index

January 25, 2022

1 Model Fit in Linear Regression - Lab

1.1 Introduction

In this lab, you'll learn how to evaluate your model results and you'll learn how to select the appropriate features using stepwise selection.

1.2 Objectives

You will be able to: * Use stepwise selection methods to determine the most important features for a model * Use recursive feature elimination to determine the most important features for a model

1.3 The Ames Housing Data once more

[1]:
```
import pandas as pd
import numpy as np

ames = pd.read_csv('ames.csv')

continuous = ['LotArea', '1stFlrSF', 'GrLivArea', 'SalePrice']
categoricals = ['BldgType', 'KitchenQual', 'SaleType', 'MSZoning', 'Street',
                'Neighborhood']

ames_cont = ames[continuous]

# log features
log_names = [f'{column}_log' for column in ames_cont.columns]

ames_log = np.log(ames_cont)
ames_log.columns = log_names

# normalize (subract mean and divide by std)

def normalize(feature):
    return (feature - feature.mean()) / feature.std()

ames_log_norm = ames_log.apply(normalize)

# one hot encode categoricals
```

```python
ames_ohe = pd.get_dummies(ames[categoricals], prefix=categoricals,␣
↪drop_first=True)

preprocessed = pd.concat([ames_log_norm, ames_ohe], axis=1)
```

1.4 Perform stepwise selection

The function for stepwise selection is copied below. Use this provided function on your preprocessed Ames Housing data.

[2]:
```python
import statsmodels.api as sm

def stepwise_selection(X, y,
                       initial_list=[],
                       threshold_in=0.01,
                       threshold_out = 0.05,
                       verbose=True):
    """
    Perform a forward-backward feature selection
    based on p-value from statsmodels.api.OLS
    Arguments:
        X - pandas.DataFrame with candidate features
        y - list-like with the target
        initial_list - list of features to start with (column names of X)
        threshold_in - include a feature if its p-value < threshold_in
        threshold_out - exclude a feature if its p-value > threshold_out
        verbose - whether to print the sequence of inclusions and exclusions
    Returns: list of selected features
    Always set threshold_in < threshold_out to avoid infinite looping.
    See https://en.wikipedia.org/wiki/Stepwise_regression for the details
    """
    included = list(initial_list)
    while True:
        changed=False
        # forward step
        excluded = list(set(X.columns)-set(included))
        new_pval = pd.Series(index=excluded, dtype='float64')
        for new_column in excluded:
            model = sm.OLS(y, sm.add_constant(pd.
↪DataFrame(X[included+[new_column]]))).fit()
            new_pval[new_column] = model.pvalues[new_column]
        best_pval = new_pval.min()
        if best_pval < threshold_in:
            best_feature = new_pval.idxmin()
            included.append(best_feature)
            changed=True
            if verbose:
```

```
                print('Add  {:30} with p-value {:.6}'.format(best_feature,␣
↪best_pval))

        # backward step
        model = sm.OLS(y, sm.add_constant(pd.DataFrame(X[included]))).fit()
        # use all coefs except intercept
        pvalues = model.pvalues.iloc[1:]
        worst_pval = pvalues.max() # null if pvalues is empty
        if worst_pval > threshold_out:
            changed=True
            worst_feature = pvalues.idxmax()
            included.remove(worst_feature)
            if verbose:
                print('Drop {:30} with p-value {:.6}'.format(worst_feature,␣
↪worst_pval))
        if not changed:
            break
    return included
```

[6]:

```
# Your code here
X = preprocessed.drop("SalePrice_log", axis = 1)
y = preprocessed["SalePrice_log"]
columns_to_pick = stepwise_selection(X,
                   y,
                   initial_list=[],
                   threshold_in=0.01,
                   threshold_out = 0.05,
                   verbose=True)
columns_to_pick
```

```
Add  GrLivArea_log                  with p-value 1.59847e-243
Add  KitchenQual_TA                 with p-value 1.56401e-67
Add  1stFlrSF_log                   with p-value 7.00069e-48
Add  KitchenQual_Fa                 with p-value 1.70471e-37
Add  Neighborhood_OldTown           with p-value 3.20105e-23
Add  KitchenQual_Gd                 with p-value 4.12635e-21
Add  Neighborhood_Edwards           with p-value 9.05184e-17
Add  Neighborhood_IDOTRR            with p-value 1.10068e-18
Add  LotArea_log                    with p-value 1.71728e-13
Add  Neighborhood_NridgHt           with p-value 7.05633e-12
Add  BldgType_Duplex                with p-value 4.30647e-11
Add  Neighborhood_NAmes             with p-value 2.25803e-09
Add  Neighborhood_SWISU             with p-value 5.40743e-09
Add  Neighborhood_BrkSide           with p-value 8.79638e-10
Add  Neighborhood_Sawyer            with p-value 6.92011e-09
Add  Neighborhood_NoRidge           with p-value 5.87105e-08
Add  Neighborhood_Somerst           with p-value 3.00722e-08
```

```
Add  Neighborhood_StoneBr           with p-value 6.58621e-10
Add  Neighborhood_MeadowV           with p-value 2.26069e-05
Add  SaleType_New                   with p-value 0.000485363
Add  SaleType_WD                    with p-value 0.00253157
Add  Neighborhood_BrDale            with p-value 0.00374541
Add  MSZoning_RM                    with p-value 8.29694e-05
Add  MSZoning_RL                    with p-value 0.00170469
Add  MSZoning_FV                    with p-value 0.00114668
Add  MSZoning_RH                    with p-value 3.95797e-05
Add  Neighborhood_NWAmes            with p-value 0.00346099
Drop SaleType_WD                    with p-value 0.0554448
Add  Neighborhood_Mitchel           with p-value 0.00994666
Drop Neighborhood_Somerst           with p-value 0.0500753
Add  Neighborhood_SawyerW           with p-value 0.00427685
```

[6]:
```
['GrLivArea_log',
 'KitchenQual_TA',
 '1stFlrSF_log',
 'KitchenQual_Fa',
 'Neighborhood_OldTown',
 'KitchenQual_Gd',
 'Neighborhood_Edwards',
 'Neighborhood_IDOTRR',
 'LotArea_log',
 'Neighborhood_NridgHt',
 'BldgType_Duplex',
 'Neighborhood_NAmes',
 'Neighborhood_SWISU',
 'Neighborhood_BrkSide',
 'Neighborhood_Sawyer',
 'Neighborhood_NoRidge',
 'Neighborhood_StoneBr',
 'Neighborhood_MeadowV',
 'SaleType_New',
 'Neighborhood_BrDale',
 'MSZoning_RM',
 'MSZoning_RL',
 'MSZoning_FV',
 'MSZoning_RH',
 'Neighborhood_NWAmes',
 'Neighborhood_Mitchel',
 'Neighborhood_SawyerW']
```

1.4.1 Build the final model again in Statsmodels

[8]:
```python
# Your code here
# Import statsmodels.api as sm
import statsmodels.api as sm

outcome = preprocessed['SalePrice_log']
predictors = preprocessed[columns_to_pick]
predictors_with_intercept = sm.add_constant(predictors)

model = sm.OLS(outcome,predictors_with_intercept).fit()
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           SalePrice_log   R-squared:                       0.835
Model:                             OLS   Adj. R-squared:                  0.832
Method:                  Least Squares   F-statistic:                     269.0
Date:                 Tue, 25 Jan 2022   Prob (F-statistic):               0.00
Time:                         23:19:31   Log-Likelihood:                -754.40
No. Observations:                 1460   AIC:                             1565.
Df Residuals:                     1432   BIC:                             1713.
Df Model:                           27
Covariance Type:             nonrobust
================================================================================
========
                           coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
--------
const                   -0.2174      0.164     -1.323      0.186      -0.540
0.105
GrLivArea_log            0.3694      0.015     24.477      0.000       0.340
0.399
KitchenQual_TA          -0.7020      0.055    -12.859      0.000      -0.809
-0.595
1stFlrSF_log             0.1445      0.015      9.645      0.000       0.115
0.174
KitchenQual_Fa          -1.0372      0.087    -11.864      0.000      -1.209
-0.866
Neighborhood_OldTown    -0.8625      0.063    -13.615      0.000      -0.987
-0.738
KitchenQual_Gd          -0.4021      0.050     -8.046      0.000      -0.500
-0.304
Neighborhood_Edwards    -0.7019      0.048    -14.530      0.000      -0.797
-0.607
Neighborhood_IDOTRR     -0.8583      0.097     -8.855      0.000      -1.048
-0.668
```

```
LotArea_log              0.1096      0.015      7.387      0.000       0.081
0.139
Neighborhood_NridgHt     0.3854      0.057      6.809      0.000       0.274
0.496
BldgType_Duplex         -0.4073      0.061     -6.678      0.000      -0.527
-0.288
Neighborhood_NAmes      -0.3763      0.038     -9.981      0.000      -0.450
-0.302
Neighborhood_SWISU      -0.6263      0.089     -7.020      0.000      -0.801
-0.451
Neighborhood_BrkSide    -0.5641      0.066     -8.493      0.000      -0.694
-0.434
Neighborhood_Sawyer     -0.4026      0.055     -7.342      0.000      -0.510
-0.295
Neighborhood_NoRidge     0.4347      0.070      6.221      0.000       0.298
0.572
Neighborhood_StoneBr     0.4538      0.087      5.226      0.000       0.283
0.624
Neighborhood_MeadowV    -0.6622      0.118     -5.592      0.000      -0.895
-0.430
SaleType_New             0.1483      0.044      3.388      0.001       0.062
0.234
Neighborhood_BrDale     -0.4733      0.123     -3.839      0.000      -0.715
-0.231
MSZoning_RM              1.0820      0.147      7.363      0.000       0.794
1.370
MSZoning_RL              0.9916      0.156      6.356      0.000       0.686
1.298
MSZoning_FV              1.2052      0.165      7.284      0.000       0.881
1.530
MSZoning_RH              0.8503      0.189      4.490      0.000       0.479
1.222
Neighborhood_NWAmes     -0.2055      0.054     -3.837      0.000      -0.311
-0.100
Neighborhood_Mitchel    -0.1943      0.065     -3.004      0.003      -0.321
-0.067
Neighborhood_SawyerW    -0.1666      0.058     -2.862      0.004      -0.281
-0.052
==============================================================================
Omnibus:                      295.535   Durbin-Watson:                   1.965
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1270.571
Skew:                          -0.903   Prob(JB):                    1.26e-276
Kurtosis:                       7.198   Cond. No.                         48.7
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

[9]:
```
### From internent
results_as_html1 = model.summary().tables[0].as_html()
pd.read_html(results_as_html1, header=0, index_col=0)[0]
```

[9]:

	SalePrice_log	R-squared:	0.835
Dep. Variable:			
Model:	OLS	Adj. R-squared:	0.832
Method:	Least Squares	F-statistic:	269.000
Date:	Tue, 25 Jan 2022	Prob (F-statistic):	0.000
Time:	23:20:18	Log-Likelihood:	-754.400
No. Observations:	1460	AIC:	1565.000
Df Residuals:	1432	BIC:	1713.000
Df Model:	27	NaN	NaN
Covariance Type:	nonrobust	NaN	NaN

[10]:
```
### From internent
results_as_html1 = model.summary().tables[1].as_html()
pd.read_html(results_as_html1, header=0, index_col=0)[0]
```

[10]:

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.2174	0.164	-1.323	0.186	-0.540	0.105
GrLivArea_log	0.3694	0.015	24.477	0.000	0.340	0.399
KitchenQual_TA	-0.7020	0.055	-12.859	0.000	-0.809	-0.595
1stFlrSF_log	0.1445	0.015	9.645	0.000	0.115	0.174
KitchenQual_Fa	-1.0372	0.087	-11.864	0.000	-1.209	-0.866
Neighborhood_OldTown	-0.8625	0.063	-13.615	0.000	-0.987	-0.738
KitchenQual_Gd	-0.4021	0.050	-8.046	0.000	-0.500	-0.304
Neighborhood_Edwards	-0.7019	0.048	-14.530	0.000	-0.797	-0.607
Neighborhood_IDOTRR	-0.8583	0.097	-8.855	0.000	-1.048	-0.668
LotArea_log	0.1096	0.015	7.387	0.000	0.081	0.139
Neighborhood_NridgHt	0.3854	0.057	6.809	0.000	0.274	0.496
BldgType_Duplex	-0.4073	0.061	-6.678	0.000	-0.527	-0.288
Neighborhood_NAmes	-0.3763	0.038	-9.981	0.000	-0.450	-0.302
Neighborhood_SWISU	-0.6263	0.089	-7.020	0.000	-0.801	-0.451
Neighborhood_BrkSide	-0.5641	0.066	-8.493	0.000	-0.694	-0.434
Neighborhood_Sawyer	-0.4026	0.055	-7.342	0.000	-0.510	-0.295
Neighborhood_NoRidge	0.4347	0.070	6.221	0.000	0.298	0.572
Neighborhood_StoneBr	0.4538	0.087	5.226	0.000	0.283	0.624
Neighborhood_MeadowV	-0.6622	0.118	-5.592	0.000	-0.895	-0.430
SaleType_New	0.1483	0.044	3.388	0.001	0.062	0.234
Neighborhood_BrDale	-0.4733	0.123	-3.839	0.000	-0.715	-0.231
MSZoning_RM	1.0820	0.147	7.363	0.000	0.794	1.370
MSZoning_RL	0.9916	0.156	6.356	0.000	0.686	1.298
MSZoning_FV	1.2052	0.165	7.284	0.000	0.881	1.530
MSZoning_RH	0.8503	0.189	4.490	0.000	0.479	1.222
Neighborhood_NWAmes	-0.2055	0.054	-3.837	0.000	-0.311	-0.100
Neighborhood_Mitchel	-0.1943	0.065	-3.004	0.003	-0.321	-0.067

```
Neighborhood_SawyerW -0.1666    0.058  -2.862  0.004  -0.281  -0.052
```

[11]:
```
### From internent
results_as_html1 = model.summary().tables[2].as_html()
pd.read_html(results_as_html1, header=0, index_col=0)[0]
```

[11]:
```
                  295.535     Durbin-Watson:          1.965
Omnibus:
Prob(Omnibus):      0.000  Jarque-Bera (JB):    1.270571e+03
Skew:              -0.903          Prob(JB):   1.260000e-276
Kurtosis:           7.198          Cond. No.    4.870000e+01
```

1.5 Use Feature ranking with recursive feature elimination

Use feature ranking to select the 5 most important features

[12]:
```
# Your code here

from sklearn.feature_selection import RFE
from sklearn.linear_model import LinearRegression

linreg = LinearRegression()
selector = RFE(linreg, n_features_to_select=5)
selector = selector.fit(predictors, preprocessed['SalePrice_log'])
```

Fit the linear regression model again using the 5 selected columns

[14]:
```
# Your code here
columns_to_pick_2 = []
for i,item in enumerate(columns_to_pick):
    if selector.support_[i] == True:
        columns_to_pick_2.append(item)

columns_to_pick_2
```

[14]:
```
['Neighborhood_NoRidge',
 'MSZoning_RM',
 'MSZoning_RL',
 'MSZoning_FV',
 'MSZoning_RH']
```

[15]:
```
import statsmodels.api as sm

outcome = preprocessed['SalePrice_log']
predictors = preprocessed[columns_to_pick_2]
predictors_with_intercept = sm.add_constant(predictors)

model = sm.OLS(outcome,predictors_with_intercept).fit()
```

```python
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:          SalePrice_log   R-squared:                       0.239
Model:                            OLS   Adj. R-squared:                  0.237
Method:                 Least Squares   F-statistic:                     91.55
Date:                Tue, 25 Jan 2022   Prob (F-statistic):           6.73e-84
Time:                        23:26:49   Log-Likelihood:                -1871.4
No. Observations:                1460   AIC:                             3755.
Df Residuals:                    1454   BIC:                             3786.
Df Model:                           5
Covariance Type:            nonrobust
================================================================================
========
                           coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
--------
const                   -2.2676      0.276     -8.208      0.000      -2.809
-1.726
Neighborhood_NoRidge     1.5319      0.139     11.026      0.000       1.259
1.804
MSZoning_RM              1.4386      0.283      5.092      0.000       0.884
1.993
MSZoning_RL              2.3678      0.277      8.533      0.000       1.823
2.912
MSZoning_FV              2.8248      0.297      9.519      0.000       2.243
3.407
MSZoning_RH              1.5811      0.352      4.490      0.000       0.890
2.272
==============================================================================
Omnibus:                       47.664   Durbin-Watson:                   1.976
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               79.387
Skew:                           0.272   Prob(JB):                     5.77e-18
Kurtosis:                       4.004   Cond. No.                         35.8
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

[24]:
```python
# Your code here
model.predict(predictors_with_intercept)
```

[24]:
```
0       0.10023
1       0.10023
```

```
2        0.10023
3        0.10023
4        1.63211
           …
1455     0.10023
1456     0.10023
1457     0.10023
1458     0.10023
1459     0.10023
Length: 1460, dtype: float64
```

Now, predict $\hat{y}$ using your model. You can use `.predict()` in scikit-learn.

[26]:
```
### From GitHub
y = preprocessed['SalePrice_log']
x = preprocessed[columns_to_pick_2]
linreg.fit(x,y)
y_pred = linreg.predict(variables)
y_pred
```

[26]:
```
array([0.10023007, 0.10023007, 0.10023007, …, 0.10023007, 0.10023007,
       0.10023007])
```

Now, using the formulas of R-squared and adjusted R-squared below, and your Python/numpy knowledge, compute them and contrast them with the R-squared and adjusted R-squared in your statsmodels output using stepwise selection. Which of the two models would you prefer?

$$SS_{residual} = \sum (y - \hat{y})^2 \tag{1}$$

$$SS_{total} = \sum (y - \bar{y})^2 \tag{2}$$

$$R^2 = 1 - \frac{SS_{residual}}{SS_{total}} \tag{3}$$

$$R^2_{adj} = 1 - (1 - R^2)\frac{n-1}{n-p-1} \tag{4}$$

[41]:
```
# Your code here
y_real = np.array(preprocessed['SalePrice_log'])
y_mean = np.mean(y_real)

n = len(y_real)
p = len(columns_to_pick_2)
## From github p is preprocessed[selected_columns].shape[1] which
# gives the length of the list columns_to_pick_2
```

```
res = y_real - y_pred
res_mean = y_real - y_mean
SS_res = np.inner(res,res)
SS_tot = np.inner(res_mean,res_mean)

R2 = 1 - SS_res / SS_tot
R2_adj = 1 - (1 - R2)*(n-1) / (n-p-1)
R2_adj
# r_squared is 0.239434
# adjusted_r_squared is 0.236818
```

[41]: 0.2368187559863113

1.6 Level up (Optional)

- Perform variable selection using forward selection, using this resource: https://planspace.org/20150423-forward_selection_with_statsmodels/. Note that this time features are added based on the adjusted R-squared!
- Tweak the code in the `stepwise_selection()` function written above to just perform forward selection based on the p-value

1.7 Summary

Great! You practiced your feature selection skills by applying stepwise selection and recursive feature elimination to the Ames Housing dataset!