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0035 - val_loss:
0.0042
Epoch 188/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0038 - val_loss:
0.0042
Epoch 189/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0037 - val_loss:
0.0042
Epoch 190/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0038 - val_loss:
0.0049
Epoch 191/200
5/5 [==============================] - 0s 15ms/step - loss: 0.0040 - val_loss:
0.0046
Epoch 192/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0036 - val_loss:
0.0042
Epoch 193/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0039 - val_loss:
0.0042
Epoch 194/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0036 - val_loss:
0.0045
Epoch 195/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0037 - val_loss:
0.0042
Epoch 196/200
5/5 [==============================] - 0s 12ms/step - loss: 0.0035 - val_loss:
0.0041
Epoch 197/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0035 - val_loss:
0.0042
Epoch 198/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0034 - val_loss:
0.0042
Epoch 199/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0037 - val_loss:
0.0042
Epoch 200/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0035 - val_loss:
0.0042
Epoch 1/200
5/5 [==============================] - 1s 47ms/step - loss: 0.0043 - val_loss:
```

```
0.0040
Epoch 2/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0037 - val_loss:
0.0044
Epoch 3/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0035 - val_loss:
0.0039
Epoch 4/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0034 - val_loss:
0.0042
Epoch 5/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0034 - val_loss:
0.0039
Epoch 6/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0035 - val_loss:
0.0041
Epoch 7/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0036 - val_loss:
0.0039
Epoch 8/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0034 - val_loss:
0.0041
Epoch 9/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0034 - val_loss:
0.0038
Epoch 10/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0034 - val_loss:
0.0040
Epoch 11/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0034 - val_loss:
0.0039
Epoch 12/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0036 - val_loss:
0.0040
Epoch 13/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0037 - val_loss:
0.0040
Epoch 14/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0035 - val_loss:
0.0041
Epoch 15/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0037 - val_loss:
0.0039
Epoch 16/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0034 - val_loss:
0.0043
Epoch 17/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0034 - val_loss:
```

```
0.0039
Epoch 18/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0035 - val_loss:
0.0041
Epoch 19/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0034 - val_loss:
0.0040
Epoch 20/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0038 - val_loss:
0.0039
Epoch 21/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0035 - val_loss:
0.0042
Epoch 22/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0033 - val_loss:
0.0039
Epoch 23/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0034 - val_loss:
0.0041
Epoch 24/200
5/5 [==============================] - 0s 14ms/step - loss: 0.0034 - val_loss:
0.0041
Epoch 25/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0035 - val_loss:
0.0039
Epoch 26/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0034 - val_loss:
0.0039
Epoch 27/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0034 - val_loss:
0.0039
Epoch 28/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0035 - val_loss:
0.0040
Epoch 29/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0034 - val_loss:
0.0040
Epoch 30/200
5/5 [==============================] - 0s 14ms/step - loss: 0.0035 - val_loss:
0.0039
Epoch 31/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0035 - val_loss:
0.0039
Epoch 32/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0033 - val_loss:
0.0042
Epoch 33/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0038 - val_loss:
```

```
0.0040
Epoch 34/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0034 - val_loss:
0.0046
Epoch 35/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0035 - val_loss:
0.0039
Epoch 36/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0036 - val_loss:
0.0039
Epoch 37/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0034 - val_loss:
0.0043
Epoch 38/200
5/5 [==============================] - 0s 15ms/step - loss: 0.0037 - val_loss:
0.0039
Epoch 39/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0034 - val_loss:
0.0040
Epoch 40/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0035 - val_loss:
0.0041
Epoch 41/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0034 - val_loss:
0.0038
Epoch 42/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0036 - val_loss:
0.0042
Epoch 43/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0035 - val_loss:
0.0038
Epoch 44/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0035 - val_loss:
0.0039
Epoch 45/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0035 - val_loss:
0.0039
Epoch 46/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0034 - val_loss:
0.0038
Epoch 47/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0037 - val_loss:
0.0042
Epoch 48/200
5/5 [==============================] - 0s 15ms/step - loss: 0.0036 - val_loss:
0.0038
Epoch 49/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0033 - val_loss:
```

```
0.0038
Epoch 50/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0032 - val_loss:
0.0038
Epoch 51/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0033 - val_loss:
0.0040
Epoch 52/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0035 - val_loss:
0.0038
Epoch 53/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0034 - val_loss:
0.0039
Epoch 54/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0032 - val_loss:
0.0038
Epoch 55/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0035 - val_loss:
0.0039
Epoch 56/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0032 - val_loss:
0.0038
Epoch 57/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0036 - val_loss:
0.0038
Epoch 58/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0033 - val_loss:
0.0041
Epoch 59/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0033 - val_loss:
0.0040
Epoch 60/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0036 - val_loss:
0.0038
Epoch 61/200
5/5 [==============================] - 0s 14ms/step - loss: 0.0035 - val_loss:
0.0039
Epoch 62/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0034 - val_loss:
0.0042
Epoch 63/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0035 - val_loss:
0.0038
Epoch 64/200
5/5 [==============================] - 0s 14ms/step - loss: 0.0032 - val_loss:
0.0039
Epoch 65/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0036 - val_loss:
```

```
0.0044
Epoch 66/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0036 - val_loss:
0.0038
Epoch 67/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0036 - val_loss:
0.0042
Epoch 68/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0035 - val_loss:
0.0038
Epoch 69/200
5/5 [==============================] - 0s 15ms/step - loss: 0.0036 - val_loss:
0.0039
Epoch 70/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0033 - val_loss:
0.0038
Epoch 71/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0037 - val_loss:
0.0038
Epoch 72/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0033 - val_loss:
0.0040
Epoch 73/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0033 - val_loss:
0.0038
Epoch 74/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0033 - val_loss:
0.0039
Epoch 75/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0035 - val_loss:
0.0039
Epoch 76/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0034 - val_loss:
0.0038
Epoch 77/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0036 - val_loss:
0.0040
Epoch 78/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0033 - val_loss:
0.0037
Epoch 79/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0034 - val_loss:
0.0038
Epoch 80/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0034 - val_loss:
0.0040
Epoch 81/200
5/5 [==============================] - 0s 14ms/step - loss: 0.0035 - val_loss:
```

```
0.0038
Epoch 82/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0032 - val_loss:
0.0038
Epoch 83/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0034 - val_loss:
0.0038
Epoch 84/200
5/5 [==============================] - 0s 9ms/step - loss: 0.0033 - val_loss:
0.0039
Epoch 85/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0031 - val_loss:
0.0039
Epoch 86/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0033 - val_loss:
0.0038
Epoch 87/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0033 - val_loss:
0.0040
Epoch 88/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0033 - val_loss:
0.0038
Epoch 89/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0034 - val_loss:
0.0038
Epoch 90/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0034 - val_loss:
0.0041
Epoch 91/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0033 - val_loss:
0.0038
Epoch 92/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0034 - val_loss:
0.0038
Epoch 93/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0035 - val_loss:
0.0040
Epoch 94/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0036 - val_loss:
0.0039
Epoch 95/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0034 - val_loss:
0.0039
Epoch 96/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0032 - val_loss:
0.0039
Epoch 97/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0033 - val_loss:
```

```
0.0039
Epoch 98/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0031 - val_loss:
0.0039
Epoch 99/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0033 - val_loss:
0.0039
Epoch 100/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0030 - val_loss:
0.0042
Epoch 101/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0032 - val_loss:
0.0038
Epoch 102/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0032 - val_loss:
0.0041
Epoch 103/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0033 - val_loss:
0.0039
Epoch 104/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0032 - val_loss:
0.0039
Epoch 105/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0035 - val_loss:
0.0040
Epoch 106/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0033 - val_loss:
0.0039
Epoch 107/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0032 - val_loss:
0.0040
Epoch 108/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0034 - val_loss:
0.0039
Epoch 109/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0033 - val_loss:
0.0039
Epoch 110/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0032 - val_loss:
0.0040
Epoch 111/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0034 - val_loss:
0.0038
Epoch 112/200
5/5 [==============================] - 0s 15ms/step - loss: 0.0033 - val_loss:
0.0039
Epoch 113/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0032 - val_loss:
```

```
0.0040
Epoch 114/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0032 - val_loss:
0.0038
Epoch 115/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0033 - val_loss:
0.0037
Epoch 116/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0032 - val_loss:
0.0040
Epoch 117/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0033 - val_loss:
0.0038
Epoch 118/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0033 - val_loss:
0.0038
Epoch 119/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0032 - val_loss:
0.0039
Epoch 120/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0033 - val_loss:
0.0039
Epoch 121/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0032 - val_loss:
0.0037
Epoch 122/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0032 - val_loss:
0.0037
Epoch 123/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0031 - val_loss:
0.0039
Epoch 124/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0033 - val_loss:
0.0037
Epoch 125/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0033 - val_loss:
0.0037
Epoch 126/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0034 - val_loss:
0.0042
Epoch 127/200
5/5 [==============================] - 0s 15ms/step - loss: 0.0035 - val_loss:
0.0037
Epoch 128/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0032 - val_loss:
0.0037
Epoch 129/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0033 - val_loss:
```

```
0.0039
Epoch 130/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0033 - val_loss:
0.0037
Epoch 131/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0033 - val_loss:
0.0040
Epoch 132/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0030 - val_loss:
0.0037
Epoch 133/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0031 - val_loss:
0.0039
Epoch 134/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0031 - val_loss:
0.0038
Epoch 135/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0032 - val_loss:
0.0037
Epoch 136/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0032 - val_loss:
0.0038
Epoch 137/200
5/5 [==============================] - 0s 12ms/step - loss: 0.0031 - val_loss:
0.0040
Epoch 138/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0032 - val_loss:
0.0037
Epoch 139/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0031 - val_loss:
0.0038
Epoch 140/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0031 - val_loss:
0.0038
Epoch 141/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0032 - val_loss:
0.0037
Epoch 142/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0034 - val_loss:
0.0037
Epoch 143/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0033 - val_loss:
0.0040
Epoch 144/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0032 - val_loss:
0.0038
Epoch 145/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0033 - val_loss:
```

```
0.0038
Epoch 146/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0031 - val_loss:
0.0038
Epoch 147/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0034 - val_loss:
0.0037
Epoch 148/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0033 - val_loss:
0.0040
Epoch 149/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0032 - val_loss:
0.0037
Epoch 150/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0030 - val_loss:
0.0038
Epoch 151/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0033 - val_loss:
0.0038
Epoch 152/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0032 - val_loss:
0.0037
Epoch 153/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0032 - val_loss:
0.0038
Epoch 154/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0033 - val_loss:
0.0039
Epoch 155/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0031 - val_loss:
0.0038
Epoch 156/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0033 - val_loss:
0.0038
Epoch 157/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0032 - val_loss:
0.0037
Epoch 158/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0032 - val_loss:
0.0038
Epoch 159/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0031 - val_loss:
0.0037
Epoch 160/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0032 - val_loss:
0.0039
Epoch 161/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0033 - val_loss:
```

```
0.0040
Epoch 162/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0032 - val_loss:
0.0038
Epoch 163/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0032 - val_loss:
0.0037
Epoch 164/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0031 - val_loss:
0.0038
Epoch 165/200
5/5 [==============================] - 0s 14ms/step - loss: 0.0030 - val_loss:
0.0038
Epoch 166/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0034 - val_loss:
0.0038
Epoch 167/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0032 - val_loss:
0.0037
Epoch 168/200
5/5 [==============================] - 0s 13ms/step - loss: 0.0031 - val_loss:
0.0039
Epoch 169/200
5/5 [==============================] - 0s 14ms/step - loss: 0.0031 - val_loss:
0.0039
Epoch 170/200
5/5 [==============================] - 0s 13ms/step - loss: 0.0032 - val_loss:
0.0037
Epoch 171/200
5/5 [==============================] - 0s 14ms/step - loss: 0.0032 - val_loss:
0.0040
Epoch 172/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0032 - val_loss:
0.0039
Epoch 173/200
5/5 [==============================] - 0s 12ms/step - loss: 0.0033 - val_loss:
0.0037
Epoch 174/200
5/5 [==============================] - 0s 18ms/step - loss: 0.0033 - val_loss:
0.0038
Epoch 175/200
5/5 [==============================] - 0s 16ms/step - loss: 0.0031 - val_loss:
0.0038
Epoch 176/200
5/5 [==============================] - 0s 16ms/step - loss: 0.0032 - val_loss:
0.0039
Epoch 177/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0032 - val_loss:
```

```
0.0040
Epoch 178/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0031 - val_loss:
0.0038
Epoch 179/200
5/5 [==============================] - 0s 13ms/step - loss: 0.0032 - val_loss:
0.0037
Epoch 180/200
5/5 [==============================] - 0s 13ms/step - loss: 0.0033 - val_loss:
0.0039
Epoch 181/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0031 - val_loss:
0.0038
Epoch 182/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0031 - val_loss:
0.0037
Epoch 183/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0030 - val_loss:
0.0040
Epoch 184/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0033 - val_loss:
0.0037
Epoch 185/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0032 - val_loss:
0.0038
Epoch 186/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0033 - val_loss:
0.0041
Epoch 187/200
5/5 [==============================] - 0s 15ms/step - loss: 0.0030 - val_loss:
0.0037
Epoch 188/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0031 - val_loss:
0.0038
Epoch 189/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0033 - val_loss:
0.0038
Epoch 190/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0030 - val_loss:
0.0039
Epoch 191/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0030 - val_loss:
0.0039
Epoch 192/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0033 - val_loss:
0.0039
Epoch 193/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0033 - val_loss:
```

```
0.0037
Epoch 194/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0031 - val_loss:
0.0039
Epoch 195/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0030 - val_loss:
0.0039
Epoch 196/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0029 - val_loss:
0.0038
Epoch 197/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0029 - val_loss:
0.0038
Epoch 198/200
5/5 [==============================] - 0s 11ms/step - loss: 0.0032 - val_loss:
0.0040
Epoch 199/200
5/5 [==============================] - 0s 10ms/step - loss: 0.0032 - val_loss:
0.0038
Epoch 200/200
5/5 [==============================] - 0s 15ms/step - loss: 0.0031 - val_loss:
0.0038
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 15)]              0

 dense (Dense)               (None, 90)                1440

 dropout (Dropout)           (None, 90)                0

 dense_1 (Dense)             (None, 45)                4095

 dense_2 (Dense)             (None, 15)                690

 dense_3 (Dense)             (None, 5)                 80

 dense_4 (Dense)             (None, 30)                180

 dense_5 (Dense)             (None, 90)                2790

 dense_6 (Dense)             (None, 45)                4095

 dense_7 (Dense)             (None, 14)                644

 dense_8 (Dense)             (None, 1)                 15
```

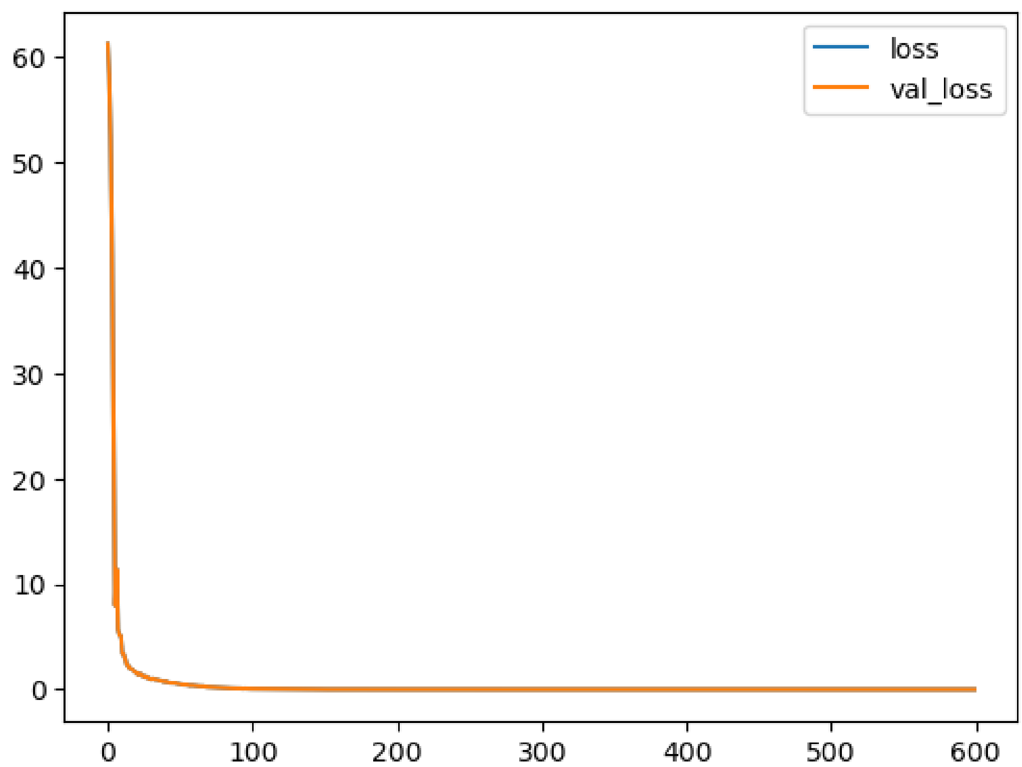
```
=================================================================
Total params: 14,029
Trainable params: 14,029
Non-trainable params: 0
_________________________________________________________________
```

[71]:
```
y_pred = model.predict(x_test)
print("MAE:", metrics.mean_absolute_error(y_test, y_pred))
print("MSE:", metrics.mean_squared_error(y_test, y_pred))
print("RMSE:", np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
print("VarScore:", metrics.explained_variance_score(y_test, y_pred))
```

```
10/10 [==============================] - 0s 2ms/step
MAE: 0.04323374840943305
MSE: 0.0038056033449608514
RMSE: 0.06168957241674521
VarScore: 0.8500695661576034
```

[72]:
```
flatten_result = [i for j in results for i in j]
```

[73]:
```
plt.plot(flatten_result, label="loss")
plt.plot(flatten_result, label="val_loss")
plt.legend()
```



6.1.1 Result:

Considering the results of ANN regression, it can be seen that we had a slight improvement in the accuracy of the model. It is good to mention here that we used explained variance score, which is almost equivalent to r-squared, however, technically, there are some differences between these metrics, but here we can use them interchangeably. Although there was a small improvement in the result compared to statsmodel multivariate regression, the ANN regression is resource-consuming instead, so basically, in my opinion, statsmodel is the winner for this case.