# Week11

November 10, 2024

# 1 STOR 320 Introduction to Data Science

## 1.1 Week 10: Cross validation

[ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.formula.api as smf
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

### 1.1.1 Helper Functions

[ ]:
```
def OSR2(y_train, y_test, y_pred):

    SSE = np.sum((y_test - y_pred)**2)
    SST = np.sum((y_test - np.mean(y_train))**2)

    return (1 - SSE/SST)
```

[ ]:
```
def MAE(y_test, y_pred):

    return (np.mean(abs(y_test - y_pred)))
```

[ ]:
```
def RMSE(y_test, y_pred):

    return np.sqrt(np.mean((y_test - y_pred)**2))
```

[ ]:
```
def print_metrics(model, X_train, y_train, X_test, y_test,␣
 ↪flag_log_sale_price=False):

    if (flag_log_sale_price == True):

        y_pred_train = pd.Series(model.predict(X_train)).reset_index(drop=True)
        y_pred_test = pd.Series(model.predict(X_test)).reset_index(drop=True)
        y_train = y_train.copy().reset_index(drop=True)
        y_test = y_test.copy().reset_index(drop=True)
```

```
        print("\nMetrics for Log(Sale Price):\n")

    elif (flag_log_sale_price == False):

        y_pred_train = pd.Series(model.predict(X_train)).apply(np.exp).
↪reset_index(drop=True)
        y_pred_test = pd.Series(model.predict(X_test)).apply(np.exp).
↪reset_index(drop=True)
        y_train = y_train.copy().apply(np.exp).reset_index(drop=True)
        y_test = y_test.copy().apply(np.exp).reset_index(drop=True)

        print("\nMetrics for Sale Price:\n")

    print('Training R2', OSR2(y_train, y_train, y_pred_train))
    print('Training MAE', MAE(y_train, y_pred_train))
    print('Training RMSE', RMSE(y_train, y_pred_train))

    print('Out-of-sample R2', OSR2(y_train, y_test, y_pred_test))
    print('Out-of-sample MAE', MAE(y_test, y_pred_test))
    print('Out-of-sample RMSE', RMSE(y_test, y_pred_test))

    return None
```

[ ]:

```
```

[ ]:

```
ames = pd.read_csv('cleaned_Ames.csv')
ames.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2765 entries, 0 to 2764
Columns: 105 entries, Unnamed: 0 to YearsSince1950GarageBuilt
dtypes: float64(21), int64(45), object(39)
memory usage: 2.2+ MB
```

[ ]:

```
ames = pd.read_feather('cleaned_ames.feather')
ames
```

[ ]:

```
      LogSalePrice MSSubClass MSZoning  LotFrontage  LotArea Street     Alley  \
0        12.278393         20       RL        141.0  31770.0   Pave  NoAccess
1        11.561716         20       RH         80.0  11622.0   Pave  NoAccess
2        12.055250         20       RL         81.0  14267.0   Pave  NoAccess
3        12.404924         20       RL         93.0  11160.0   Pave  NoAccess
4        12.154253         60       RL         74.0  13830.0   Pave  NoAccess
…              …        …        …          …        …      …       …
2924     11.782953         20       RL        160.0  20000.0   Pave  NoAccess
2925     11.867097         80       RL         37.0   7937.0   Pave  NoAccess
2926     11.782953         20       RL          0.0   8885.0   Pave  NoAccess
```

```
2928     12.043554         20       RL         77.0  10010.0   Pave  NoAccess
2929     12.144197         60       RL         74.0   9627.0   Pave  NoAccess

     LotShape LandContour Utilities  … PreCast Stone Stucco VinylSd WdSdng  \
0         IR1         Lvl    AllPub  …       0     0      0       0      0
1         Reg         Lvl    AllPub  …       0     0      0       1      0
2         IR1         Lvl    AllPub  …       0     0      0       0      1
3         Reg         Lvl    AllPub  …       0     0      0       0      0
4         IR1         Lvl    AllPub  …       0     0      0       1      0
…       …         …       …  …     …   …    …     …    …
2924      Reg         Lvl    AllPub  …       0     0      0       1      0
2925      IR1         Lvl    AllPub  …       0     0      0       0      0
2926      IR1         Low    AllPub  …       0     0      0       0      0
2928      Reg         Lvl    AllPub  …       0     0      0       0      0
2929      Reg         Lvl    AllPub  …       0     0      0       0      0

     WdShing WdShng YearsSince1950Built YearsSince1950Remod  \
0          0      0                  10                  10
1          0      0                  11                  11
2          0      0                   8                   8
3          0      0                  18                  18
4          0      0                  47                  48
…      …     …                   …                  …
2924       0      0                  10                  46
2925       0      0                  34                  34
2926       0      0                  33                  33
2928       0      0                  24                  25
2929       0      0                  43                  44

     YearsSince1950GarageBuilt
0                         10.0
1                         11.0
2                          8.0
3                         18.0
4                         47.0
…                          …
2924                      10.0
2925                      34.0
2926                      33.0
2928                      25.0
2929                      43.0

[2765 rows x 105 columns]
```

[ ]:
```
ames.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2765 entries, 0 to 2929
```

```
Columns: 105 entries, LogSalePrice to YearsSince1950GarageBuilt
dtypes: category(53), float64(21), int64(31)
memory usage: 1.3 MB
```

[ ]:
```
import warnings
warnings.filterwarnings('ignore')
```

[ ]:
```
ames_train = ames.loc[ames['YrSold'].isin([2006, 2007, 2008])]
ames_test = ames.loc[ames['YrSold'].isin([2009, 2010])]

ames = ames.drop(columns = ['YrSold'])
ames_train = ames_train.drop(columns = ['YrSold'])
ames_test = ames_test.drop(columns = ['YrSold'])

y_train = ames_train['LogSalePrice']
y_test = ames_test['LogSalePrice']

print(ames.shape, ames_train.shape, ames_test.shape)
```

```
(2765, 104) (1828, 104) (937, 104)
```

### 1.1.2 Simple Linear model with higher-order Variables

[ ]:
```
def create_polynomial_features(df, n_degree):

    new_df = None

    for i in range(2, n_degree+1):

        tmp = df.pow(i)

        affix = '_p'+str(i)
        tmp.columns = list(map(lambda x: x + affix, df.columns))

        if new_df is not None:
            new_df = pd.concat([new_df, tmp], axis=1)
        else:
            new_df = tmp

    return new_df
```

NOTE: An important consideration when creating higher-order variables is that the resulting features will tend to have some degree of linear dependence amongst themselves. This is normal as several new features are based on their zero-th power peer. Such correlation can also yield a high degree of multicollinearity in the regression models. The `sklearn` implementations that we will be using do not automatically account for this phenomenon, therefore we must be careful in selection the `n_degree`, and analyzing the model fit.

```
[ ]: # We only choose a select list of variables to do polynomial transformation.
poly_cols = ['LotFrontage', 'LotArea', 'MasVnrArea', 'BsmtFinSF1',␣
 ↪'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF',
             'X1stFlrSF', 'X2ndFlrSF', 'LowQualFinSF', 'GrLivArea',␣
 ↪'GarageArea', 'WoodDeckSF', 'OpenPorchSF',
             'EnclosedPorch', 'X3SsnPorch', 'ScreenPorch', 'MiscVal',␣
 ↪'YearsSince1950Built',
             'YearsSince1950Remod', 'YearsSince1950GarageBuilt']
```

```
[ ]: n_degree = 2

train_poly_temp = create_polynomial_features(ames_train[poly_cols], n_degree)
test_poly_temp = create_polynomial_features(ames_test[poly_cols], n_degree)

ames_train_poly = pd.concat([ames_train, train_poly_temp], axis=1)
ames_test_poly = pd.concat([ames_test, test_poly_temp], axis=1)

print(ames_train.shape, ames_test.shape)
print(train_poly_temp.shape, test_poly_temp.shape)
print(ames_train_poly.shape, ames_test_poly.shape)
```

```
(1828, 104) (937, 104)
(1828, 21) (937, 21)
(1828, 125) (937, 125)
```

```
[ ]: print(ames_train_poly.shape)
all_columns = "+".join(ames_train_poly.columns.difference(["LogSalePrice"]))
my_formula = "LogSalePrice~" + all_columns +'-1'
print(my_formula)

mod_naive_poly = smf.ols(my_formula, data=ames_train_poly)
nlr_poly = mod_naive_poly.fit()

print(nlr_poly.summary())
```

```
(1828, 125)
LogSalePrice~Alley+Artery+AsbShng+AsphShn+BedroomAbvGr+BldgType+BrkCmn+BrkComm+B
rkFace+BsmtCond+BsmtExposure+BsmtFinSF1+BsmtFinSF1_p2+BsmtFinSF2+BsmtFinSF2_p2+B
smtFinType1+BsmtFinType2+BsmtFullBath+BsmtHalfBath+BsmtQual+BsmtUnfSF+BsmtUnfSF_
p2+CBlock+CemntBd+CentralAir+CmentBd+Electrical+EnclosedPorch+EnclosedPorch_p2+E
xterCond+ExterQual+Feedr+Fence+FireplaceQu+Fireplaces+Foundation+FullBath+Functi
onal+GarageArea+GarageArea_p2+GarageCars+GarageCond+GarageFinish+GarageQual+Gara
geType+GrLivArea+GrLivArea_p2+HalfBath+HdBoard+Heating+HeatingQC+HouseStyle+ImSt
ucc+KitchenAbvGr+KitchenQual+LandContour+LandSlope+LotArea+LotArea_p2+LotConfig+
LotFrontage+LotFrontage_p2+LotShape+LowQualFinSF+LowQualFinSF_p2+MSSubClass+MSZo
ning+MasVnrArea+MasVnrArea_p2+MasVnrType+MetalSd+MiscFeature+MiscVal+MiscVal_p2+
MoSold+Neighborhood+Norm+OpenPorchSF+OpenPorchSF_p2+Other+OverallCond+OverallQua
l+PavedDrive+Plywood+PoolArea+PoolQC+PosA+PosN+PreCast+RRAe+RRAn+RRNe+RRNn+RoofM
```

```
atl+RoofStyle+SaleCondition+SaleType+ScreenPorch+ScreenPorch_p2+Stone+Street+Stu
cco+TotRmsAbvGrd+TotalBsmtSF+TotalBsmtSF_p2+Utilities+VinylSd+WdSdng+WdShing+WdS
hng+WoodDeckSF+WoodDeckSF_p2+X1stFlrSF+X1stFlrSF_p2+X2ndFlrSF+X2ndFlrSF_p2+X3Ssn
Porch+X3SsnPorch_p2+YearsSince1950Built+YearsSince1950Built_p2+YearsSince1950Gar
ageBuilt+YearsSince1950GarageBuilt_p2+YearsSince1950Remod+YearsSince1950Remod_p2
-1

                            OLS Regression Results
==============================================================================
Dep. Variable:           LogSalePrice   R-squared:                       0.962
Model:                            OLS   Adj. R-squared:                  0.954
Method:                 Least Squares   F-statistic:                     118.8
Date:                Wed, 06 Nov 2024   Prob (F-statistic):               0.00
Time:                        18:17:35   Log-Likelihood:                 2140.1
No. Observations:                1828   AIC:                            -3630.
Df Residuals:                    1503   BIC:                            -1839.
Df Model:                         324
Covariance Type:            nonrobust
================================================================================
================
                                  coef    std err          t      P>|t|
[0.025      0.975]
--------------------------------------------------------------------------------
----------------
Alley[Grvl]                     4.9635      0.140     35.451      0.000
4.689       5.238
Alley[NoAccess]                 4.9859      0.139     35.754      0.000
4.712       5.259
Alley[Pave]                     4.9939      0.140     35.609      0.000
4.719       5.269
BedroomAbvGr[T.1]              -0.0107      0.074     -0.145      0.885
-0.155       0.134
BedroomAbvGr[T.2]              -0.0093      0.074     -0.126      0.900
-0.154       0.136
BedroomAbvGr[T.3]              -0.0123      0.074     -0.165      0.869
-0.158       0.133
BedroomAbvGr[T.4]              -0.0144      0.075     -0.193      0.847
-0.161       0.132
BedroomAbvGr[T.5]              -0.0677      0.077     -0.880      0.379
-0.218       0.083
BedroomAbvGr[T.6]               0.0342      0.088      0.389      0.697
-0.138       0.206
BldgType[T.2fmCon]             -0.1128      0.106     -1.061      0.289
-0.321       0.096
BldgType[T.Duplex]             -0.0287      0.017     -1.726      0.085
-0.061       0.004
BldgType[T.Twnhs]              -0.0321      0.042     -0.764      0.445
-0.115       0.050
```

| | | | | | | |
|---|---|---|---|---|---|---|
| BldgType[T.TwnhsE] | 0.0022 | 0.039 | 0.057 | 0.954 | -0.074 | 0.079 |
| BsmtCond[T.Fa] | 0.0069 | 0.055 | 0.127 | 0.899 | -0.100 | 0.114 |
| BsmtCond[T.Gd] | 0.0116 | 0.054 | 0.215 | 0.830 | -0.094 | 0.117 |
| BsmtCond[T.NaN] | -0.7545 | 0.043 | -17.473 | 0.000 | -0.839 | -0.670 |
| BsmtCond[T.NoBasement] | -0.0166 | 0.014 | -1.153 | 0.249 | -0.045 | 0.012 |
| BsmtCond[T.Po] | -0.1085 | 0.089 | -1.216 | 0.224 | -0.284 | 0.067 |
| BsmtCond[T.TA] | 0.0084 | 0.053 | 0.158 | 0.875 | -0.096 | 0.112 |
| BsmtExposure[T.Gd] | 0.0269 | 0.010 | 2.826 | 0.005 | 0.008 | 0.046 |
| BsmtExposure[T.Mn] | -0.0260 | 0.010 | -2.672 | 0.008 | -0.045 | -0.007 |
| BsmtExposure[T.NaN] | 0.0131 | 0.061 | 0.215 | 0.830 | -0.107 | 0.133 |
| BsmtExposure[T.No] | -0.0217 | 0.007 | -2.959 | 0.003 | -0.036 | -0.007 |
| BsmtExposure[T.NoBasement] | -0.0166 | 0.014 | -1.153 | 0.249 | -0.045 | 0.012 |
| BsmtFinType1[T.BLQ] | -0.0129 | 0.009 | -1.366 | 0.172 | -0.031 | 0.006 |
| BsmtFinType1[T.GLQ] | 0.0008 | 0.008 | 0.095 | 0.924 | -0.016 | 0.017 |
| BsmtFinType1[T.LwQ] | -0.0370 | 0.012 | -3.138 | 0.002 | -0.060 | -0.014 |
| BsmtFinType1[T.NaN] | -0.7545 | 0.043 | -17.473 | 0.000 | -0.839 | -0.670 |
| BsmtFinType1[T.NoBasement] | -0.0166 | 0.014 | -1.153 | 0.249 | -0.045 | 0.012 |
| BsmtFinType1[T.Rec] | -0.0333 | 0.009 | -3.594 | 0.000 | -0.052 | -0.015 |
| BsmtFinType1[T.Unf] | -0.0155 | 0.011 | -1.378 | 0.168 | -0.037 | 0.007 |
| BsmtFinType2[T.BLQ] | -0.0291 | 0.023 | -1.248 | 0.212 | -0.075 | 0.017 |
| BsmtFinType2[T.GLQ] | -0.0021 | 0.027 | -0.078 | 0.938 | -0.056 | 0.051 |
| BsmtFinType2[T.LwQ] | -0.0309 | 0.022 | -1.406 | 0.160 | -0.074 | 0.012 |
| BsmtFinType2[T.NaN] | -0.7545 | 0.043 | -17.473 | 0.000 | -0.839 | -0.670 |
| BsmtFinType2[T.NoBasement] | -0.0166 | 0.014 | -1.153 | 0.249 | -0.045 | 0.012 |

| | | | | | | |
|---|---|---|---|---|---|---|
| BsmtFinType2[T.Rec] | -0.0371 | 0.021 | -1.762 | 0.078 | -0.078 | 0.004 |
| BsmtFinType2[T.Unf] | -0.0284 | 0.025 | -1.138 | 0.255 | -0.077 | 0.021 |
| BsmtFullBath[T.1] | 0.0120 | 0.007 | 1.798 | 0.072 | -0.001 | 0.025 |
| BsmtFullBath[T.2] | 0.0812 | 0.033 | 2.465 | 0.014 | 0.017 | 0.146 |
| BsmtHalfBath[T.1] | 0.0145 | 0.010 | 1.454 | 0.146 | -0.005 | 0.034 |
| BsmtHalfBath[T.2] | 0.1422 | 0.120 | 1.187 | 0.235 | -0.093 | 0.377 |
| BsmtQual[T.Fa] | -0.0543 | 0.020 | -2.696 | 0.007 | -0.094 | -0.015 |
| BsmtQual[T.Gd] | -0.0198 | 0.011 | -1.860 | 0.063 | -0.041 | 0.001 |
| BsmtQual[T.NaN] | -0.7545 | 0.043 | -17.473 | 0.000 | -0.839 | -0.670 |
| BsmtQual[T.NoBasement] | -0.0166 | 0.014 | -1.153 | 0.249 | -0.045 | 0.012 |
| BsmtQual[T.Po] | 0.0744 | 0.089 | 0.836 | 0.403 | -0.100 | 0.249 |
| BsmtQual[T.TA] | -0.0123 | 0.014 | -0.900 | 0.368 | -0.039 | 0.015 |
| CentralAir[T.Y] | 0.0222 | 0.013 | 1.661 | 0.097 | -0.004 | 0.048 |
| Electrical[T.FuseF] | -0.0063 | 0.022 | -0.288 | 0.774 | -0.049 | 0.037 |
| Electrical[T.FuseP] | 0.0237 | 0.056 | 0.420 | 0.674 | -0.087 | 0.134 |
| Electrical[T.Mix] | -0.1085 | 0.089 | -1.216 | 0.224 | -0.284 | 0.067 |
| Electrical[T.NaN] | 0.0382 | 0.088 | 0.434 | 0.664 | -0.135 | 0.211 |
| Electrical[T.SBrkr] | -0.0080 | 0.010 | -0.786 | 0.432 | -0.028 | 0.012 |
| ExterCond[T.Fa] | -0.0025 | 0.041 | -0.060 | 0.952 | -0.084 | 0.079 |
| ExterCond[T.Gd] | 0.0187 | 0.036 | 0.516 | 0.606 | -0.052 | 0.090 |
| ExterCond[T.Po] | 1.4030 | 0.234 | 6.000 | 0.000 | 0.944 | 1.862 |
| ExterCond[T.TA] | 0.0310 | 0.036 | 0.861 | 0.390 | -0.040 | 0.102 |
| ExterQual[T.Fa] | -0.0100 | 0.039 | -0.256 | 0.798 | -0.087 | 0.067 |
| ExterQual[T.Gd] | -0.0005 | 0.018 | -0.030 | 0.976 | -0.035 | 0.034 |

| | | | | | | |
|---|---|---|---|---|---|---|
| ExterQual[T.TA] | -0.0108 | 0.019 | -0.554 | 0.580 | -0.049 | 0.027 |
| Fence[T.GdWo] | 0.0020 | 0.016 | 0.119 | 0.905 | -0.030 | 0.034 |
| Fence[T.MnPrv] | -0.0081 | 0.013 | -0.613 | 0.540 | -0.034 | 0.018 |
| Fence[T.MnWw] | -0.0169 | 0.031 | -0.549 | 0.583 | -0.077 | 0.043 |
| Fence[T.NoFence] | -0.0032 | 0.012 | -0.273 | 0.785 | -0.026 | 0.020 |
| FireplaceQu[T.Fa] | -0.0012 | 0.023 | -0.052 | 0.958 | -0.046 | 0.044 |
| FireplaceQu[T.Gd] | 0.0051 | 0.018 | 0.278 | 0.781 | -0.031 | 0.041 |
| FireplaceQu[T.NoFirePlace] | 2.9952 | 0.082 | 36.637 | 0.000 | 2.835 | 3.156 |
| FireplaceQu[T.Po] | -0.0172 | 0.025 | -0.677 | 0.499 | -0.067 | 0.033 |
| FireplaceQu[T.TA] | -0.0043 | 0.019 | -0.223 | 0.824 | -0.042 | 0.033 |
| Fireplaces[T.1] | 3.0216 | 0.082 | 36.907 | 0.000 | 2.861 | 3.182 |
| Fireplaces[T.2] | 3.0565 | 0.082 | 37.086 | 0.000 | 2.895 | 3.218 |
| Fireplaces[T.3] | 3.0135 | 0.087 | 34.716 | 0.000 | 2.843 | 3.184 |
| Fireplaces[T.4] | 2.8565 | 0.186 | 15.394 | 0.000 | 2.493 | 3.220 |
| Foundation[T.CBlock] | 0.0216 | 0.011 | 1.943 | 0.052 | -0.000 | 0.043 |
| Foundation[T.PConc] | 0.0242 | 0.012 | 2.056 | 0.040 | 0.001 | 0.047 |
| Foundation[T.Slab] | 0.0018 | 0.036 | 0.049 | 0.961 | -0.068 | 0.072 |
| Foundation[T.Stone] | -0.0180 | 0.047 | -0.387 | 0.698 | -0.109 | 0.073 |
| Foundation[T.Wood] | 0.0413 | 0.064 | 0.641 | 0.521 | -0.085 | 0.168 |
| FullBath[T.1] | 0.0427 | 0.076 | 0.565 | 0.572 | -0.105 | 0.191 |
| FullBath[T.2] | 0.0605 | 0.076 | 0.792 | 0.429 | -0.089 | 0.210 |
| FullBath[T.3] | 0.1131 | 0.078 | 1.448 | 0.148 | -0.040 | 0.266 |
| FullBath[T.4] | 2.24e-14 | 1.46e-15 | 15.327 | 0.000 | 1.95e-14 | 2.53e-14 |
| Functional[T.Maj2] | 0.0230 | 0.069 | 0.334 | 0.739 | -0.112 | 0.158 |

| | | | | | | |
|---|---|---|---|---|---|---|
| Functional[T.Min1] | 0.0628 | 0.033 | 1.920 | 0.055 | -0.001 | 0.127 |
| Functional[T.Min2] | 0.0276 | 0.034 | 0.821 | 0.412 | -0.038 | 0.094 |
| Functional[T.Mod] | 0.0274 | 0.037 | 0.749 | 0.454 | -0.044 | 0.099 |
| Functional[T.Sal] | -0.3687 | 0.152 | -2.433 | 0.015 | -0.666 | -0.071 |
| Functional[T.Sev] | -0.2319 | 0.109 | -2.126 | 0.034 | -0.446 | -0.018 |
| Functional[T.Typ] | 0.0875 | 0.029 | 3.020 | 0.003 | 0.031 | 0.144 |
| GarageCars[T.2] | 0.0186 | 0.010 | 1.856 | 0.064 | -0.001 | 0.038 |
| GarageCars[T.3] | 0.0462 | 0.016 | 2.849 | 0.004 | 0.014 | 0.078 |
| GarageCars[T.4] | 0.0584 | 0.040 | 1.466 | 0.143 | -0.020 | 0.137 |
| GarageCars[T.5] | 2.698e-15 | 1.03e-15 | 2.622 | 0.009 | 6.79e-16 | 4.72e-15 |
| GarageCond[T.Fa] | 0.0675 | 0.082 | 0.822 | 0.411 | -0.094 | 0.229 |
| GarageCond[T.Gd] | 0.0868 | 0.083 | 1.043 | 0.297 | -0.076 | 0.250 |
| GarageCond[T.Po] | 0.1992 | 0.091 | 2.185 | 0.029 | 0.020 | 0.378 |
| GarageCond[T.TA] | 0.0833 | 0.081 | 1.033 | 0.302 | -0.075 | 0.241 |
| GarageFinish[T.RFn] | -0.0010 | 0.006 | -0.156 | 0.876 | -0.013 | 0.011 |
| GarageFinish[T.Unf] | -0.0060 | 0.008 | -0.781 | 0.435 | -0.021 | 0.009 |
| GarageQual[T.Fa] | -0.1885 | 0.098 | -1.923 | 0.055 | -0.381 | 0.004 |
| GarageQual[T.Gd] | -0.1530 | 0.096 | -1.597 | 0.110 | -0.341 | 0.035 |
| GarageQual[T.Po] | -0.1660 | 0.129 | -1.284 | 0.199 | -0.420 | 0.088 |
| GarageQual[T.TA] | -0.1662 | 0.097 | -1.715 | 0.086 | -0.356 | 0.024 |
| GarageType[T.Attchd] | 0.0432 | 0.024 | 1.821 | 0.069 | -0.003 | 0.090 |
| GarageType[T.Basment] | 0.0214 | 0.033 | 0.643 | 0.520 | -0.044 | 0.087 |
| GarageType[T.BuiltIn] | 0.0358 | 0.026 | 1.378 | 0.168 | -0.015 | 0.087 |
| GarageType[T.CarPort] | -0.0107 | 0.040 | -0.271 | 0.786 | -0.088 | 0.067 |

| | | | | | | |
|---|---|---|---|---|---|---|
| GarageType[T.Detchd] | 0.0555 | 0.024 | 2.343 | 0.019 | 0.009 | 0.102 |
| HalfBath[T.1] | 0.0209 | 0.007 | 2.830 | 0.005 | 0.006 | 0.035 |
| HalfBath[T.2] | -0.0971 | 0.033 | -2.970 | 0.003 | -0.161 | -0.033 |
| Heating[T.GasA] | 0.1664 | 0.097 | 1.716 | 0.086 | -0.024 | 0.357 |
| Heating[T.GasW] | 0.1591 | 0.100 | 1.593 | 0.111 | -0.037 | 0.355 |
| Heating[T.Grav] | -0.0336 | 0.158 | -0.213 | 0.831 | -0.343 | 0.276 |
| Heating[T.OthW] | 0.1002 | 0.135 | 0.742 | 0.458 | -0.165 | 0.365 |
| Heating[T.Wall] | 0.1925 | 0.119 | 1.624 | 0.105 | -0.040 | 0.425 |
| HeatingQC[T.Fa] | -0.0532 | 0.016 | -3.393 | 0.001 | -0.084 | -0.022 |
| HeatingQC[T.Gd] | -0.0075 | 0.007 | -1.114 | 0.265 | -0.021 | 0.006 |
| HeatingQC[T.Po] | -0.1181 | 0.069 | -1.702 | 0.089 | -0.254 | 0.018 |
| HeatingQC[T.TA] | -0.0306 | 0.007 | -4.461 | 0.000 | -0.044 | -0.017 |
| HouseStyle[T.1.5Unf] | -0.0012 | 0.074 | -0.016 | 0.987 | -0.146 | 0.144 |
| HouseStyle[T.1Story] | 0.0209 | 0.033 | 0.636 | 0.525 | -0.044 | 0.085 |
| HouseStyle[T.2.5Fin] | -0.0739 | 0.070 | -1.055 | 0.292 | -0.211 | 0.063 |
| HouseStyle[T.2.5Unf] | 0.0440 | 0.046 | 0.957 | 0.339 | -0.046 | 0.134 |
| HouseStyle[T.2Story] | 0.0046 | 0.031 | 0.146 | 0.884 | -0.057 | 0.066 |
| HouseStyle[T.SFoyer] | 0.0292 | 0.045 | 0.652 | 0.515 | -0.059 | 0.117 |
| HouseStyle[T.SLvl] | 0.0588 | 0.047 | 1.262 | 0.207 | -0.033 | 0.150 |
| KitchenAbvGr[T.1] | -0.1250 | 0.091 | -1.367 | 0.172 | -0.304 | 0.054 |
| KitchenAbvGr[T.2] | -0.1718 | 0.097 | -1.778 | 0.076 | -0.361 | 0.018 |
| KitchenAbvGr[T.3] | -4.375e-15 | 1.12e-15 | -3.897 | 0.000 | -6.58e-15 | -2.17e-15 |
| KitchenQual[T.Fa] | -0.0534 | 0.022 | -2.404 | 0.016 | -0.097 | -0.010 |
| KitchenQual[T.Gd] | -0.0446 | 0.012 | -3.674 | 0.000 | -0.068 | -0.021 |

| | | | | | | |
|---|---|---|---|---|---|---|
| KitchenQual[T.Po] | -5.42e-16 | 1.14e-15 | -0.477 | 0.634 | -2.77e-15 | 1.69e-15 |
| KitchenQual[T.TA] | -0.0542 | 0.014 | -3.990 | 0.000 | -0.081 | -0.028 |
| LandContour[T.HLS] | 0.0213 | 0.017 | 1.256 | 0.209 | -0.012 | 0.055 |
| LandContour[T.Low] | 0.0109 | 0.024 | 0.454 | 0.650 | -0.036 | 0.058 |
| LandContour[T.Lvl] | 0.0052 | 0.013 | 0.402 | 0.688 | -0.020 | 0.030 |
| LandSlope[T.Mod] | 0.0163 | 0.014 | 1.191 | 0.234 | -0.011 | 0.043 |
| LandSlope[T.Sev] | 0.0025 | 0.042 | 0.060 | 0.952 | -0.081 | 0.086 |
| LotConfig[T.CulDSac] | 0.0096 | 0.011 | 0.874 | 0.382 | -0.012 | 0.031 |
| LotConfig[T.FR2] | -0.0247 | 0.014 | -1.741 | 0.082 | -0.053 | 0.003 |
| LotConfig[T.FR3] | -0.0092 | 0.028 | -0.329 | 0.742 | -0.064 | 0.046 |
| LotConfig[T.Inside] | -0.0009 | 0.006 | -0.150 | 0.880 | -0.013 | 0.011 |
| LotShape[T.IR2] | -0.0044 | 0.013 | -0.329 | 0.742 | -0.031 | 0.022 |
| LotShape[T.IR3] | 0.0037 | 0.028 | 0.132 | 0.895 | -0.051 | 0.059 |
| LotShape[T.Reg] | 0.0005 | 0.005 | 0.093 | 0.926 | -0.010 | 0.011 |
| MSSubClass[T.30] | -0.0582 | 0.017 | -3.491 | 0.000 | -0.091 | -0.026 |
| MSSubClass[T.40] | -0.0310 | 0.049 | -0.626 | 0.531 | -0.128 | 0.066 |
| MSSubClass[T.45] | 0.0251 | 0.076 | 0.330 | 0.742 | -0.124 | 0.175 |
| MSSubClass[T.50] | 0.0112 | 0.031 | 0.357 | 0.721 | -0.050 | 0.073 |
| MSSubClass[T.60] | -0.0209 | 0.031 | -0.665 | 0.506 | -0.082 | 0.041 |
| MSSubClass[T.70] | 0.0243 | 0.033 | 0.733 | 0.464 | -0.041 | 0.090 |
| MSSubClass[T.75] | 0.0014 | 0.049 | 0.028 | 0.978 | -0.095 | 0.098 |
| MSSubClass[T.80] | -0.0559 | 0.043 | -1.305 | 0.192 | -0.140 | 0.028 |
| MSSubClass[T.85] | 0.0159 | 0.039 | 0.412 | 0.680 | -0.060 | 0.092 |
| MSSubClass[T.90] | -0.0287 | 0.017 | -1.726 | 0.085 | -0.061 | 0.004 |

| | | | | | | |
|---|---|---|---|---|---|---|
| MSSubClass[T.120] | -0.0302 | 0.039 | -0.766 | 0.444 | -0.108 | 0.047 |
| MSSubClass[T.150] | -0.1521 | 0.118 | -1.284 | 0.199 | -0.384 | 0.080 |
| MSSubClass[T.160] | -0.1089 | 0.053 | -2.065 | 0.039 | -0.212 | -0.005 |
| MSSubClass[T.180] | -0.0694 | 0.062 | -1.116 | 0.265 | -0.192 | 0.053 |
| MSSubClass[T.190] | 0.0679 | 0.104 | 0.652 | 0.514 | -0.136 | 0.272 |
| MSZoning[T.C] | 1.4713 | 0.059 | 24.956 | 0.000 | 1.356 | 1.587 |
| MSZoning[T.FV] | 1.7984 | 0.051 | 35.292 | 0.000 | 1.698 | 1.898 |
| MSZoning[T.I] | 1.4924 | 0.141 | 10.608 | 0.000 | 1.216 | 1.768 |
| MSZoning[T.RH] | 1.7684 | 0.053 | 33.281 | 0.000 | 1.664 | 1.873 |
| MSZoning[T.RL] | 1.7799 | 0.048 | 37.131 | 0.000 | 1.686 | 1.874 |
| MSZoning[T.RM] | 1.7332 | 0.049 | 35.721 | 0.000 | 1.638 | 1.828 |
| MasVnrType[T.BrkFace] | 0.0196 | 0.023 | 0.862 | 0.389 | -0.025 | 0.064 |
| MasVnrType[T.CBlock] | -0.3480 | 0.117 | -2.965 | 0.003 | -0.578 | -0.118 |
| MasVnrType[T.NaN] | 0.0277 | 0.024 | 1.180 | 0.238 | -0.018 | 0.074 |
| MasVnrType[T.Stone] | 0.0157 | 0.024 | 0.651 | 0.515 | -0.032 | 0.063 |
| MiscFeature[T.NaN] | 0.0236 | 0.087 | 0.272 | 0.786 | -0.147 | 0.194 |
| MiscFeature[T.Othr] | 0.0872 | 0.087 | 0.999 | 0.318 | -0.084 | 0.258 |
| MiscFeature[T.Shed] | -0.0027 | 0.079 | -0.034 | 0.973 | -0.157 | 0.151 |
| MiscFeature[T.TenC] | -0.3784 | 0.162 | -2.341 | 0.019 | -0.695 | -0.061 |
| MoSold[T.2] | -0.0384 | 0.015 | -2.507 | 0.012 | -0.068 | -0.008 |
| MoSold[T.3] | -0.0219 | 0.014 | -1.577 | 0.115 | -0.049 | 0.005 |
| MoSold[T.4] | -0.0083 | 0.014 | -0.610 | 0.542 | -0.035 | 0.018 |
| MoSold[T.5] | 0.0011 | 0.013 | 0.086 | 0.932 | -0.024 | 0.026 |
| MoSold[T.6] | -0.0017 | 0.012 | -0.139 | 0.890 | -0.026 | 0.023 |

| | | | | |
|---|---|---|---|---|
| MoSold[T.7] | 0.0033 | 0.012 | 0.266 | 0.790 |
| -0.021 0.028 | | | | |
| MoSold[T.8] | -0.0147 | 0.013 | -1.112 | 0.266 |
| -0.041 0.011 | | | | |
| MoSold[T.9] | 0.0037 | 0.014 | 0.257 | 0.798 |
| -0.024 0.032 | | | | |
| MoSold[T.10] | -0.0216 | 0.014 | -1.533 | 0.126 |
| -0.049 0.006 | | | | |
| MoSold[T.11] | -0.0103 | 0.014 | -0.714 | 0.475 |
| -0.038 0.018 | | | | |
| MoSold[T.12] | -0.0104 | 0.015 | -0.677 | 0.499 |
| -0.040 0.020 | | | | |
| Neighborhood[T.Blueste] | 0.1267 | 0.054 | 2.326 | 0.020 |
| 0.020 0.234 | | | | |
| Neighborhood[T.BrDale] | 0.0042 | 0.039 | 0.107 | 0.915 |
| -0.073 0.081 | | | | |
| Neighborhood[T.BrkSide] | 0.0211 | 0.032 | 0.657 | 0.511 |
| -0.042 0.084 | | | | |
| Neighborhood[T.ClearCr] | 0.0209 | 0.034 | 0.621 | 0.534 |
| -0.045 0.087 | | | | |
| Neighborhood[T.CollgCr] | -0.0314 | 0.025 | -1.238 | 0.216 |
| -0.081 0.018 | | | | |
| Neighborhood[T.Crawfor] | 0.0772 | 0.029 | 2.663 | 0.008 |
| 0.020 0.134 | | | | |
| Neighborhood[T.Edwards] | -0.0495 | 0.028 | -1.789 | 0.074 |
| -0.104 0.005 | | | | |
| Neighborhood[T.Gilbert] | -0.0302 | 0.026 | -1.154 | 0.249 |
| -0.082 0.021 | | | | |
| Neighborhood[T.Greens] | 0.0771 | 0.052 | 1.483 | 0.138 |
| -0.025 0.179 | | | | |
| Neighborhood[T.GrnHill] | 0.4596 | 0.069 | 6.657 | 0.000 |
| 0.324 0.595 | | | | |
| Neighborhood[T.IDOTRR] | 0.0129 | 0.035 | 0.367 | 0.714 |
| -0.056 0.082 | | | | |
| Neighborhood[T.Landmrk] | 0.0131 | 0.094 | 0.139 | 0.890 |
| -0.172 0.198 | | | | |
| Neighborhood[T.MeadowV] | -0.0561 | 0.041 | -1.361 | 0.174 |
| -0.137 0.025 | | | | |
| Neighborhood[T.Mitchel] | -0.0392 | 0.028 | -1.384 | 0.166 |
| -0.095 0.016 | | | | |
| Neighborhood[T.NAmes] | -0.0209 | 0.027 | -0.773 | 0.440 |
| -0.074 0.032 | | | | |
| Neighborhood[T.NPkVill] | 0.0059 | 0.057 | 0.103 | 0.918 |
| -0.106 0.118 | | | | |
| Neighborhood[T.NWAmes] | -0.0254 | 0.028 | -0.901 | 0.368 |
| -0.081 0.030 | | | | |
| Neighborhood[T.NoRidge] | 0.0299 | 0.030 | 1.001 | 0.317 |
| -0.029 0.089 | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Neighborhood[T.NridgHt] | 0.0288 | 0.026 | 1.104 | 0.270 | -0.022 | 0.080 |
| Neighborhood[T.OldTown] | -0.0178 | 0.032 | -0.551 | 0.581 | -0.081 | 0.046 |
| Neighborhood[T.SWISU] | -0.0364 | 0.035 | -1.046 | 0.296 | -0.105 | 0.032 |
| Neighborhood[T.Sawyer] | 0.0084 | 0.028 | 0.302 | 0.762 | -0.046 | 0.063 |
| Neighborhood[T.SawyerW] | -0.0412 | 0.028 | -1.487 | 0.137 | -0.096 | 0.013 |
| Neighborhood[T.Somerst] | 0.0144 | 0.030 | 0.488 | 0.626 | -0.044 | 0.073 |
| Neighborhood[T.StoneBr] | 0.0710 | 0.030 | 2.367 | 0.018 | 0.012 | 0.130 |
| Neighborhood[T.Timber] | -0.0154 | 0.027 | -0.563 | 0.574 | -0.069 | 0.038 |
| Neighborhood[T.Veenker] | -0.0114 | 0.034 | -0.341 | 0.733 | -0.077 | 0.054 |
| OverallCond[T.2] | 0.0450 | 0.158 | 0.284 | 0.776 | -0.266 | 0.356 |
| OverallCond[T.3] | 0.0153 | 0.122 | 0.126 | 0.900 | -0.224 | 0.255 |
| OverallCond[T.4] | 0.1183 | 0.122 | 0.971 | 0.332 | -0.121 | 0.357 |
| OverallCond[T.5] | 0.2038 | 0.122 | 1.673 | 0.094 | -0.035 | 0.443 |
| OverallCond[T.6] | 0.2354 | 0.122 | 1.926 | 0.054 | -0.004 | 0.475 |
| OverallCond[T.7] | 0.2765 | 0.122 | 2.259 | 0.024 | 0.036 | 0.517 |
| OverallCond[T.8] | 0.2983 | 0.122 | 2.438 | 0.015 | 0.058 | 0.538 |
| OverallCond[T.9] | 0.3423 | 0.125 | 2.748 | 0.006 | 0.098 | 0.587 |
| OverallQual[T.2] | 0.7610 | 0.058 | 13.213 | 0.000 | 0.648 | 0.874 |
| OverallQual[T.3] | 1.0193 | 0.037 | 27.679 | 0.000 | 0.947 | 1.092 |
| OverallQual[T.4] | 1.0455 | 0.031 | 33.832 | 0.000 | 0.985 | 1.106 |
| OverallQual[T.5] | 1.1034 | 0.030 | 36.215 | 0.000 | 1.044 | 1.163 |
| OverallQual[T.6] | 1.1296 | 0.031 | 36.839 | 0.000 | 1.069 | 1.190 |
| OverallQual[T.7] | 1.1648 | 0.031 | 37.683 | 0.000 | 1.104 | 1.225 |
| OverallQual[T.8] | 1.2162 | 0.032 | 38.586 | 0.000 | 1.154 | 1.278 |

| | | | | | | |
|---|---|---|---|---|---|---|
| OverallQual[T.9] | 1.2776 | 0.034 | 38.117 | 0.000 | 1.212 | 1.343 |
| OverallQual[T.10] | 1.3260 | 0.038 | 34.507 | 0.000 | 1.251 | 1.401 |
| PavedDrive[T.P] | 0.0032 | 0.018 | 0.177 | 0.859 | -0.032 | 0.039 |
| PavedDrive[T.Y] | 0.0293 | 0.013 | 2.332 | 0.020 | 0.005 | 0.054 |
| PoolQC[T.Fa] | 0.3093 | 0.167 | 1.852 | 0.064 | -0.018 | 0.637 |
| PoolQC[T.Gd] | 0.4395 | 0.156 | 2.824 | 0.005 | 0.134 | 0.745 |
| PoolQC[T.NoPool] | -0.2178 | 0.108 | -2.012 | 0.044 | -0.430 | -0.005 |
| PoolQC[T.TA] | 0.0348 | 0.104 | 0.333 | 0.739 | -0.170 | 0.239 |
| RoofMatl[T.Membran] | 0.0795 | 0.115 | 0.689 | 0.491 | -0.147 | 0.306 |
| RoofMatl[T.Metal] | 0.0230 | 0.112 | 0.206 | 0.837 | -0.197 | 0.243 |
| RoofMatl[T.Roll] | 0.0743 | 0.096 | 0.774 | 0.439 | -0.114 | 0.262 |
| RoofMatl[T.Tar&Grv] | -0.0012 | 0.045 | -0.026 | 0.979 | -0.090 | 0.087 |
| RoofMatl[T.WdShake] | -0.0181 | 0.038 | -0.472 | 0.637 | -0.093 | 0.057 |
| RoofMatl[T.WdShngl] | 0.0631 | 0.051 | 1.227 | 0.220 | -0.038 | 0.164 |
| RoofStyle[T.Gable] | -0.0082 | 0.052 | -0.158 | 0.874 | -0.110 | 0.093 |
| RoofStyle[T.Gambrel] | -0.0521 | 0.058 | -0.904 | 0.366 | -0.165 | 0.061 |
| RoofStyle[T.Hip] | -0.0008 | 0.052 | -0.015 | 0.988 | -0.103 | 0.101 |
| RoofStyle[T.Mansard] | -0.1099 | 0.064 | -1.711 | 0.087 | -0.236 | 0.016 |
| RoofStyle[T.Shed] | -0.0457 | 0.081 | -0.563 | 0.574 | -0.205 | 0.113 |
| SaleCondition[T.AdjLand] | 0.2002 | 0.043 | 4.658 | 0.000 | 0.116 | 0.285 |
| SaleCondition[T.Alloca] | 0.0725 | 0.034 | 2.127 | 0.034 | 0.006 | 0.139 |
| SaleCondition[T.Family] | 0.0227 | 0.018 | 1.233 | 0.218 | -0.013 | 0.059 |
| SaleCondition[T.Normal] | 0.0390 | 0.010 | 3.881 | 0.000 | 0.019 | 0.059 |
| SaleCondition[T.Partial] | 0.0961 | 0.046 | 2.077 | 0.038 | 0.005 | 0.187 |

| | | | | | | |
|---|---|---|---|---|---|---|
| SaleType[T.CWD] | 0.0156 | 0.029 | 0.539 | 0.590 | -0.041 | 0.072 |
| SaleType[T.Con] | 0.0544 | 0.054 | 1.007 | 0.314 | -0.052 | 0.160 |
| SaleType[T.ConLD] | 0.0277 | 0.031 | 0.898 | 0.369 | -0.033 | 0.088 |
| SaleType[T.ConLI] | -0.2024 | 0.057 | -3.531 | 0.000 | -0.315 | -0.090 |
| SaleType[T.ConLw] | -0.1059 | 0.048 | -2.196 | 0.028 | -0.201 | -0.011 |
| SaleType[T.New] | -0.0274 | 0.048 | -0.567 | 0.571 | -0.122 | 0.067 |
| SaleType[T.Oth] | 0.1478 | 0.088 | 1.678 | 0.094 | -0.025 | 0.321 |
| SaleType[T.VWD] | -0.0313 | 0.090 | -0.348 | 0.728 | -0.208 | 0.145 |
| SaleType[T.WD ] | 0.0034 | 0.014 | 0.245 | 0.806 | -0.024 | 0.031 |
| Street[T.Pave] | 0.0204 | 0.044 | 0.460 | 0.645 | -0.067 | 0.107 |
| TotRmsAbvGrd[T.4] | 0.0211 | 0.032 | 0.670 | 0.503 | -0.041 | 0.083 |
| TotRmsAbvGrd[T.5] | 0.0158 | 0.032 | 0.497 | 0.619 | -0.046 | 0.078 |
| TotRmsAbvGrd[T.6] | 0.0185 | 0.033 | 0.568 | 0.570 | -0.045 | 0.082 |
| TotRmsAbvGrd[T.7] | 0.0113 | 0.034 | 0.337 | 0.736 | -0.055 | 0.077 |
| TotRmsAbvGrd[T.8] | 0.0164 | 0.035 | 0.474 | 0.636 | -0.052 | 0.084 |
| TotRmsAbvGrd[T.9] | -0.0057 | 0.036 | -0.158 | 0.874 | -0.077 | 0.065 |
| TotRmsAbvGrd[T.10] | -0.0178 | 0.039 | -0.462 | 0.644 | -0.093 | 0.058 |
| TotRmsAbvGrd[T.11] | -0.0326 | 0.042 | -0.768 | 0.443 | -0.116 | 0.051 |
| TotRmsAbvGrd[T.12] | 0.0897 | 0.076 | 1.180 | 0.238 | -0.059 | 0.239 |
| TotRmsAbvGrd[T.13] | -3.155e-16 | 9.89e-17 | -3.189 | 0.001 | -5.1e-16 | -1.21e-16 |
| Utilities[T.NoSeWa] | 2.269e-16 | 1.48e-16 | 1.532 | 0.126 | -6.37e-17 | 5.17e-16 |
| Utilities[T.NoSewr] | -0.1244 | 0.137 | -0.907 | 0.364 | -0.394 | 0.145 |
| Artery | -0.0918 | 0.015 | -6.155 | 0.000 | -0.121 | -0.063 |
| AsbShng | -0.0204 | 0.024 | -0.844 | 0.399 | -0.068 | 0.027 |

| | | | | | | |
|---|---|---|---|---|---|---|
| AsphShn | 0.1508 | 0.068 | 2.231 | 0.026 | 0.018 | 0.283 |
| BrkCmn | 0.0462 | 0.062 | 0.747 | 0.455 | -0.075 | 0.167 |
| BrkComm | 0.1891 | 0.066 | 2.877 | 0.004 | 0.060 | 0.318 |
| BrkFace | 0.0648 | 0.015 | 4.445 | 0.000 | 0.036 | 0.093 |
| BsmtFinSF1 | 9.94e-05 | 2.54e-05 | 3.915 | 0.000 | 4.96e-05 | 0.000 |
| BsmtFinSF1_p2 | -3.784e-08 | 1.73e-08 | -2.187 | 0.029 | -7.18e-08 | -3.89e-09 |
| BsmtFinSF2 | -1.364e-06 | 5.17e-05 | -0.026 | 0.979 | -0.000 | 0.000 |
| BsmtFinSF2_p2 | 4.068e-08 | 5.78e-08 | 0.704 | 0.481 | -7.26e-08 | 1.54e-07 |
| BsmtUnfSF | -3.462e-05 | 2.4e-05 | -1.444 | 0.149 | -8.17e-05 | 1.24e-05 |
| BsmtUnfSF_p2 | 8.152e-09 | 1.12e-08 | 0.729 | 0.466 | -1.38e-08 | 3.01e-08 |
| CBlock | 2.7422 | 0.110 | 24.878 | 0.000 | 2.526 | 2.958 |
| CemntBd | -0.0743 | 0.065 | -1.142 | 0.254 | -0.202 | 0.053 |
| CmentBd | 0.0959 | 0.065 | 1.470 | 0.142 | -0.032 | 0.224 |
| EnclosedPorch | -9.118e-05 | 6.25e-05 | -1.458 | 0.145 | -0.000 | 3.15e-05 |
| EnclosedPorch_p2 | 4.909e-07 | 1.46e-07 | 3.369 | 0.001 | 2.05e-07 | 7.77e-07 |
| Feedr | -0.0555 | 0.011 | -5.251 | 0.000 | -0.076 | -0.035 |
| GarageArea | -4.593e-06 | 8.08e-05 | -0.057 | 0.955 | -0.000 | 0.000 |
| GarageArea_p2 | 5.741e-08 | 5.91e-08 | 0.971 | 0.332 | -5.86e-08 | 1.73e-07 |
| GrLivArea | 0.0003 | 7.98e-05 | 3.686 | 0.000 | 0.000 | 0.000 |
| GrLivArea_p2 | -3.041e-08 | 1.26e-08 | -2.421 | 0.016 | -5.5e-08 | -5.77e-09 |
| HdBoard | 0.0004 | 0.012 | 0.034 | 0.973 | -0.023 | 0.024 |
| ImStucc | -0.0246 | 0.030 | -0.834 | 0.404 | -0.083 | 0.033 |
| LotArea | 6.268e-06 | 8.91e-07 | 7.035 | 0.000 | 4.52e-06 | 8.02e-06 |
| LotArea_p2 | -3.289e-11 | 6.34e-12 | -5.186 | 0.000 | -4.53e-11 | -2.05e-11 |

| | | | | | | |
|---|---|---|---|---|---|---|
| LotFrontage | -7.06e-06 | 0.000 | -0.047 | 0.962 | -0.000 | 0.000 |
| LotFrontage_p2 | 5.364e-07 | 1.16e-06 | 0.464 | 0.643 | -1.73e-06 | 2.81e-06 |
| LowQualFinSF | 2.652e-05 | 0.000 | 0.119 | 0.905 | -0.000 | 0.000 |
| LowQualFinSF_p2 | -7.349e-08 | 5.25e-07 | -0.140 | 0.889 | -1.1e-06 | 9.57e-07 |
| MasVnrArea | 9.059e-05 | 4.38e-05 | 2.069 | 0.039 | 4.73e-06 | 0.000 |
| MasVnrArea_p2 | -4.991e-08 | 4e-08 | -1.249 | 0.212 | -1.28e-07 | 2.85e-08 |
| MetalSd | 0.0174 | 0.012 | 1.416 | 0.157 | -0.007 | 0.042 |
| MiscVal | 2.305e-05 | 2.05e-05 | 1.126 | 0.260 | -1.71e-05 | 6.32e-05 |
| MiscVal_p2 | -9.851e-10 | 1.21e-09 | -0.814 | 0.416 | -3.36e-09 | 1.39e-09 |
| Norm | 0.0571 | 0.026 | 2.196 | 0.028 | 0.006 | 0.108 |
| OpenPorchSF | 8.498e-05 | 7.67e-05 | 1.108 | 0.268 | -6.55e-05 | 0.000 |
| OpenPorchSF_p2 | 1.204e-07 | 2.54e-07 | 0.474 | 0.636 | -3.78e-07 | 6.19e-07 |
| Other | -0.0595 | 0.087 | -0.683 | 0.494 | -0.230 | 0.111 |
| Plywood | 0.0006 | 0.011 | 0.052 | 0.959 | -0.022 | 0.023 |
| PoolArea | -0.0007 | 0.000 | -2.091 | 0.037 | -0.001 | -4.32e-05 |
| PosA | 0.0471 | 0.027 | 1.773 | 0.076 | -0.005 | 0.099 |
| PosN | 0.0237 | 0.020 | 1.164 | 0.245 | -0.016 | 0.064 |
| PreCast | -3.194e-19 | 1.39e-17 | -0.023 | 0.982 | -2.75e-17 | 2.69e-17 |
| RRAe | -0.0554 | 0.023 | -2.460 | 0.014 | -0.100 | -0.011 |
| RRAn | -0.0160 | 0.016 | -0.989 | 0.323 | -0.048 | 0.016 |
| RRNe | -0.0211 | 0.046 | -0.456 | 0.649 | -0.112 | 0.070 |
| RRNn | 0.0639 | 0.041 | 1.559 | 0.119 | -0.016 | 0.144 |
| ScreenPorch | 0.0003 | 9.24e-05 | 3.149 | 0.002 | 0.000 | 0.000 |
| ScreenPorch_p2 | -4.331e-07 | 3.31e-07 | -1.310 | 0.190 | -1.08e-06 | 2.16e-07 |

```
Stone                              -0.0021      0.047     -0.044      0.965
-0.095       0.091
Stucco                              0.0205      0.019      1.108      0.268
-0.016       0.057
TotalBsmtSF                      6.341e-05   3.41e-05      1.859      0.063
-3.49e-06       0.000
TotalBsmtSF_p2                   3.152e-09   1.53e-08      0.206      0.837
-2.69e-08    3.32e-08
VinylSd                             0.0059      0.013      0.457      0.648
-0.019       0.031
WdSdng                              0.0088      0.012      0.727      0.467
-0.015       0.033
WdShing                             0.0061      0.022      0.278      0.781
-0.037       0.049
WdShng                              0.0041      0.018      0.231      0.818
-0.031       0.039
WoodDeckSF                       8.018e-05   3.14e-05      2.550      0.011
1.85e-05       0.000
WoodDeckSF_p2                   -7.926e-08   5.42e-08     -1.462      0.144
-1.86e-07    2.71e-08
X1stFlrSF                           0.0002   8.13e-05      2.313      0.021
2.86e-05       0.000
X1stFlrSF_p2                    -2.878e-08    1.4e-08     -2.059      0.040
-5.62e-08   -1.37e-09
X2ndFlrSF                        7.968e-05   7.94e-05      1.004      0.315
-7.6e-05       0.000
X2ndFlrSF_p2                     1.574e-08   2.92e-08      0.540      0.589
-4.15e-08    7.29e-08
X3SsnPorch                       8.546e-05      0.000      0.292      0.770
-0.000       0.001
X3SsnPorch_p2                    4.591e-07   1.04e-06      0.443      0.658
-1.57e-06    2.49e-06
YearsSince1950Built              5.089e-05      0.001      0.036      0.972
-0.003       0.003
YearsSince1950Built_p2           6.621e-05   2.77e-05      2.393      0.017
1.19e-05       0.000
YearsSince1950GarageBuilt           0.0006      0.001      0.528      0.597
-0.002       0.003
YearsSince1950GarageBuilt_p2 -1.493e-05   2.13e-05     -0.700      0.484
-5.68e-05    2.69e-05
YearsSince1950Remod                 0.0015      0.001      1.889      0.059
-5.77e-05       0.003
YearsSince1950Remod_p2          -2.076e-05   1.44e-05     -1.444      0.149
-4.9e-05    7.45e-06
==============================================================================
Omnibus:                      303.264   Durbin-Watson:                   1.912
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             2249.910
Skew:                          -0.564   Prob(JB):                         0.00
```

```
Kurtosis:                       8.317   Cond. No.                    4.48e+17
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The smallest eigenvalue is 8.45e-15. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

[ ]:
```
print_metrics(nlr_poly, ames_train_poly, y_train, ames_test_poly, y_test,␣
 ↪flag_log_sale_price = True)
print_metrics(nlr_poly, ames_train_poly, y_train, ames_test_poly, y_test,␣
 ↪flag_log_sale_price = False)
```

```
Metrics for Log(Sale Price):

Training R2 0.9624105774575185
Training MAE 0.05428713187831344
Training RMSE 0.07504602800253764
Out-of-sample R2 0.8389618470389943
Out-of-sample MAE 0.08377525015006256
Out-of-sample RMSE 0.15893909023513655

Metrics for Sale Price:

Training R2 0.9640398351503799
Training MAE 9959.9415261308
Training RMSE 14814.210314147367
Out-of-sample R2 0.42824990275266905
Out-of-sample MAE 15888.470959526549
Out-of-sample RMSE 59264.318499926754
```

[ ]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
```

### 1.2 Tuning parameters for LASSO

[ ]:
```
X_train_poly = ames_train_poly.drop(columns='LogSalePrice')
X_test_poly = ames_test_poly.drop(columns='LogSalePrice')

X_train_poly_wide = pd.get_dummies(X_train_poly)
X_test_poly_wide = pd.get_dummies(X_test_poly)
```

[ ]:
```
X_train_lasso = X_train_poly_wide
X_test_lasso = X_test_poly_wide

print(X_train_lasso.shape, X_test_lasso.shape)
```

```
(1828, 397) (937, 397)
```

[ ]:
```

```

[ ]:
```
alpha = 0.1
lasso = Lasso(alpha=alpha, random_state=88)
lasso.fit(X_train_lasso, y_train)
print_metrics(lasso, X_train_lasso, y_train, X_test_lasso, y_test,␣
 ↪flag_log_sale_price = True)
print_metrics(lasso, X_train_lasso, y_train, X_test_lasso, y_test,␣
 ↪flag_log_sale_price = False)
```

```
Metrics for Log(Sale Price):

Training R2 0.8434328276296639
Training MAE 0.10512146274745086
Training RMSE 0.15315995785273875
Out-of-sample R2 0.8272606660089641
Out-of-sample MAE 0.11098188208314018
Out-of-sample RMSE 0.1646121748477917

Metrics for Sale Price:

Training R2 0.8771753803277366
Training MAE 18703.62412014218
Training RMSE 27378.546173438564
Out-of-sample R2 0.8763647003108185
Out-of-sample MAE 19243.304403735783
Out-of-sample RMSE 27558.870499831355
```

[ ]:
```
alpha = 1e-2
lasso = Lasso(alpha=alpha, random_state=88)
lasso.fit(X_train_lasso, y_train)
print_metrics(lasso, X_train_lasso, y_train, X_test_lasso, y_test,␣
 ↪flag_log_sale_price = True)
print_metrics(lasso, X_train_lasso, y_train, X_test_lasso, y_test,␣
 ↪flag_log_sale_price = False)
```

```
Metrics for Log(Sale Price):

Training R2 0.8499429858739374
Training MAE 0.10300561266815048
```

```
Training RMSE 0.1499419084810546
Out-of-sample R2 0.8220062230158947
Out-of-sample MAE 0.11032085223765256
Out-of-sample RMSE 0.16709703425025807

Metrics for Sale Price:

Training R2 0.8817688305547458
Training MAE 18354.387859038405
Training RMSE 26861.71033834519
Out-of-sample R2 0.8640371155817245
Out-of-sample MAE 19243.388183114424
Out-of-sample RMSE 28900.16723185019
```

[ ]:
```
alpha = 1e-3
lasso = Lasso(alpha=alpha, random_state=88)
lasso.fit(X_train_lasso, y_train)
print_metrics(lasso, X_train_lasso, y_train, X_test_lasso, y_test,␣
↪flag_log_sale_price = True)
print_metrics(lasso, X_train_lasso, y_train, X_test_lasso, y_test,␣
↪flag_log_sale_price = False)
```

```
Metrics for Log(Sale Price):

Training R2 0.930121322344742
Training MAE 0.0693863840072313
Training RMSE 0.10232157655865007
Out-of-sample R2 0.8874024847675652
Out-of-sample MAE 0.08314147804623102
Out-of-sample RMSE 0.13290176107494575

Metrics for Sale Price:

Training R2 0.9414769024277362
Training MAE 12459.409164012955
Training RMSE 18898.673470545342
Out-of-sample R2 0.9194885352942028
Out-of-sample MAE 14287.152645232196
Out-of-sample RMSE 22239.193567377857
```

[ ]:
```
alpha = 1e-4
lasso = Lasso(alpha=alpha, random_state=88)
lasso.fit(X_train_lasso, y_train)
print_metrics(lasso, X_train_lasso, y_train, X_test_lasso, y_test,␣
↪flag_log_sale_price = True)
print_metrics(lasso, X_train_lasso, y_train, X_test_lasso, y_test,␣
↪flag_log_sale_price = False)
```

```
Metrics for Log(Sale Price):

Training R2 0.9591387227010627
Training MAE 0.05679795053144252
Training RMSE 0.07824396555230513
Out-of-sample R2 0.8989936623088266
Out-of-sample MAE 0.07800037399372267
Out-of-sample RMSE 0.12587533655731084

Metrics for Sale Price:

Training R2 0.9611684976129613
Training MAE 10369.737135417017
Training RMSE 15394.293622412848
Out-of-sample R2 0.9310061238902255
Out-of-sample MAE 13389.285161194184
Out-of-sample RMSE 20587.112764707137
```

[ ]:
```
alpha = 1e-5
lasso = Lasso(alpha=alpha, random_state=88)
lasso.fit(X_train_lasso, y_train)
print_metrics(lasso, X_train_lasso, y_train, X_test_lasso, y_test,␣
 ↪flag_log_sale_price = True)
print_metrics(lasso, X_train_lasso, y_train, X_test_lasso, y_test,␣
 ↪flag_log_sale_price = False)
```

```
Metrics for Log(Sale Price):

Training R2 0.9622832237041625
Training MAE 0.05457420034317798
Training RMSE 0.0751730492656569
Out-of-sample R2 0.8881083316233747
Out-of-sample MAE 0.07994225740632856
Out-of-sample RMSE 0.13248454152958558

Metrics for Sale Price:

Training R2 0.9638757207480595
Training MAE 10010.81646880421
Training RMSE 14847.976256226326
Out-of-sample R2 0.9272909790616876
Out-of-sample MAE 13681.307280622397
Out-of-sample RMSE 21134.127319826755
```

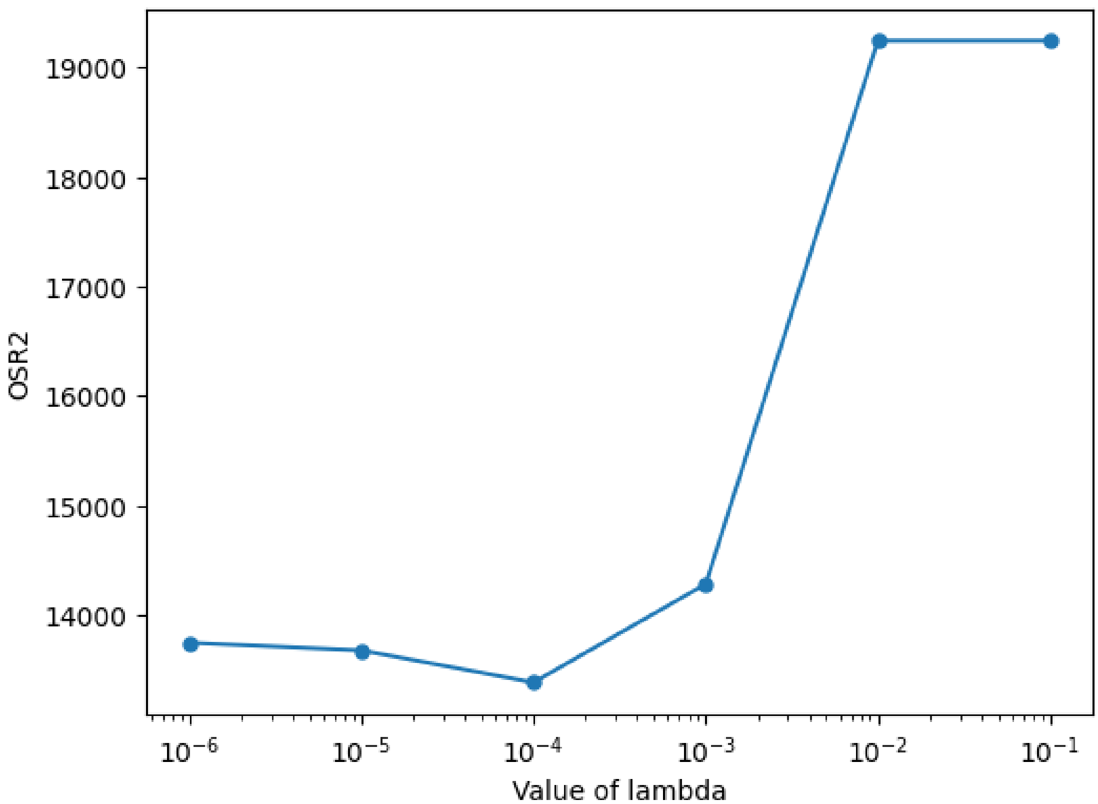
[ ]:
```
alpha = 1e-6
lasso = Lasso(alpha=alpha, random_state=88)
```

```
lasso.fit(X_train_lasso, y_train)
print_metrics(lasso, X_train_lasso, y_train, X_test_lasso, y_test,
    flag_log_sale_price = True)
print_metrics(lasso, X_train_lasso, y_train, X_test_lasso, y_test,
    flag_log_sale_price = False)
```

```
Metrics for Log(Sale Price):

Training R2 0.9624088075098128
Training MAE 0.05432119722133529
Training RMSE 0.07504779480251142
Out-of-sample R2 0.887174332291282
Out-of-sample MAE 0.08045121076601408
Out-of-sample RMSE 0.13303634007223722

Metrics for Sale Price:

Training R2 0.9640275202363419
Training MAE 9965.285133817228
Training RMSE 14816.7467334153
Out-of-sample R2 0.9267533208150691
Out-of-sample MAE 13751.7824584604
Out-of-sample RMSE 21212.12320497363
```

[ ]:
```
MAE_list = []
candidate_values = [1e-6, 1e-5, 1e-4, 1e-3, 1e-2, 1e-1]

for alpha in candidate_values:
    lasso = Lasso(alpha=alpha, random_state=88)
    lasso.fit(X_train_lasso, y_train)
    y_pred_test = pd.Series(lasso.predict(X_test_lasso)).apply(np.exp).
    reset_index(drop=True)
    y_train_exp = y_train.copy().apply(np.exp).reset_index(drop=True)
    y_test_exp = y_test.copy().apply(np.exp).reset_index(drop=True)
    MAE_list.append(MAE(y_test_exp, y_pred_test))
```

[ ]:
```
plt.plot(candidate_values, MAE_list,'o-', markersize = 5)
plt.xlabel('Value of lambda')
plt.ylabel('OSR2')
plt.xscale('log')
```




### 1.3 In-class activity 1: Create a similar plot for Ridge regression. The candidate value for labmda is `[ 1e-1, 1, 10, 1e2, 1e3, 1e4 ]`. Y-axis is the OSR2 and X-axis is the value of lambda.

[ ]:
```
OSR2_list = []
candidate = [ 1e-1, 1, 10, 1e2, 1e3, 1e4 ]

for alpha in candidate:
    ridge = Ridge(alpha=alpha, random_state=88)
    ridge.fit(X_train_lasso, y_train)
    y_pred_test = pd.Series(ridge.predict(X_test_lasso)).apply(np.exp).
 ↪reset_index(drop=True)
    y_train_exp = y_train.copy().apply(np.exp).reset_index(drop=True)
    y_test_exp = y_test.copy().apply(np.exp).reset_index(drop=True)
    MAE_list.append(MAE(y_test_exp, y_pred_test))
```

Add Training set performance to the graph.

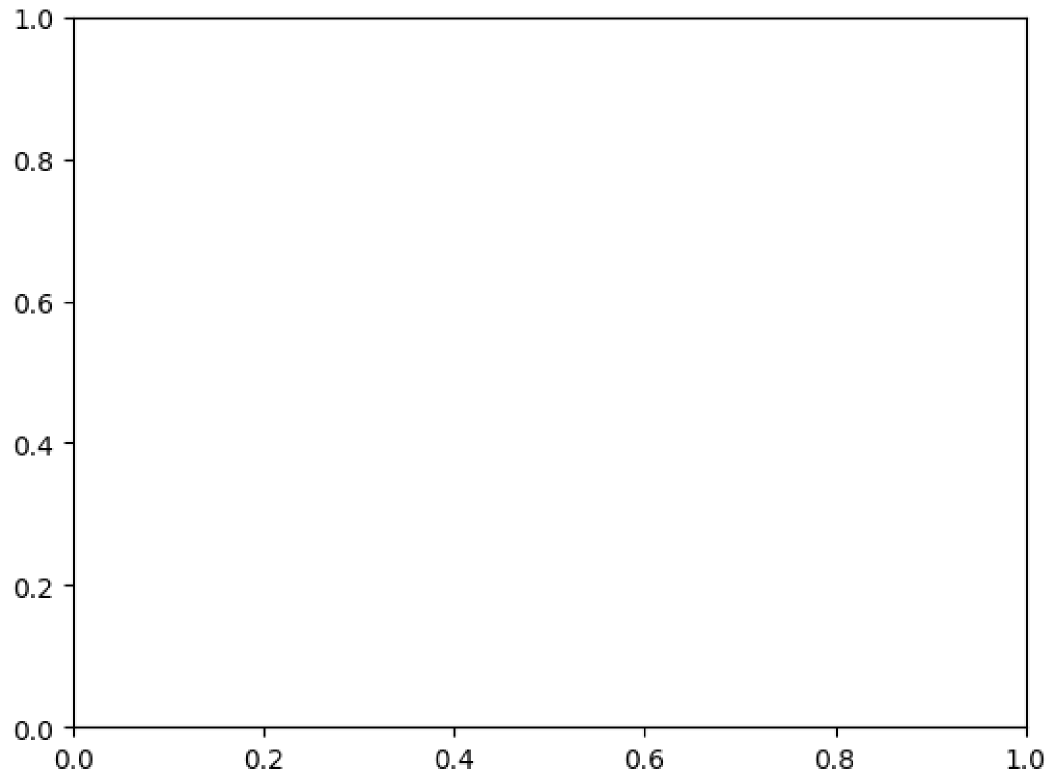
[ ]:
```
TrainingMAE_list = []
candidate_values = [1e-6, 1e-5, 1e-4, 1e-3, 1e-2, 1e-1]
```

```
for alpha in candidate_values:
    lasso = Lasso(alpha=alpha, random_state=88)
    lasso.fit(X_train_lasso, y_train)
    y_pred_train = pd.Series(lasso.predict(X_train_lasso)).apply(np.exp).
 ↪reset_index(drop=True)
    y_train_exp = y_train.copy().apply(np.exp).reset_index(drop=True)
    y_test_exp = y_test.copy().apply(np.exp).reset_index(drop=True)
    TrainingMAE_list.append(MAE(y_train_exp, y_pred_train))
```

[ ]:
```
plt.plot(candidate_values, MAE_list,'o-', markersize = 5, label =␣
 ↪'Out-of-sample')
plt.plot(candidate_values, TrainingMAE_list,'o-', markersize = 5, label =␣
 ↪'In-sample')
plt.xlabel('Value of lambda')
plt.ylabel('')
plt.xscale('log')
plt.legend();
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[78], line 1
----> 1␣
 ↪plt.plot(candidate_values, MAE_list,'o-', markersize = 5, label = 'Out-of-sample')
      2 plt.plot(candidate_values, TrainingMAE_list,'o-', markersize = 5, label␣
 ↪= 'In-sample')
      3 plt.xlabel('Value of lambda')

File /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
 ↪site-packages/matplotlib/pyplot.py:3794, in plot(scalex, scaley, data, *args,␣
 ↪**kwargs)
   3786 @_copy_docstring_and_deprecators(Axes.plot)
   3787 def plot(
   3788     *args: float | ArrayLike | str,
   (…)
   3792     **kwargs,
   3793 ) -> list[Line2D]:
-> 3794     return gca().plot(
   3795         *args,
   3796         scalex=scalex,
   3797         scaley=scaley,
   3798         **({"data": data} if data is not None else {}),
   3799         **kwargs,
   3800     )

File /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
 ↪site-packages/matplotlib/axes/_axes.py:1779, in Axes.plot(self, scalex,␣
 ↪scaley, data, *args, **kwargs)
```

```
   1536 """
   1537 Plot y versus x as lines and/or markers.
   1538
   (…)
   1776 (``'green'``) or hex strings (``'#008000'``).
   1777 """
   1778 kwargs = cbook.normalize_kwargs(kwargs, mlines.Line2D)
-> 1779 lines = [*self._get_lines(self, *args, data=data, **kwargs)]
   1780 for line in lines:
   1781     self.add_line(line)

File /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
↪site-packages/matplotlib/axes/_base.py:296, in _process_plot_var_args.
↪__call__(self, axes, data, *args, **kwargs)
    294     this += args[0],
    295     args = args[1:]
--> 296 yield from self._plot_args(
    297     axes, this, kwargs, ambiguous_fmt_datakey=ambiguous_fmt_datakey)

File /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
↪site-packages/matplotlib/axes/_base.py:486, in _process_plot_var_args.
↪_plot_args(self, axes, tup, kwargs, return_kwargs, ambiguous_fmt_datakey)
    483     axes.yaxis.update_units(y)
    485 if x.shape[0] != y.shape[0]:
--> 486     raise ValueError(f"x and y must have same first dimension, but "
    487                      f"have shapes {x.shape} and {y.shape}")
    488 if x.ndim > 2 or y.ndim > 2:
    489     raise ValueError(f"x and y can be no greater than 2D, but have "
    490                      f"shapes {x.shape} and {y.shape}")

ValueError: x and y must have same first dimension, but have shapes (6,) and␣
↪(12,)
```




### 1.3.1 K-fold cross validation

[ ]:
```
from sklearn.model_selection import GridSearchCV


alpha_grid = {'alpha': np.logspace(-8, -1, num=10, base=10)}

lasso_cv = GridSearchCV(lasso, param_grid = alpha_grid, scoring='neg_mean_squared_error', cv=10, verbose=1)
lasso_cv.fit(X_train_lasso, y_train)
```

```
Fitting 10 folds for each of 10 candidates, totalling 100 fits
```

[ ]:
```
GridSearchCV(cv=10, estimator=Lasso(alpha=0.1, random_state=88),
             param_grid={'alpha': array([1.00000000e-08, 5.99484250e-08,
3.59381366e-07, 2.15443469e-06,
       1.29154967e-05, 7.74263683e-05, 4.64158883e-04, 2.78255940e-03,
       1.66810054e-02, 1.00000000e-01])},
             scoring='neg_mean_squared_error', verbose=1)
```

[ ]:
```
lasso_cv.cv_results_
```

```
[ ]: {'mean_fit_time': array([0.74864337, 0.74456458, 0.75606918, 0.73773813,
0.74517162,
        0.62181613, 0.51850934, 0.35173864, 0.2392029 , 0.24281828]),
 'std_fit_time': array([0.04160104, 0.05409118, 0.07124824, 0.03803379,
0.05941938,
        0.00888817, 0.01800875, 0.01517724, 0.01328081, 0.0157893 ]),
 'mean_score_time': array([0.00316212, 0.00462024, 0.00691543, 0.00376642,
0.00333533,
        0.00310757, 0.0040307 , 0.00296528, 0.00315723, 0.00292087]),
 'std_score_time': array([0.00066263, 0.00237882, 0.00603587, 0.00131657,
0.00151474,
        0.00073947, 0.00188376, 0.00093391, 0.00169379, 0.00095648]),
 'param_alpha': masked_array(data=[1e-08, 5.99484250318941e-08,
3.5938136638046254e-07,
                    2.1544346900318865e-06, 1.2915496650148827e-05,
                    7.742636826811278e-05, 0.0004641588833612782,
                    0.0027825594022071257, 0.016681005372000592, 0.1],
              mask=[False, False, False, False, False, False, False, False,
                    False, False],
        fill_value=1e+20),
 'params': [{'alpha': 1e-08},
  {'alpha': 5.99484250318941e-08},
  {'alpha': 3.5938136638046254e-07},
  {'alpha': 2.1544346900318865e-06},
  {'alpha': 1.2915496650148827e-05},
  {'alpha': 7.742636826811278e-05},
  {'alpha': 0.0004641588833612782},
  {'alpha': 0.0027825594022071257},
  {'alpha': 0.016681005372000592},
  {'alpha': 0.1}],
 'split0_test_score': array([-0.00787318, -0.00787117, -0.00785983, -0.00779085,
-0.00757535,
        -0.00735782, -0.00761434, -0.0105296 , -0.01173297, -0.01166006]),
 'split1_test_score': array([-0.01815299, -0.01778185, -0.01597704, -0.01470858,
-0.01397911,
        -0.01117959, -0.01165344, -0.01702969, -0.02329114, -0.02158746]),
 'split2_test_score': array([-0.0254296 , -0.02436969, -0.02344383, -0.02317906,
-0.02275288,
        -0.02615171, -0.02970623, -0.03616987, -0.04693504, -0.04688338]),
 'split3_test_score': array([-0.00790345, -0.00787734, -0.00787569, -0.00787901,
-0.00800425,
        -0.0079026 , -0.00748859, -0.0101973 , -0.01401935, -0.01400183]),
 'split4_test_score': array([-0.02904953, -0.02900245, -0.0287643 , -0.02792024,
-0.0238768 ,
        -0.00986928, -0.01259665, -0.01794151, -0.03566135, -0.03559733]),
 'split5_test_score': array([-0.0155111 , -0.01513605, -0.01498391, -0.01480683,
-0.01465653,
```

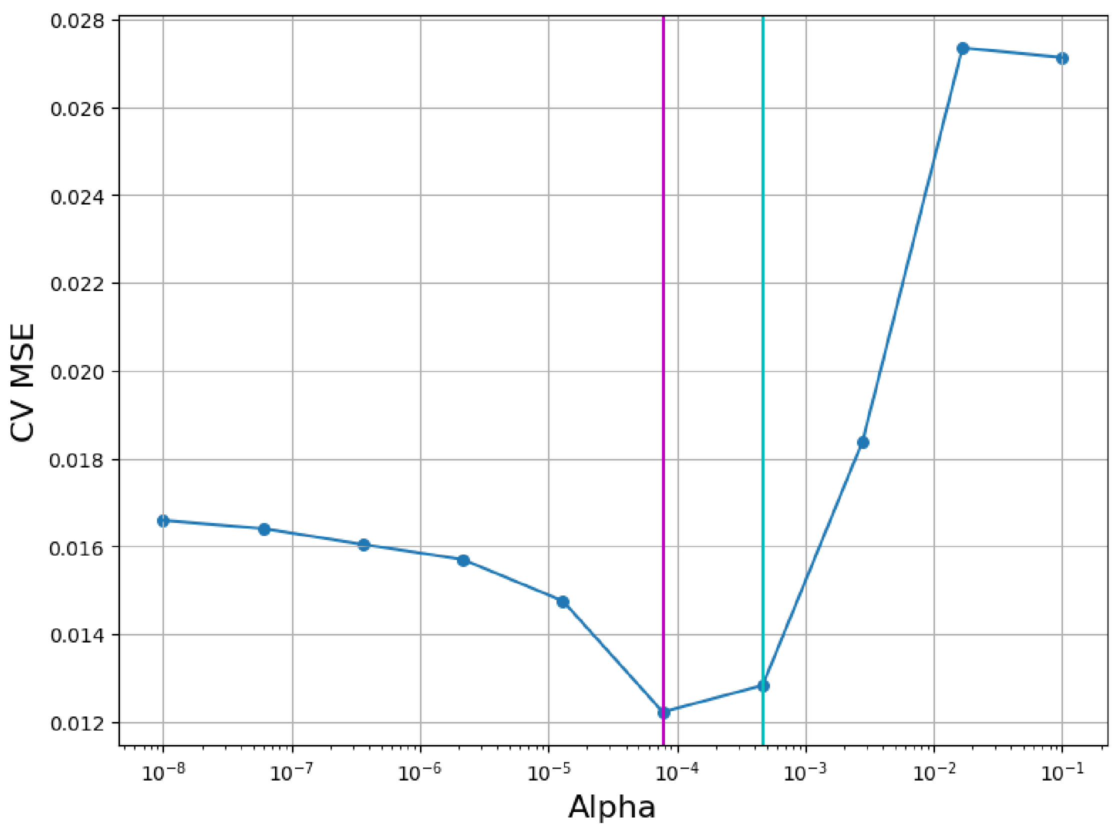
```
        -0.01424154, -0.01478704, -0.01855934, -0.02740716, -0.02776815]),
 'split6_test_score': array([-0.01502746, -0.01501102, -0.01497681, -0.01488101,
-0.01448217,
        -0.01287819, -0.01325205, -0.02035065, -0.03199917, -0.03186222]),
 'split7_test_score': array([-0.008832  , -0.00873671, -0.00841158, -0.00825042,
-0.00775649,
        -0.00641255, -0.00612379, -0.00853876, -0.01248738, -0.01208519]),
 'split8_test_score': array([-0.01122054, -0.01138903, -0.01146731, -0.01147103,
-0.01126702,
        -0.01001889, -0.01098486, -0.01832119, -0.03018501, -0.03030998]),
 'split9_test_score': array([-0.0268892 , -0.02685823, -0.02667113, -0.02610903,
-0.0232181 ,
        -0.0162867 , -0.01412417, -0.02621292, -0.03973004, -0.03953425]),
 'mean_test_score': array([-0.0165889 , -0.01640335, -0.01604314, -0.01569961,
-0.01475687,
        -0.01222989, -0.01283312, -0.01838508, -0.02734486, -0.02712898]),
 'std_test_score': array([0.00765478, 0.0075257 , 0.00738017, 0.00716361,
0.00614612,
        0.005501  , 0.00629317, 0.00780515, 0.0113778 , 0.01147782]),
 'rank_test_score': array([ 7,  6,  5,  4,  3,  1,  2,  8, 10,  9],
dtype=int32)}
```

[ ]:

```python
from sklearn.model_selection import GridSearchCV

def one_standard_error_rule(model, results, param_grid, n_splits,␣
 ↪neg_mean_squared_error=True):

    assert neg_mean_squared_error == True # function is defined specifically␣
 ↪for neg_mean_squared_error

    range_x = param_grid # results['param_'+list(param_grid.keys())[0]].data
    std_vs_x  = pd.Series(results['std_test_score'], index = range_x)
    sem_vs_x  = std_vs_x/np.sqrt(n_splits)

    mean_vs_x = pd.Series(results['mean_test_score'], index = range_x)
    mean_vs_x = mean_vs_x*(-1)

    x_min = mean_vs_x.idxmin()
    sem = sem_vs_x[x_min]

    if (model=='pcr'):
        x_1se = mean_vs_x[mean_vs_x <= min(mean_vs_x) + sem].index.min()
    elif (model=='ridge') | (model=='lasso'):
        x_1se = mean_vs_x[mean_vs_x <= min(mean_vs_x) + sem].index.max()

    #x_1se_idx = int(np.argwhere(range_x == x_1se)[0])
```

```
    return x_min, x_1se
```

[ ]:
```
range_alpha = lasso_cv.cv_results_['param_alpha'].data
MSE_scores = lasso_cv.cv_results_['mean_test_score']*(-1)
x_min, x_1se = one_standard_error_rule(model='lasso',
                                       results=lasso_cv.cv_results_,
                                       param_grid=range_alpha,
                                       n_splits=10,
                                       neg_mean_squared_error=True)
plt.figure(figsize=(8, 6))
ax = plt.gca()
ax.set_xscale('log')
plt.xlabel('Alpha', fontsize=16)
plt.ylabel('CV MSE', fontsize=16)
plt.scatter(range_alpha, MSE_scores, s=30)
plt.plot(range_alpha, MSE_scores)
plt.axvline(x=x_min, color='m')
plt.axvline(x=x_1se, color='c')
plt.grid()

plt.tight_layout()
plt.show()
```



Magenta vertical line is the minimizer, the cyan vertical line is the "1 Standard Error" selection.

```
[ ]: acc = lasso_cv.cv_results_['mean_test_score'] # what sklearn calls␣
     ↪mean_test_score is the holdout set, i.e. the validation set.
     ccp = lasso_cv.cv_results_['param_alpha'].data

     pd.DataFrame({'ccp alpha' : ccp, 'Validation Accuracy': acc})
```

```
[ ]:       ccp alpha  Validation Accuracy
     0  1.000000e-08            -0.016589
     1  5.994843e-08            -0.016403
     2  3.593814e-07            -0.016043
     3  2.154435e-06            -0.015700
     4  1.291550e-05            -0.014757
     5  7.742637e-05            -0.012230
     6  4.641589e-04            -0.012833
     7  2.782559e-03            -0.018385
     8  1.668101e-02            -0.027345
     9  1.000000e-01            -0.027129
```

```
[ ]: print('Alpha one standard error rule:', x_1se)
```

```
Alpha one standard error rule: 0.0004641588833612782
```

### 1.3.2 Lasso Refit with One Standard Error Rule

```
[ ]: lasso_cv = GridSearchCV(lasso, {'alpha': [x_1se]},␣
     ↪scoring='neg_mean_squared_error', cv=10)
     lasso_cv.fit(X_train_lasso, y_train)

     print_metrics(lasso_cv, X_train_lasso, y_train, X_test_lasso, y_test,␣
     ↪flag_log_sale_price = True)
     print_metrics(lasso_cv, X_train_lasso, y_train, X_test_lasso, y_test,␣
     ↪flag_log_sale_price = False)
```

```
Metrics for Log(Sale Price):

Training R2 0.9467465562233576
Training MAE 0.06238524036959335
Training RMSE 0.08932411725332272
Out-of-sample R2 0.9007191016790816
Out-of-sample MAE 0.07862128213660351
Out-of-sample RMSE 0.1247955735851673

Metrics for Sale Price:
```

```
Training R2 0.9526034800011082
Training MAE 11250.968361557421
Training RMSE 17007.516334830016
Out-of-sample R2 0.9325057084639106
Out-of-sample MAE 13461.47428911071
Out-of-sample RMSE 20362.152823875873
```

### 1.4 Shuffle the dataset for k-fold cross validation

[ ]:
```
from sklearn.model_selection import KFold

alpha_grid = {'alpha': np.logspace(-8, -1, num=15, base=10)}
cv = KFold(n_splits = 10, random_state = 1, shuffle = True)
lasso_cv = GridSearchCV(lasso, param_grid = alpha_grid,␣
↪scoring='neg_mean_squared_error', cv=cv, verbose=2)
lasso_cv.fit(X_train_lasso, y_train)
```

```
Fitting 10 folds for each of 15 candidates, totalling 150 fits
[CV] END …alpha=1e-08; total time=   0.7s
[CV] END …alpha=1e-08; total time=   0.7s
[CV] END …alpha=1e-08; total time=   0.7s
[CV] END …alpha=1e-08; total time=   0.7s
[CV] END …alpha=1e-08; total time=   0.8s
[CV] END …alpha=1e-08; total time=   0.8s
[CV] END …alpha=1e-08; total time=   0.8s
[CV] END …alpha=1e-08; total time=   0.8s
[CV] END …alpha=1e-08; total time=   0.7s
[CV] END …alpha=1e-08; total time=   0.7s
[CV] END …alpha=3.162277660168379e-08; total time=   0.7s
[CV] END …alpha=3.162277660168379e-08; total time=   0.8s
[CV] END …alpha=3.162277660168379e-08; total time=   0.7s
[CV] END …alpha=3.162277660168379e-08; total time=   0.7s
[CV] END …alpha=3.162277660168379e-08; total time=   0.8s
[CV] END …alpha=3.162277660168379e-08; total time=   0.8s
[CV] END …alpha=3.162277660168379e-08; total time=   0.8s
[CV] END …alpha=3.162277660168379e-08; total time=   0.8s
[CV] END …alpha=3.162277660168379e-08; total time=   0.7s
[CV] END …alpha=3.162277660168379e-08; total time=   0.8s
[CV] END …alpha=1e-07; total time=   0.7s
[CV] END …alpha=1e-07; total time=   0.7s
[CV] END …alpha=1e-07; total time=   0.7s
[CV] END …alpha=1e-07; total time=   0.7s
[CV] END …alpha=1e-07; total time=   0.7s
[CV] END …alpha=1e-07; total time=   0.7s
[CV] END …alpha=1e-07; total time=   0.7s
[CV] END …alpha=1e-07; total time=   0.8s
[CV] END …alpha=1e-07; total time=   0.7s
[CV] END …alpha=1e-07; total time=   0.7s
```

```
[CV] END …alpha=3.162277660168379e-07; total time=   0.8s
[CV] END …alpha=3.162277660168379e-07; total time=   0.7s
[CV] END …alpha=3.162277660168379e-07; total time=   0.7s
[CV] END …alpha=3.162277660168379e-07; total time=   0.7s
[CV] END …alpha=3.162277660168379e-07; total time=   0.7s
[CV] END …alpha=3.162277660168379e-07; total time=   0.7s
[CV] END …alpha=3.162277660168379e-07; total time=   0.7s
[CV] END …alpha=3.162277660168379e-07; total time=   0.8s
[CV] END …alpha=3.162277660168379e-07; total time=   0.8s
[CV] END …alpha=3.162277660168379e-07; total time=   0.8s
[CV] END …alpha=1e-06; total time=   0.7s
[CV] END …alpha=1e-06; total time=   0.8s
[CV] END …alpha=1e-06; total time=   0.8s
[CV] END …alpha=1e-06; total time=   0.8s
[CV] END …alpha=1e-06; total time=   0.8s
[CV] END …alpha=1e-06; total time=   0.8s
[CV] END …alpha=1e-06; total time=   0.7s
[CV] END …alpha=1e-06; total time=   0.7s
[CV] END …alpha=1e-06; total time=   0.7s
[CV] END …alpha=1e-06; total time=   0.9s
[CV] END …alpha=3.162277660168379e-06; total time=   0.8s
[CV] END …alpha=3.162277660168379e-06; total time=   0.8s
[CV] END …alpha=3.162277660168379e-06; total time=   0.8s
[CV] END …alpha=3.162277660168379e-06; total time=   0.7s
[CV] END …alpha=3.162277660168379e-06; total time=   0.7s
[CV] END …alpha=3.162277660168379e-06; total time=   0.7s
[CV] END …alpha=3.162277660168379e-06; total time=   0.8s
[CV] END …alpha=3.162277660168379e-06; total time=   0.7s
[CV] END …alpha=3.162277660168379e-06; total time=   0.7s
[CV] END …alpha=3.162277660168379e-06; total time=   0.8s
[CV] END …alpha=1e-05; total time=   0.8s
[CV] END …alpha=1e-05; total time=   0.8s
[CV] END …alpha=1e-05; total time=   0.8s
[CV] END …alpha=1e-05; total time=   0.7s
[CV] END …alpha=1e-05; total time=   0.7s
[CV] END …alpha=1e-05; total time=   0.7s
[CV] END …alpha=1e-05; total time=   0.7s
[CV] END …alpha=1e-05; total time=   0.7s
[CV] END …alpha=1e-05; total time=   0.7s
[CV] END …alpha=1e-05; total time=   0.7s
[CV] END …alpha=3.1622776601683795e-05; total time=   0.7s
[CV] END …alpha=3.1622776601683795e-05; total time=   0.7s
[CV] END …alpha=3.1622776601683795e-05; total time=   0.7s
[CV] END …alpha=3.1622776601683795e-05; total time=   0.7s
[CV] END …alpha=3.1622776601683795e-05; total time=   0.7s
[CV] END …alpha=3.1622776601683795e-05; total time=   0.7s
[CV] END …alpha=3.1622776601683795e-05; total time=   0.7s
[CV] END …alpha=3.1622776601683795e-05; total time=   0.7s
```

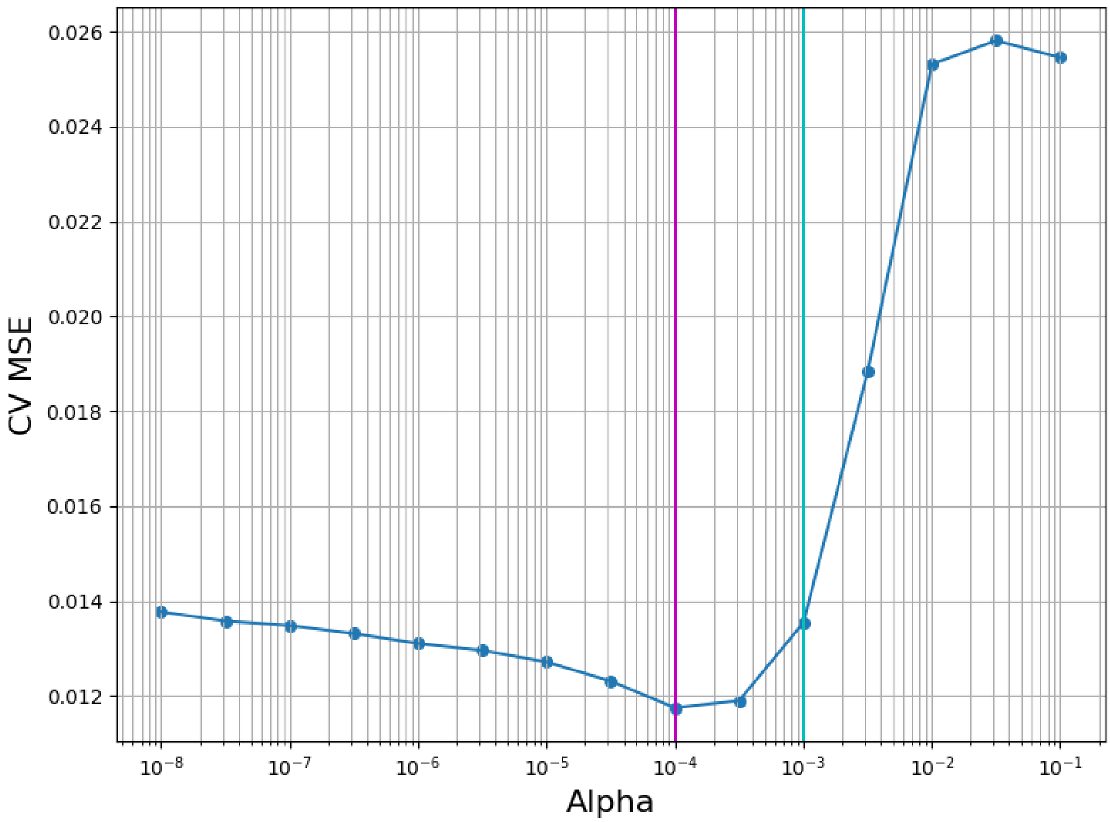
```
[CV] END …alpha=3.1622776601683795e-05; total time=   0.7s
[CV] END …alpha=3.1622776601683795e-05; total time=   0.7s
[CV] END …alpha=0.0001; total time=   0.6s
[CV] END …alpha=0.0001; total time=   0.6s
[CV] END …alpha=0.0001; total time=   0.6s
[CV] END …alpha=0.0001; total time=   0.6s
[CV] END …alpha=0.0001; total time=   0.6s
[CV] END …alpha=0.0001; total time=   0.6s
[CV] END …alpha=0.0001; total time=   0.6s
[CV] END …alpha=0.0001; total time=   0.6s
[CV] END …alpha=0.0001; total time=   0.6s
[CV] END …alpha=0.0001; total time=   0.6s
[CV] END …alpha=0.00031622776601683794; total time=   0.6s
[CV] END …alpha=0.00031622776601683794; total time=   0.6s
[CV] END …alpha=0.00031622776601683794; total time=   0.5s
[CV] END …alpha=0.00031622776601683794; total time=   0.5s
[CV] END …alpha=0.00031622776601683794; total time=   0.5s
[CV] END …alpha=0.00031622776601683794; total time=   0.6s
[CV] END …alpha=0.00031622776601683794; total time=   0.5s
[CV] END …alpha=0.00031622776601683794; total time=   0.5s
[CV] END …alpha=0.00031622776601683794; total time=   0.5s
[CV] END …alpha=0.00031622776601683794; total time=   0.5s
[CV] END …alpha=0.001; total time=   0.5s
[CV] END …alpha=0.001; total time=   0.5s
[CV] END …alpha=0.001; total time=   0.5s
[CV] END …alpha=0.001; total time=   0.5s
[CV] END …alpha=0.001; total time=   0.5s
[CV] END …alpha=0.001; total time=   0.5s
[CV] END …alpha=0.001; total time=   0.5s
[CV] END …alpha=0.001; total time=   0.4s
[CV] END …alpha=0.001; total time=   0.4s
[CV] END …alpha=0.001; total time=   0.4s
[CV] END …alpha=0.0031622776601683794; total time=   0.4s
[CV] END …alpha=0.0031622776601683794; total time=   0.3s
[CV] END …alpha=0.0031622776601683794; total time=   0.4s
[CV] END …alpha=0.0031622776601683794; total time=   0.4s
[CV] END …alpha=0.0031622776601683794; total time=   0.4s
[CV] END …alpha=0.0031622776601683794; total time=   0.3s
[CV] END …alpha=0.0031622776601683794; total time=   0.3s
[CV] END …alpha=0.0031622776601683794; total time=   0.4s
[CV] END …alpha=0.0031622776601683794; total time=   0.3s
[CV] END …alpha=0.0031622776601683794; total time=   0.4s
[CV] END …alpha=0.01; total time=   0.3s
[CV] END …alpha=0.01; total time=   0.3s
[CV] END …alpha=0.01; total time=   0.4s
[CV] END …alpha=0.01; total time=   0.4s
[CV] END …alpha=0.01; total time=   0.4s
[CV] END …alpha=0.01; total time=   0.4s
```

```
[CV] END …alpha=0.01; total time=   0.4s
[CV] END …alpha=0.01; total time=   0.3s
[CV] END …alpha=0.01; total time=   0.3s
[CV] END …alpha=0.01; total time=   0.3s
[CV] END …alpha=0.03162277660168379; total time=   0.3s
[CV] END …alpha=0.03162277660168379; total time=   0.3s
[CV] END …alpha=0.03162277660168379; total time=   0.3s
[CV] END …alpha=0.03162277660168379; total time=   0.3s
[CV] END …alpha=0.03162277660168379; total time=   0.3s
[CV] END …alpha=0.03162277660168379; total time=   0.3s
[CV] END …alpha=0.03162277660168379; total time=   0.3s
[CV] END …alpha=0.03162277660168379; total time=   0.2s
[CV] END …alpha=0.03162277660168379; total time=   0.3s
[CV] END …alpha=0.03162277660168379; total time=   0.3s
[CV] END …alpha=0.1; total time=   0.3s
[CV] END …alpha=0.1; total time=   0.3s
[CV] END …alpha=0.1; total time=   0.3s
[CV] END …alpha=0.1; total time=   0.3s
[CV] END …alpha=0.1; total time=   0.3s
[CV] END …alpha=0.1; total time=   0.3s
[CV] END …alpha=0.1; total time=   0.3s
[CV] END …alpha=0.1; total time=   0.3s
[CV] END …alpha=0.1; total time=   0.3s
[CV] END …alpha=0.1; total time=   0.3s
```

[ ]:
```
GridSearchCV(cv=KFold(n_splits=10, random_state=1, shuffle=True),
             estimator=Lasso(alpha=0.1, random_state=88),
             param_grid={'alpha': array([1.00000000e-08, 3.16227766e-08,
1.00000000e-07, 3.16227766e-07,
       1.00000000e-06, 3.16227766e-06, 1.00000000e-05, 3.16227766e-05,
       1.00000000e-04, 3.16227766e-04, 1.00000000e-03, 3.16227766e-03,
       1.00000000e-02, 3.16227766e-02, 1.00000000e-01])},
             scoring='neg_mean_squared_error', verbose=2)
```

[ ]:
```python
range_alpha = lasso_cv.cv_results_['param_alpha'].data
MSE_scores = lasso_cv.cv_results_['mean_test_score']*(-1)
x_min, x_1se = one_standard_error_rule(model='lasso',
                                       results=lasso_cv.cv_results_,
                                       param_grid=range_alpha,
                                       n_splits=10,
                                       neg_mean_squared_error=True)
plt.figure(figsize=(8, 6))
ax = plt.gca()
ax.set_xscale('log')
plt.xlabel('Alpha', fontsize=16)
plt.ylabel('CV MSE', fontsize=16)
plt.scatter(range_alpha, MSE_scores, s=30)
```

```
plt.plot(range_alpha, MSE_scores)
plt.axvline(x=x_min, color='m')
plt.axvline(x=x_1se, color='c')
plt.grid(True, which='both')

plt.tight_layout()
plt.show()
```




## 1.5 Custom loss function

[ ]:
```
def large_prediction_error_count(y_test, y_pred, threshold = 2000):
    y_pred_test = pd.Series(y_pred).copy().apply(np.exp).reset_index(drop=True)
    y_test_exp = pd.Series(y_test).copy().apply(np.exp).reset_index(drop=True)
    count = [(y_pred_test - y_test_exp) > threshold]
    return np.sum(count)
```

[ ]:
```
from sklearn.metrics import make_scorer

alpha_grid = {'alpha': np.logspace(-8, -1, num=10, base=10)}
cv = KFold(n_splits = 10, random_state = 1, shuffle = True)
```

```
lasso_cv = GridSearchCV(lasso, param_grid = alpha_grid,␣
↪scoring=make_scorer(large_prediction_error_count, greater_is_better=False),␣
↪cv=cv, verbose=2)
lasso_cv.fit(X_train_lasso, y_train)
```

```
Fitting 10 folds for each of 10 candidates, totalling 100 fits
[CV] END …alpha=1e-08; total time=   0.7s
[CV] END …alpha=1e-08; total time=   0.7s
[CV] END …alpha=1e-08; total time=   0.7s
[CV] END …alpha=1e-08; total time=   0.7s
[CV] END …alpha=1e-08; total time=   0.7s
[CV] END …alpha=1e-08; total time=   0.8s
[CV] END …alpha=1e-08; total time=   0.7s
[CV] END …alpha=1e-08; total time=   0.8s
[CV] END …alpha=1e-08; total time=   0.7s
[CV] END …alpha=1e-08; total time=   0.7s
[CV] END …alpha=5.99484250318941e-08; total time=   0.7s
[CV] END …alpha=5.99484250318941e-08; total time=   0.7s
[CV] END …alpha=5.99484250318941e-08; total time=   0.7s
[CV] END …alpha=5.99484250318941e-08; total time=   0.8s
[CV] END …alpha=5.99484250318941e-08; total time=   0.7s
[CV] END …alpha=5.99484250318941e-08; total time=   0.7s
[CV] END …alpha=5.99484250318941e-08; total time=   0.7s
[CV] END …alpha=5.99484250318941e-08; total time=   0.8s
[CV] END …alpha=5.99484250318941e-08; total time=   0.7s
[CV] END …alpha=5.99484250318941e-08; total time=   0.7s
[CV] END …alpha=3.593813663804626e-07; total time=   0.7s
[CV] END …alpha=3.593813663804626e-07; total time=   0.7s
[CV] END …alpha=3.593813663804626e-07; total time=   0.7s
[CV] END …alpha=3.593813663804626e-07; total time=   0.7s
[CV] END …alpha=3.593813663804626e-07; total time=   0.7s
[CV] END …alpha=3.593813663804626e-07; total time=   0.7s
[CV] END …alpha=3.593813663804626e-07; total time=   0.7s
[CV] END …alpha=3.593813663804626e-07; total time=   0.7s
[CV] END …alpha=3.593813663804626e-07; total time=   0.7s
[CV] END …alpha=3.593813663804626e-07; total time=   0.7s
[CV] END …alpha=2.1544346900318865e-06; total time=   0.8s
[CV] END …alpha=2.1544346900318865e-06; total time=   0.8s
[CV] END …alpha=2.1544346900318865e-06; total time=   0.8s
[CV] END …alpha=2.1544346900318865e-06; total time=   0.7s
[CV] END …alpha=2.1544346900318865e-06; total time=   0.7s
[CV] END …alpha=2.1544346900318865e-06; total time=   0.8s
[CV] END …alpha=2.1544346900318865e-06; total time=   0.7s
[CV] END …alpha=2.1544346900318865e-06; total time=   0.7s
[CV] END …alpha=2.1544346900318865e-06; total time=   0.7s
[CV] END …alpha=2.1544346900318865e-06; total time=   0.7s
[CV] END …alpha=1.2915496650148827e-05; total time=   0.7s
[CV] END …alpha=1.2915496650148827e-05; total time=   0.7s
```

```
[CV] END …alpha=1.2915496650148827e-05; total time=   0.7s
[CV] END …alpha=1.2915496650148827e-05; total time=   0.7s
[CV] END …alpha=1.2915496650148827e-05; total time=   0.8s
[CV] END …alpha=1.2915496650148827e-05; total time=   0.7s
[CV] END …alpha=1.2915496650148827e-05; total time=   0.7s
[CV] END …alpha=1.2915496650148827e-05; total time=   0.7s
[CV] END …alpha=1.2915496650148827e-05; total time=   0.7s
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[89], line 6
      4 cv = KFold(n_splits = 10, random_state = 1, shuffle = True)
      5 lasso_cv = GridSearchCV(lasso, param_grid = alpha_grid,␣
 ↪scoring=make_scorer(large_prediction_error_count, greater_is_better=False),␣
 ↪cv=cv, verbose=2)
----> 6 lasso_cv.fit(X_train_lasso, y_train)

File /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
 ↪site-packages/sklearn/base.py:1473, in _fit_context.<locals>.decorator.
 ↪<locals>.wrapper(estimator, *args, **kwargs)
   1466     estimator._validate_params()
   1468 with config_context(
   1469     skip_parameter_validation=(
   1470         prefer_skip_nested_validation or global_skip_validation
   1471     )
   1472 ):
-> 1473     return fit_method(estimator, *args, **kwargs)

File /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
 ↪site-packages/sklearn/model_selection/_search.py:1019, in BaseSearchCV.
 ↪fit(self, X, y, **params)
   1013     results = self._format_results(
   1014         all_candidate_params, n_splits, all_out, all_more_results
   1015     )
   1017     return results
-> 1019 self._run_search(evaluate_candidates)
   1021 # multimetric is determined here because in the case of a callable
   1022 # self.scoring the return type is only known after calling
   1023 first_test_score = all_out[0]["test_scores"]

File /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
 ↪site-packages/sklearn/model_selection/_search.py:1573, in GridSearchCV.
 ↪_run_search(self, evaluate_candidates)
   1571 def _run_search(self, evaluate_candidates):
   1572     """Search all candidates in param_grid"""
-> 1573     evaluate_candidates(ParameterGrid(self.param_grid))
```

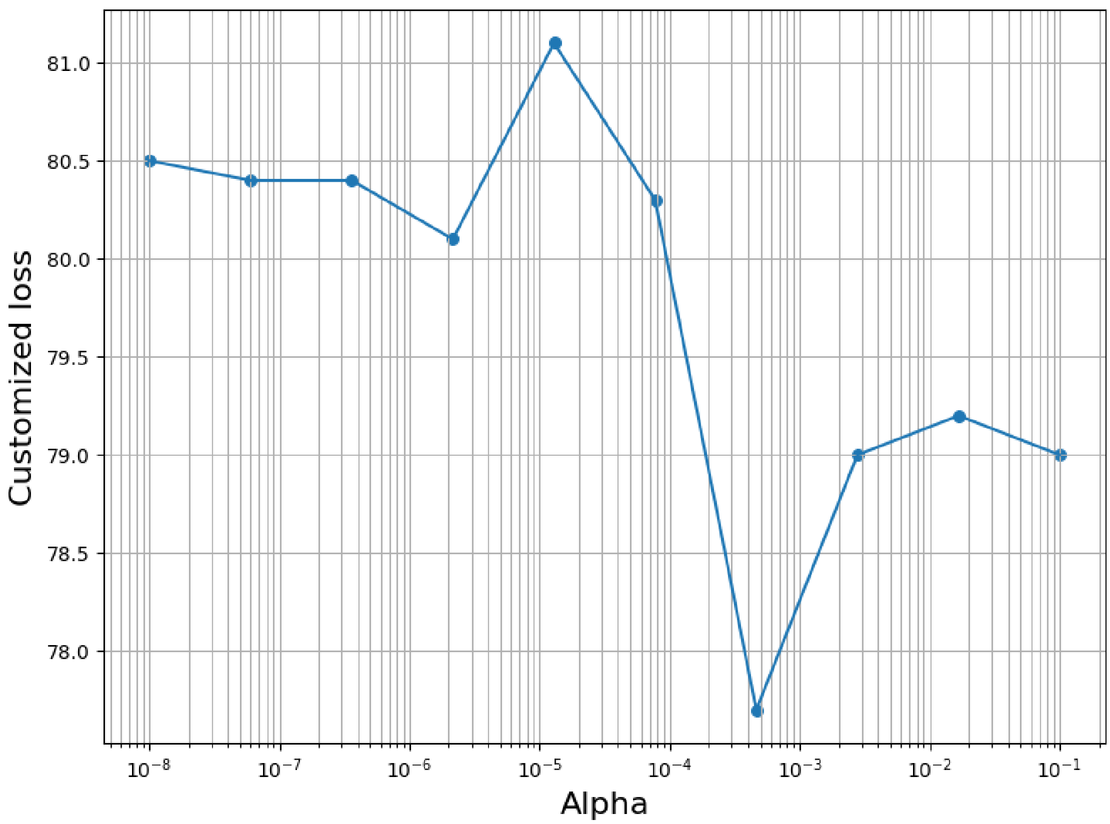
```
File /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
↪site-packages/sklearn/model_selection/_search.py:965, in BaseSearchCV.fit.
↪<locals>.evaluate_candidates(candidate_params, cv, more_results)
    957 if self.verbose > 0:
    958     print(
    959         "Fitting {0} folds for each of {1} candidates,"
    960         " totalling {2} fits".format(
    961             n_splits, n_candidates, n_candidates * n_splits
    962         )
    963     )
--> 965 out = parallel(
    966     delayed(_fit_and_score)(
    967         clone(base_estimator),
    968         X,
    969         y,
    970         train=train,
    971         test=test,
    972         parameters=parameters,
    973         split_progress=(split_idx, n_splits),
    974         candidate_progress=(cand_idx, n_candidates),
    975         **fit_and_score_kwargs,
    976     )
    977     for (cand_idx, parameters), (split_idx, (train, test)) in product(
    978         enumerate(candidate_params),
    979         enumerate(cv.split(X, y, **routed_params.splitter.split)),
    980     )
    981 )
    983 if len(out) < 1:
    984     raise ValueError(
    985         "No fits were performed. "
    986         "Was the CV iterator empty? "
    987         "Were there no candidates?"
    988     )

File /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
↪site-packages/sklearn/utils/parallel.py:74, in Parallel.__call__(self,␣
↪iterable)
     69 config = get_config()
     70 iterable_with_config = (
     71     (_with_config(delayed_func, config), args, kwargs)
     72     for delayed_func, args, kwargs in iterable
     73 )
---> 74 return super().__call__(iterable_with_config)

File /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
↪site-packages/joblib/parallel.py:1918, in Parallel.__call__(self, iterable)
   1916     output = self._get_sequential_output(iterable)
   1917     next(output)
```

```
-> 1918     return output if self.return_generator else list(output)
   1920 # Let's create an ID that uniquely identifies the current call. If the
   1921 # call is interrupted early and that the same instance is immediately
   1922 # re-used, this id will be used to prevent workers that were
   1923 # concurrently finalizing a task from the previous call to run the
   1924 # callback.
   1925 with self._lock:

File /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages/joblib/parallel.py:1847, in Parallel._get_sequential_output(self, iterable)
   1845 self.n_dispatched_batches += 1
   1846 self.n_dispatched_tasks += 1
-> 1847 res = func(*args, **kwargs)
   1848 self.n_completed_tasks += 1
   1849 self.print_progress()

File /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages/sklearn/utils/parallel.py:136, in _FuncWrapper.__call__(self, *args, **kwargs)
    134     config = {}
    135 with config_context(**config):
--> 136     return self.function(*args, **kwargs)

File /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages/sklearn/model_selection/_validation.py:888, in _fit_and_score(estimator, X, y, scorer, train, test, verbose, parameters, fit_params, score_params, return_train_score, return_parameters, return_n_test_samples, return_times, return_estimator, split_progress, candidate_progress, error_score)
    886         estimator.fit(X_train, **fit_params)
    887     else:
--> 888         estimator.fit(X_train, y_train, **fit_params)
    890 except Exception:
    891     # Note fit time as time until error
    892     fit_time = time.time() - start_time

File /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages/sklearn/base.py:1473, in _fit_context.<locals>.decorator.<locals>.wrapper(estimator, *args, **kwargs)
   1466     estimator._validate_params()
   1468 with config_context(
   1469     skip_parameter_validation=(
   1470         prefer_skip_nested_validation or global_skip_validation
   1471     )
   1472 ):
-> 1473     return fit_method(estimator, *args, **kwargs)
```

```
File /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
↪site-packages/sklearn/linear_model/_coordinate_descent.py:1077, in ElasticNet
↪fit(self, X, y, sample_weight, check_input)
   1075 else:
   1076     this_Xy = None
-> 1077 _, this_coef, this_dual_gap, this_iter = self.path(
   1078     X,
   1079     y[:, k],
   1080     l1_ratio=self.l1_ratio,
   1081     eps=None,
   1082     n_alphas=None,
   1083     alphas=[alpha],
   1084     precompute=precompute,
   1085     Xy=this_Xy,
   1086     copy_X=True,
   1087     coef_init=coef_[k],
   1088     verbose=False,
   1089     return_n_iter=True,
   1090     positive=self.positive,
   1091     check_input=False,
   1092     # from here on **params
   1093     tol=self.tol,
   1094     X_offset=X_offset,
   1095     X_scale=X_scale,
   1096     max_iter=self.max_iter,
   1097     random_state=self.random_state,
   1098     selection=self.selection,
   1099     sample_weight=sample_weight,
   1100 )
   1101 coef_[k] = this_coef[:, 0]
   1102 dual_gaps_[k] = this_dual_gap[0]

File /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
↪site-packages/sklearn/utils/_param_validation.py:186, in validate_params.
↪<locals>.decorator.<locals>.wrapper(*args, **kwargs)
    184 global_skip_validation = get_config()["skip_parameter_validation"]
    185 if global_skip_validation:
--> 186     return func(*args, **kwargs)
    188 func_sig = signature(func)
    190 # Map *args/**kwargs to the function signature

File /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
↪site-packages/sklearn/linear_model/_coordinate_descent.py:697, in enet_path(X, ␣
↪y, l1_ratio, eps, n_alphas, alphas, precompute, Xy, copy_X, coef_init,␣
↪verbose, return_n_iter, positive, check_input, **params)
    683     model = cd_fast.enet_coordinate_descent_gram(
    684         coef_,
    685         l1_reg,
   (…)
```

```
    694         positive,
    695     )
    696 elif precompute is False:
--> 697     model = cd_fast.enet_coordinate_descent(
    698         coef_, l1_reg, l2_reg, X, y, max_iter, tol, rng, random, positive
    699     )
    700 else:
    701     raise ValueError(
    702         "Precompute should be one of True, False, 'auto' or array-like.
↪Got %r"
    703         % precompute
    704     )

File _cd_fast.pyx:262, in sklearn.linear_model._cd_fast.enet_coordinate_descent()

File ~/Library/Python/3.12/lib/python/site-packages/numpy/core/getlimits.py:484
↪in finfo.__new__(cls, dtype)
    380 """
    381 finfo(dtype)
    382
   (…)
    479
    480 """
    482 _finfo_cache = {}
--> 484 def __new__(cls, dtype):
    485     try:
    486         obj = cls._finfo_cache.get(dtype)  # most common path

KeyboardInterrupt:
```

[ ]:
```python
range_alpha = lasso_cv.cv_results_['param_alpha'].data
new_scores = lasso_cv.cv_results_['mean_test_score']*(-1)

plt.figure(figsize=(8, 6))
ax = plt.gca()
ax.set_xscale('log')
plt.xlabel('Alpha', fontsize=16)
plt.ylabel('Customized loss', fontsize=16)
plt.scatter(range_alpha, new_scores, s=30)
plt.plot(range_alpha, new_scores)
plt.grid(True, which='both')

plt.tight_layout()
plt.show()
```




[ ]:

```

```

### 1.6 Cross validation for Principal Components Regression

[ ]:

```
y_train = ames_train['LogSalePrice']
y_test = ames_test['LogSalePrice']

X_train_pcr = X_train_poly_wide
X_test_pcr = X_test_poly_wide

print(X_train_poly_wide.shape, X_train_pcr.shape)
print(X_test_poly_wide.shape, X_test_pcr.shape)
```

```
(1828, 397) (1828, 397)
(937, 397) (937, 397)
```

We also standardize the data before feeding it to the PCA step, as recommended by good practice.

[ ]:

```
from sklearn.pipeline import Pipeline
scaler = StandardScaler()
pca = PCA(random_state=88)
```

```
lr = LinearRegression()
pipe = Pipeline(steps=[('scaler', scaler), ('pca', pca), ('lr', lr)])
```

Basic PCR

[ ]:
```
pipe.set_params(pca__n_components=5)
pipe.fit(X_train_pcr, y_train)
print_metrics(pipe, X_train_pcr, y_train, X_test_pcr, y_test,␣
 ↪flag_log_sale_price = True)
print_metrics(pipe, X_train_pcr, y_train, X_test_pcr, y_test,␣
 ↪flag_log_sale_price = False)
```

```
Metrics for Log(Sale Price):

Training R2 0.8669860382703731
Training MAE 0.09962977875968801
Training RMSE 0.14117035228048616
Out-of-sample R2 0.8396729663435233
Out-of-sample MAE 0.10399547659855168
Out-of-sample RMSE 0.15858777688396108

Metrics for Sale Price:

Training R2 0.897132348091005
Training MAE 17529.924419135274
Training RMSE 25055.73125787469
Out-of-sample R2 0.9018033283445104
Out-of-sample MAE 17377.607979840453
Out-of-sample RMSE 24560.576408359433
```

# 2 In-class activity 2: For PCR, do a 5 fold cross validation value for `n_components` in terms of the `R-squared`. The potential `n_components` is between 1 and 300.

- What are the best R2 value and its corresponding n_components?
- What is the value of `n_components` according to the one standard error rule?
- Refit the model using the `n_components` selected by the one standard error rule.

[ ]:
```
param_grid = {'pca__n_components': np.linspace(1, 300, 35).astype('int')}

pcr_cv = GridSearchCV(pipe, # model
                      param_grid, # candidate value
                      scoring='r2',
                      cv=5,
                     verbose = 2)
pcr_cv.fit(X_train_pcr, y_train)
```

```
Fitting 5 folds for each of 35 candidates, totalling 175 fits
[CV] END …pca__n_components=1; total time=   0.1s
[CV] END …pca__n_components=1; total time=   0.1s
[CV] END …pca__n_components=1; total time=   0.0s
[CV] END …pca__n_components=1; total time=   0.1s
[CV] END …pca__n_components=1; total time=   0.1s
[CV] END …pca__n_components=9; total time=   0.1s
[CV] END …pca__n_components=9; total time=   0.2s
[CV] END …pca__n_components=9; total time=   0.1s
[CV] END …pca__n_components=9; total time=   0.1s
[CV] END …pca__n_components=9; total time=   0.1s
[CV] END …pca__n_components=18; total time=   0.1s
[CV] END …pca__n_components=18; total time=   0.0s
[CV] END …pca__n_components=18; total time=   0.1s
[CV] END …pca__n_components=18; total time=   0.1s
[CV] END …pca__n_components=18; total time=   0.1s
[CV] END …pca__n_components=27; total time=   0.1s
[CV] END …pca__n_components=27; total time=   0.1s
[CV] END …pca__n_components=27; total time=   0.1s
[CV] END …pca__n_components=27; total time=   0.2s
[CV] END …pca__n_components=27; total time=   0.1s
[CV] END …pca__n_components=36; total time=   0.1s
[CV] END …pca__n_components=36; total time=   0.1s
[CV] END …pca__n_components=36; total time=   0.1s
[CV] END …pca__n_components=36; total time=   0.1s
[CV] END …pca__n_components=36; total time=   0.1s
[CV] END …pca__n_components=44; total time=   0.1s
[CV] END …pca__n_components=44; total time=   0.1s
[CV] END …pca__n_components=44; total time=   0.1s
[CV] END …pca__n_components=44; total time=   0.1s
[CV] END …pca__n_components=44; total time=   0.1s
[CV] END …pca__n_components=53; total time=   0.1s
[CV] END …pca__n_components=53; total time=   0.1s
[CV] END …pca__n_components=53; total time=   0.2s
[CV] END …pca__n_components=53; total time=   0.1s
[CV] END …pca__n_components=53; total time=   0.1s
[CV] END …pca__n_components=62; total time=   0.2s
[CV] END …pca__n_components=62; total time=   0.1s
[CV] END …pca__n_components=62; total time=   0.1s
[CV] END …pca__n_components=62; total time=   0.1s
[CV] END …pca__n_components=62; total time=   0.1s
[CV] END …pca__n_components=71; total time=   0.1s
[CV] END …pca__n_components=71; total time=   0.1s
[CV] END …pca__n_components=71; total time=   0.1s
[CV] END …pca__n_components=71; total time=   0.1s
[CV] END …pca__n_components=71; total time=   0.1s
[CV] END …pca__n_components=80; total time=   0.1s
[CV] END …pca__n_components=80; total time=   0.1s
```

```
[CV] END …pca__n_components=80; total time=   0.1s
[CV] END …pca__n_components=80; total time=   0.1s
[CV] END …pca__n_components=80; total time=   0.1s
[CV] END …pca__n_components=88; total time=   0.1s
[CV] END …pca__n_components=88; total time=   0.1s
[CV] END …pca__n_components=88; total time=   0.1s
[CV] END …pca__n_components=88; total time=   0.1s
[CV] END …pca__n_components=88; total time=   0.1s
[CV] END …pca__n_components=97; total time=   0.1s
[CV] END …pca__n_components=97; total time=   0.1s
[CV] END …pca__n_components=97; total time=   0.1s
[CV] END …pca__n_components=97; total time=   0.1s
[CV] END …pca__n_components=97; total time=   0.1s
[CV] END …pca__n_components=106; total time=   0.1s
[CV] END …pca__n_components=106; total time=   0.1s
[CV] END …pca__n_components=106; total time=   0.1s
[CV] END …pca__n_components=106; total time=   0.1s
[CV] END …pca__n_components=106; total time=   0.2s
[CV] END …pca__n_components=115; total time=   0.1s
[CV] END …pca__n_components=115; total time=   0.2s
[CV] END …pca__n_components=115; total time=   0.2s
[CV] END …pca__n_components=115; total time=   0.2s
[CV] END …pca__n_components=115; total time=   0.2s
[CV] END …pca__n_components=124; total time=   0.1s
[CV] END …pca__n_components=124; total time=   0.2s
[CV] END …pca__n_components=124; total time=   0.1s
[CV] END …pca__n_components=124; total time=   0.1s
[CV] END …pca__n_components=124; total time=   0.1s
[CV] END …pca__n_components=132; total time=   0.1s
[CV] END …pca__n_components=132; total time=   0.1s
[CV] END …pca__n_components=132; total time=   0.2s
[CV] END …pca__n_components=132; total time=   0.1s
[CV] END …pca__n_components=132; total time=   0.1s
[CV] END …pca__n_components=141; total time=   0.2s
[CV] END …pca__n_components=141; total time=   0.2s
[CV] END …pca__n_components=141; total time=   0.1s
[CV] END …pca__n_components=141; total time=   0.2s
[CV] END …pca__n_components=141; total time=   0.1s
[CV] END …pca__n_components=150; total time=   0.2s
[CV] END …pca__n_components=150; total time=   0.1s
[CV] END …pca__n_components=150; total time=   0.1s
[CV] END …pca__n_components=150; total time=   0.2s
[CV] END …pca__n_components=150; total time=   0.1s
[CV] END …pca__n_components=159; total time=   0.2s
[CV] END …pca__n_components=159; total time=   0.1s
[CV] END …pca__n_components=159; total time=   0.2s
[CV] END …pca__n_components=159; total time=   0.2s
[CV] END …pca__n_components=159; total time=   0.2s
```

```
[CV] END …pca__n_components=168; total time=   0.1s
[CV] END …pca__n_components=168; total time=   0.2s
[CV] END …pca__n_components=168; total time=   0.2s
[CV] END …pca__n_components=168; total time=   0.1s
[CV] END …pca__n_components=168; total time=   0.2s
[CV] END …pca__n_components=176; total time=   0.2s
[CV] END …pca__n_components=176; total time=   0.4s
[CV] END …pca__n_components=176; total time=   0.3s
[CV] END …pca__n_components=176; total time=   0.1s
[CV] END …pca__n_components=176; total time=   0.3s
[CV] END …pca__n_components=185; total time=   0.2s
[CV] END …pca__n_components=185; total time=   0.2s
[CV] END …pca__n_components=185; total time=   0.3s
[CV] END …pca__n_components=185; total time=   0.2s
[CV] END …pca__n_components=185; total time=   0.2s
[CV] END …pca__n_components=194; total time=   0.3s
[CV] END …pca__n_components=194; total time=   0.3s
[CV] END …pca__n_components=194; total time=   0.2s
[CV] END …pca__n_components=194; total time=   0.2s
[CV] END …pca__n_components=194; total time=   0.2s
[CV] END …pca__n_components=203; total time=   0.2s
[CV] END …pca__n_components=203; total time=   0.2s
[CV] END …pca__n_components=203; total time=   0.2s
[CV] END …pca__n_components=203; total time=   0.5s
[CV] END …pca__n_components=203; total time=   1.1s
[CV] END …pca__n_components=212; total time=   0.4s
[CV] END …pca__n_components=212; total time=   0.3s
[CV] END …pca__n_components=212; total time=   0.3s
[CV] END …pca__n_components=212; total time=   0.3s
[CV] END …pca__n_components=212; total time=   0.3s
[CV] END …pca__n_components=220; total time=   0.2s
[CV] END …pca__n_components=220; total time=   0.2s
[CV] END …pca__n_components=220; total time=   0.3s
[CV] END …pca__n_components=220; total time=   0.2s
[CV] END …pca__n_components=220; total time=   0.2s
[CV] END …pca__n_components=229; total time=   0.3s
[CV] END …pca__n_components=229; total time=   0.2s
[CV] END …pca__n_components=229; total time=   0.2s
[CV] END …pca__n_components=229; total time=   0.3s
[CV] END …pca__n_components=229; total time=   0.2s
[CV] END …pca__n_components=238; total time=   0.3s
[CV] END …pca__n_components=238; total time=   0.2s
[CV] END …pca__n_components=238; total time=   0.3s
[CV] END …pca__n_components=238; total time=   0.3s
[CV] END …pca__n_components=238; total time=   0.3s
[CV] END …pca__n_components=247; total time=   0.3s
[CV] END …pca__n_components=247; total time=   0.2s
[CV] END …pca__n_components=247; total time=   0.3s
```

```
[CV] END …pca__n_components=247; total time=   0.2s
[CV] END …pca__n_components=247; total time=   0.2s
[CV] END …pca__n_components=256; total time=   0.2s
[CV] END …pca__n_components=256; total time=   0.3s
[CV] END …pca__n_components=256; total time=   0.2s
[CV] END …pca__n_components=256; total time=   0.3s
[CV] END …pca__n_components=256; total time=   0.3s
[CV] END …pca__n_components=264; total time=   0.3s
[CV] END …pca__n_components=264; total time=   0.7s
[CV] END …pca__n_components=264; total time=   0.4s
[CV] END …pca__n_components=264; total time=   0.3s
[CV] END …pca__n_components=264; total time=   0.3s
[CV] END …pca__n_components=273; total time=   0.3s
[CV] END …pca__n_components=273; total time=   0.4s
[CV] END …pca__n_components=273; total time=   0.3s
[CV] END …pca__n_components=273; total time=   0.3s
[CV] END …pca__n_components=273; total time=   0.4s
[CV] END …pca__n_components=282; total time=   0.4s
[CV] END …pca__n_components=282; total time=   0.3s
[CV] END …pca__n_components=282; total time=   0.3s
[CV] END …pca__n_components=282; total time=   0.2s
[CV] END …pca__n_components=282; total time=   0.2s
[CV] END …pca__n_components=291; total time=   0.3s
[CV] END …pca__n_components=291; total time=   0.5s
[CV] END …pca__n_components=291; total time=   0.3s
[CV] END …pca__n_components=291; total time=   0.2s
[CV] END …pca__n_components=291; total time=   0.3s
[CV] END …pca__n_components=300; total time=   0.3s
[CV] END …pca__n_components=300; total time=   0.2s
[CV] END …pca__n_components=300; total time=   0.4s
[CV] END …pca__n_components=300; total time=   0.2s
[CV] END …pca__n_components=300; total time=   0.3s
```

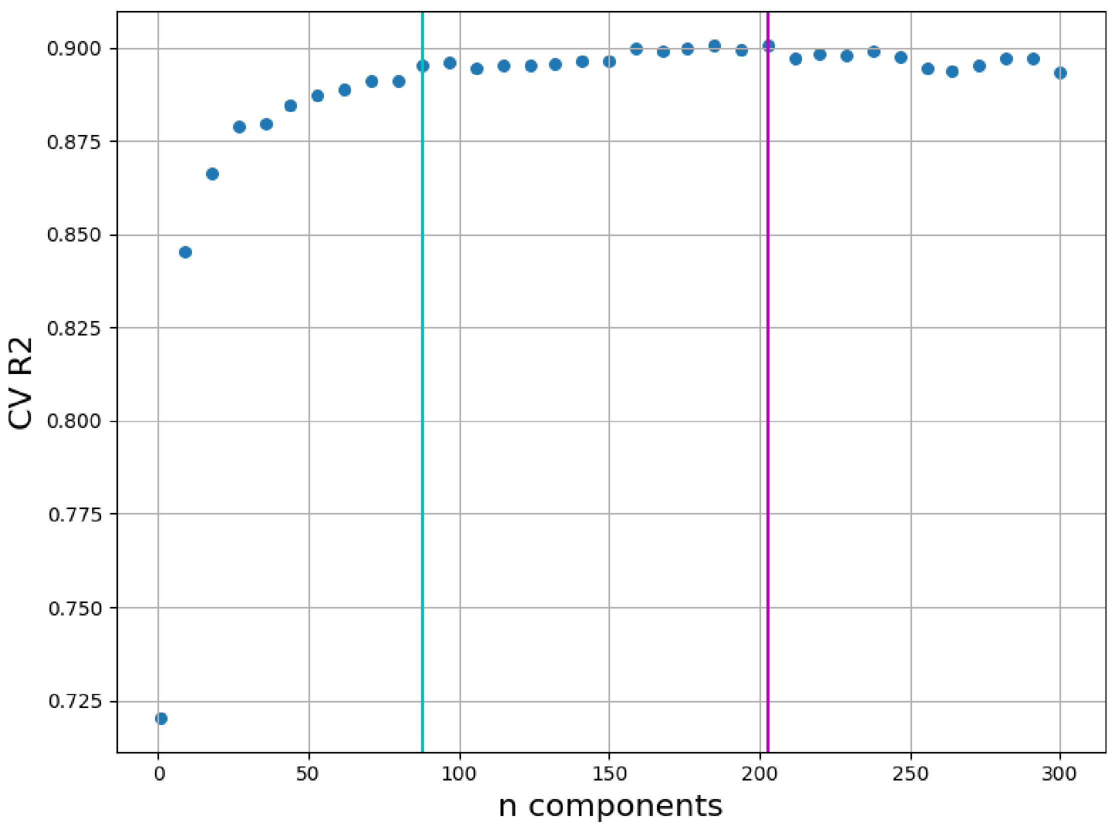
[ ]:

```
GridSearchCV(cv=5,
             estimator=Pipeline(steps=[('scaler', StandardScaler()),
                                       ('pca',
                                        PCA(n_components=5, random_state=88)),
                                       ('lr', LinearRegression())]),
             param_grid={'pca__n_components': array([  1,   9,  18,  27,  36,
44,  53,  62,  71,  80,  88,  97, 106,
       115, 124, 132, 141, 150, 159, 168, 176, 185, 194, 203, 212, 220,
       229, 238, 247, 256, 264, 273, 282, 291, 300])},
             scoring='r2', verbose=2)
```

[ ]:

```python
from scipy import stats

n_components = pcr_cv.cv_results_['param_pca__n_components'].data
```

```
R2_scores = pcr_cv.cv_results_['mean_test_score']
x_min, x_1se = one_standard_error_rule(model='pcr',
                                       results=pcr_cv.cv_results_,
                                       param_grid=n_components,
                                       n_splits=10,
                                       neg_mean_squared_error=True)

plt.figure(figsize=(8, 6))
plt.xlabel('n components', fontsize=16)
plt.ylabel('CV R2', fontsize=16)
plt.scatter(n_components, R2_scores, s=30)
plt.axvline(x=x_min, color='m')
plt.axvline(x=x_1se, color='c')
plt.grid(True, which='both')

plt.tight_layout()
plt.show()
```




[ ]:
```
best_r2_score = pcr_cv.best_score_
print("Best R2 Score from Cross-Validation:", best_r2_score)
```

```
best_n_components = pcr_cv.best_params_['pca__n_components']
print("Best n_components value:", best_n_components)
```

```
Best R2 Score from Cross-Validation: 0.9007477824665224
Best n_components value: 203
```

[ ]:
```
print('pca n_components', x_1se)
index_x_1se = np.where(n_components == x_1se)[0][0]

# Get the corresponding R2 score
R2_score_x_1se = R2_scores[index_x_1se]
print("R2 Score corresponding to x_1se:", R2_score_x_1se)
```

```
pca n_components 88
R2 Score corresponding to x_1se: 0.8954299553618595
```

[ ]:
```
print_metrics(pcr_cv, X_train_pcr, y_train, X_test_pcr, y_test,␣
 ↪flag_log_sale_price = True)
print_metrics(pcr_cv, X_train_pcr, y_train, X_test_pcr, y_test,␣
 ↪flag_log_sale_price = False)
```

```
Metrics for Log(Sale Price):

Training R2 0.9470509309926725
Training MAE 0.06613921430190121
Training RMSE 0.08906848150530083
Out-of-sample R2 0.8909564729172067
Out-of-sample MAE 0.08577621657861301
Out-of-sample RMSE 0.13078751186261808

Metrics for Sale Price:

Training R2 0.9498759628436522
Training MAE 12080.226374617378
Training RMSE 17490.035484096694
Out-of-sample R2 0.9208510195464497
Out-of-sample MAE 14744.142116330762
Out-of-sample RMSE 22050.215262039193
```

**Refit the model with the selected parameter**

[ ]:
```
pipe.set_params(pca__n_components=x_1se)
pipe.fit(X_train_lasso, y_train)
pipe.get_params()
```

[ ]:
```
{'memory': None,
 'steps': [('scaler', StandardScaler()),
  ('pca', PCA(n_components=88, random_state=88)),
```

```
  ('lr', LinearRegression())],
 'verbose': False,
 'scaler': StandardScaler(),
 'pca': PCA(n_components=88, random_state=88),
 'lr': LinearRegression(),
 'scaler__copy': True,
 'scaler__with_mean': True,
 'scaler__with_std': True,
 'pca__copy': True,
 'pca__iterated_power': 'auto',
 'pca__n_components': 88,
 'pca__n_oversamples': 10,
 'pca__power_iteration_normalizer': 'auto',
 'pca__random_state': 88,
 'pca__svd_solver': 'auto',
 'pca__tol': 0.0,
 'pca__whiten': False,
 'lr__copy_X': True,
 'lr__fit_intercept': True,
 'lr__n_jobs': None,
 'lr__positive': False}
```

[ ]:
```
print_metrics(pipe, X_train_pcr, y_train, X_test_pcr, y_test,␣
 ↪flag_log_sale_price = True)
print_metrics(pipe, X_train_pcr, y_train, X_test_pcr, y_test,␣
 ↪flag_log_sale_price = False)
```

```
Metrics for Log(Sale Price):

Training R2 0.9307885562259678
Training MAE 0.07523869573417481
Training RMSE 0.10183189800532155
Out-of-sample R2 0.8867156269855305
Out-of-sample MAE 0.09027393060383741
Out-of-sample RMSE 0.13330650277813924

Metrics for Sale Price:

Training R2 0.93768977143283
Training MAE 13632.105577276641
Training RMSE 19500.5715801474
Out-of-sample R2 0.9217227150903017
Out-of-sample MAE 15441.372024002116
Out-of-sample RMSE 21928.455702831296
```

## 2.1 Cross validation for Ridge Regression

We can choose `alpha_max` so as the value that makes all coefficientes zero, and then construct a log sequence of `alpha` values trending smaller, decreasing the degree of regularization.

For the case of `Ridge` Regression, alpha value that would make all coefficients zero would be `Inf`, however we can be satisfied with sufficiently small numbers, and work from there.

[ ]:
```
X_train_rr = X_train_poly_wide
X_test_rr = X_test_poly_wide

print(X_train_rr.shape, X_test_rr.shape)
```

```
(1828, 397) (937, 397)
```

### 2.1.1 Determine 'alpha_max'

[ ]:
```
from sklearn.linear_model import Ridge

alpha_max = 10**5
rr = Ridge(alpha=alpha_max, random_state=88)
rr.fit(X_train_rr, y_train)
```

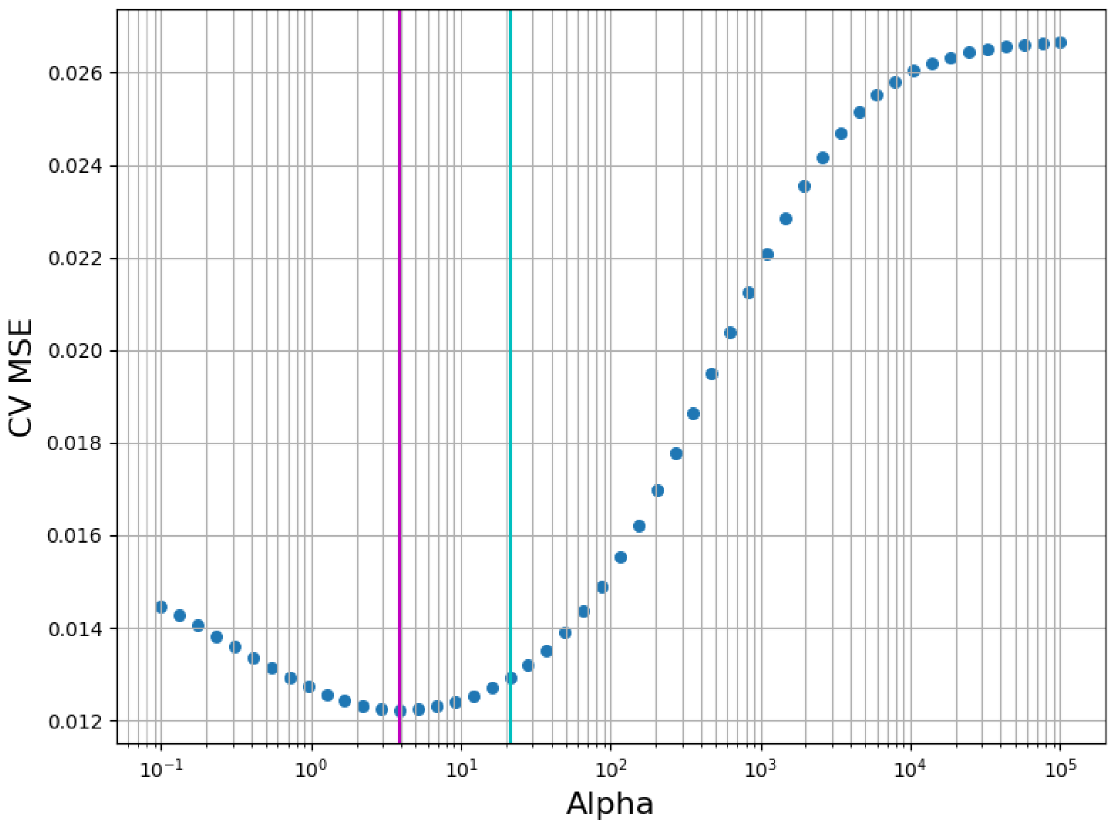
[ ]:
```
Ridge(alpha=100000, random_state=88)
```

### 2.1.2 Ridge Hyper-parameter Tuning

[ ]:
```
alpha_grid = {'alpha': np.logspace(-1, 5, num=50, base=10)}

rr = Ridge(random_state=88)
rr_cv = GridSearchCV(rr, alpha_grid, scoring='neg_mean_squared_error', cv=5)
rr_cv.fit(X_train_rr, y_train)
```

[ ]:
```
GridSearchCV(cv=5, estimator=Ridge(random_state=88),
             param_grid={'alpha': array([1.00000000e-01, 1.32571137e-01,
1.75751062e-01, 2.32995181e-01,
       3.08884360e-01, 4.09491506e-01, 5.42867544e-01, 7.19685673e-01,
       9.54095476e-01, 1.26485522e+00, 1.67683294e+00, 2.22299648e+00,
       2.94705170e+00, 3.90693994e+00, 5.17947468e+00, 6.86648845e+00,
       9.10298178e+00, 1.20679264e+01, 1…
       2.68269580e+02, 3.55648031e+02, 4.71486636e+02, 6.25055193e+02,
       8.28642773e+02, 1.09854114e+03, 1.45634848e+03, 1.93069773e+03,
       2.55954792e+03, 3.39322177e+03, 4.49843267e+03, 5.96362332e+03,
       7.90604321e+03, 1.04811313e+04, 1.38949549e+04, 1.84206997e+04,
       2.44205309e+04, 3.23745754e+04, 4.29193426e+04, 5.68986603e+04,
       7.54312006e+04, 1.00000000e+05])},
             scoring='neg_mean_squared_error')
```

[ ]:
```
range_alpha = rr_cv.cv_results_['param_alpha'].data
MSE_scores = rr_cv.cv_results_['mean_test_score']*(-1)
x_min, x_1se = one_standard_error_rule(model='ridge',
                                       results=rr_cv.cv_results_,
                                       param_grid=range_alpha,
                                       n_splits=10,
                                       neg_mean_squared_error=True)
plt.figure(figsize=(8, 6))
ax = plt.gca()
ax.set_xscale('log')
plt.xlabel('Alpha', fontsize=16)
plt.ylabel('CV MSE', fontsize=16)
plt.scatter(range_alpha, MSE_scores, s=30)
plt.axvline(x=x_min, color='m')
plt.axvline(x=x_1se, color='c')
plt.grid(True, which='both')

plt.tight_layout()
plt.show()
```



[ ]:
```
print('Alpha one standard error rule:', x_1se)
```

```
Alpha one standard error rule: 21.209508879201906
```

### 2.1.3 Ridge Refit with One Standard Error Rule

[ ]:
```
rr.set_params(alpha=x_1se)
rr.fit(X_train_lasso, y_train)
rr.get_params()
```

[ ]:
```
{'alpha': 21.209508879201906,
 'copy_X': True,
 'fit_intercept': True,
 'max_iter': None,
 'positive': False,
 'random_state': 88,
 'solver': 'auto',
 'tol': 0.0001}
```

[ ]:
```
print_metrics(rr, X_train_lasso, y_train, X_test_lasso, y_test,␣
 ↪flag_log_sale_price = True)
print_metrics(rr, X_train_lasso, y_train, X_test_lasso, y_test,␣
 ↪flag_log_sale_price = False)
```

```
Metrics for Log(Sale Price):

Training R2 0.9459931437852581
Training MAE 0.06235134218698391
Training RMSE 0.0899537623805996
Out-of-sample R2 0.893481603642305
Out-of-sample MAE 0.08065342198667663
Out-of-sample RMSE 0.1292643129340068

Metrics for Sale Price:

Training R2 0.9537554772032324
Training MAE 11195.688008014222
Training RMSE 16799.55663050555
Out-of-sample R2 0.9209438962794584
Out-of-sample MAE 13888.3660977596
Out-of-sample RMSE 22037.2741408468
```

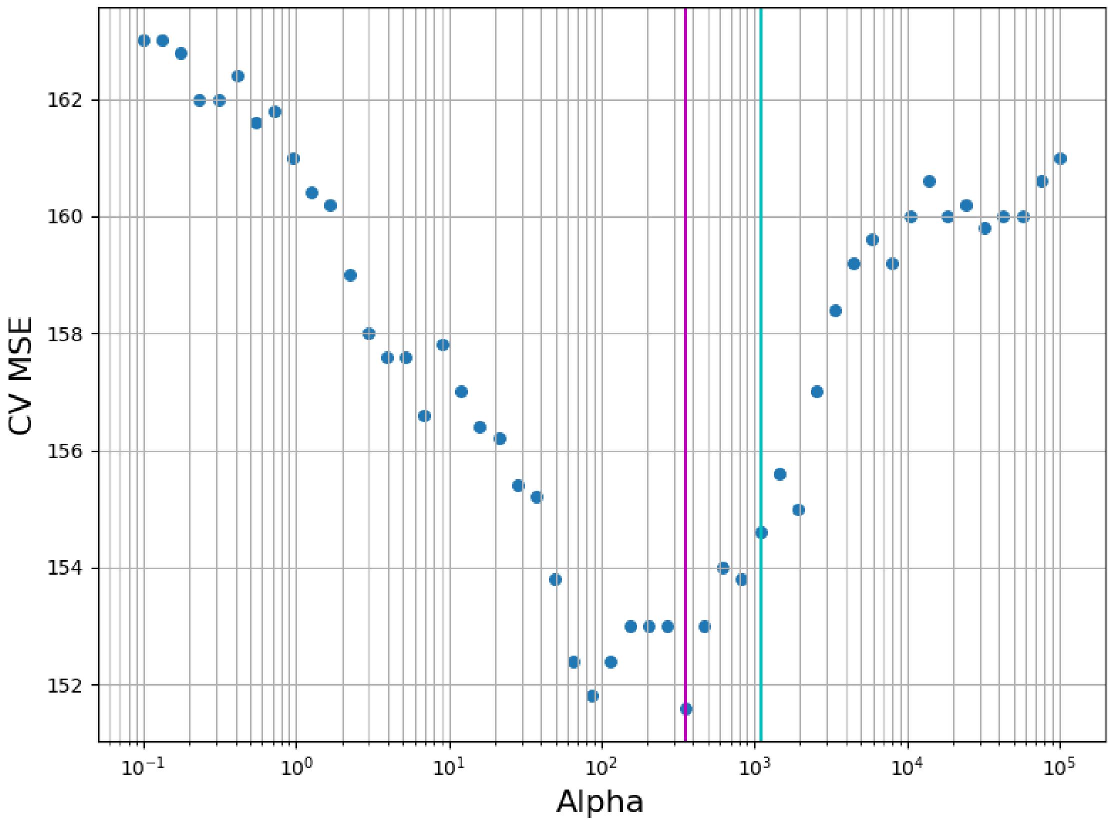
[ ]:
```
print_metrics(rr_cv, X_train_lasso, y_train, X_test_lasso, y_test,␣
 ↪flag_log_sale_price = True)
print_metrics(rr_cv, X_train_lasso, y_train, X_test_lasso, y_test,␣
 ↪flag_log_sale_price = False)
```

```
Metrics for Log(Sale Price):

Training R2 0.9566658840031975
Training MAE 0.0576670905483711
Training RMSE 0.08057677009982724
Out-of-sample R2 0.9003061738445208
Out-of-sample MAE 0.07877099524934432
Out-of-sample RMSE 0.12505482836437523

Metrics for Sale Price:

Training R2 0.9605502868413793
Training MAE 10432.00812852822
Training RMSE 15516.350946475923
Out-of-sample R2 0.9277610608337501
Out-of-sample MAE 13581.237490221118
Out-of-sample RMSE 21065.697861595443
```

# 3 In-class activity 3: Do a cross validation for Ridge regression using custom loss function `large_prediction_error_count`. What do you observe from the result?

[ ]:
```
from sklearn.linear_model import Ridge

alpha_max = 10**5
rr = Ridge(alpha=alpha_max, random_state=88)
rr.fit(X_train_rr, y_train)

alpha_grid = {'alpha': np.logspace(-1, 5, num=50, base=10)}

rr = Ridge(random_state=88)
rr_cv = GridSearchCV(rr, alpha_grid,
 ↪scoring=make_scorer(large_prediction_error_count, greater_is_better=False),
 ↪cv=5)
rr_cv.fit(X_train_rr, y_train)

range_alpha = rr_cv.cv_results_['param_alpha'].data
MSE_scores = rr_cv.cv_results_['mean_test_score']*(-1)
x_min, x_1se = one_standard_error_rule(model='ridge',
                                       results=rr_cv.cv_results_,
                                       param_grid=range_alpha,
                                       n_splits=10,
                                       neg_mean_squared_error=True)
plt.figure(figsize=(8, 6))
ax = plt.gca()
```

```
ax.set_xscale('log')
plt.xlabel('Alpha', fontsize=16)
plt.ylabel('CV MSE', fontsize=16)
plt.scatter(range_alpha, MSE_scores, s=30)
plt.axvline(x=x_min, color='m')
plt.axvline(x=x_1se, color='c')
plt.grid(True, which='both')

plt.tight_layout()
plt.show()
```




[ ]:
```
print('Alpha one standard error rule:', x_1se)
```

```
Alpha one standard error rule: 1098.5411419875572
```