Data Analysis and Machine-Learning

Chapter 5.2:

Feature Selection (2)

Forward Selection, Backward Elimination, and Stepwise Selection



1. Selection of Explanatory Features

I have illustrated the concepts of feature selection and resolving multicollinearity problem in chapter 4.1. Continuously to the previous chapter, this chapter will introduce methodologies to look for the most explanatory variables in terms of creating the best model with less periphery variables and high explanatory accuracy.

2. Forward Selection Method

Forward selection method refers to the feature selection methodology of selecting most accurate variables by adding variables, one at a time, and calculating the accuracy score for each, eventually to get the best explanatory set of variables. Since Python does not provide packages for forward selection method, it is necessary to make algorithms manually.

def rSquare( x, y, yhat ):

    if x.ndim == 1: p, n = 1, x.shape[0]

    else: p, n = x.shape[1], x.shape[0]

    r2 = 1 - np.sum( (y - yhat) \*\* 2) / np.sum( (y - np.mean(y)) \*\* 2 )

    adj\_r2 = 1 - (1 - r2) \* ( n - 1) / ( n - p - 1 )

    return {'r2': r2, 'adjr2': adj\_r2}

We will use the adj.R2 score as the criterion of scoring, as R2 score is vulnerable to score inflation when there are multiple set of independent variables.

def forward(model, x, y, col\_selected):

    col\_forw = [ cl for cl in features.columns if cl not in col\_selected ]

    rst = []

    for col in col\_forw:

        columns = col\_selected + [col]

        model = model.fit(x[columns], y)

        yhat = model.predict(x[columns])

        score = rSquare(x[columns], y, yhat)

        rst.append( {'model': model, 'score': score['adjr2'], 'columns': columns} )

    models = pd.DataFrame(rst)

    model\_best = models.loc[ models.score.argmax() ]

    return model\_best

def forward\_selection(x, y):

    col\_selected = []

    for i in range(0, x.shape[1]):

        model = LinearRegression()

        model\_best = forward(model, features, target, col\_selected)

        if not i:

            model\_prior = model\_best

        else:

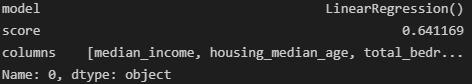
            if model\_best.score > model\_prior.score: model\_prior = model\_best

            else: break

        col\_selected = model\_best.columns

    return model\_prior

forward\_selection(features, target)



Let us check if the score above matches with the score calculated manually:

features\_forward = features[forward\_selection(features, target).columns]

model\_forward = LinearRegression().fit(features\_forward, target)

yhat = model\_forward.predict(features\_forward)

rSquare(features\_forward, target, yhat)

output:

{'r2': 0.6413378529502689, 'adjr2': 0.6411689813608158}

3. Backward Elimination Method

Contrary to the forward selection method, backward elimination refers to the methodology of starting from the base model with all variables, and eliminating less explanatory variables, one at a time. In python, the algorithm for backward elimination can be created via using combinations in itertools, as follows.

from itertools import combinations

def backward(model, x, y, col\_selected):

    rst = []

    for com in combinations(col\_selected, len(col\_selected)-1):

        columns = list(com)

        model = model.fit(x[columns], y)

        yhat = model.predict(x[columns])

        score = rSquare(x[columns], y, yhat)

        rst.append( {'model': model, 'score': score['adjr2'], 'columns': columns} )

    models = pd.DataFrame(rst)

    model\_best = models.loc[ models.score.argmax() ]

    return model\_best

def backward\_elimination(x, y):

    col\_selected = x.columns

    for i in range(0, x.shape[1]):

        model = LinearRegression()

        model\_best = backward(model, features, target, col\_selected)

        if not i:

            model\_prior = model\_best

        else:

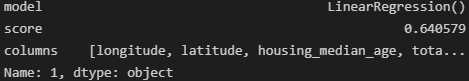
            if model\_best.score > model\_prior.score: model\_prior = model\_best

            else: break

        col\_selected = model\_best.columns

    return model\_prior

backward\_elimination(features, target)



Let us check if the score above matches with the score calculated manually:

features\_backward = features[backward\_elimination(features, target).columns]

model\_backward = LinearRegression().fit(features\_backward, target)

yhat = model\_backward.predict(features\_backward)

rSquare(features\_backward, target, yhat)

output:

{'r2': 0.6407268394704915, 'adjr2': 0.6405788338134937}

4. Stepwise Selection Method

Stepwise method performs selection and elimination for each step, and evaluates accordingly to select the best scoring model; thus the algorithm requires both forward() and backward() algorithms.

#Define forward and backward, as previously:

def forward(model, x, y, col\_selected):

    col\_forw = [ cl for cl in features.columns if cl not in col\_selected ]

    rst = []

    for col in col\_forw:

        columns = col\_selected + [col]

        model = model.fit(x[columns], y)

        yhat = model.predict(x[columns])

        score = rSquare(x[columns], y, yhat)

        rst.append( {'model': model, 'score': score['adjr2'], 'columns': columns} )

    models = pd.DataFrame(rst)

    model\_best = models.loc[ models.score.argmax() ]

    return model\_best

def backward(model, x, y, col\_selected):

    rst = []

    for com in combinations(col\_selected, len(col\_selected)-1):

        columns = list(com)

        model = model.fit(x[columns], y)

        yhat = model.predict(x[columns])

        score = rSquare(x[columns], y, yhat)

        rst.append( {'model': model, 'score': score['adjr2'], 'columns': columns} )

    models = pd.DataFrame(rst)

    model\_best = models.loc[ models.score.argmax() ]

    return model\_best

#Define stepwise algorithm

def stepwise\_selection(x, y):

    col\_selected = []

    for i in range(x.shape[1]):

        model = LinearRegression()

        model\_forw = forward(model, x, y, col\_selected)

        col\_selected = model\_forw.columns

        if i < 1: model\_prior = model\_forw; continue

        model\_back = backward(model, x, y, col\_selected)

        model\_high = model\_forw

        if model\_forw.score < model\_back.score:

            col\_selected = model\_back.columns

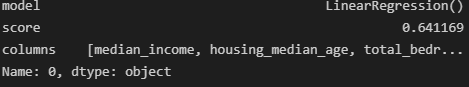
            model\_high = model\_back

        if model\_high.score > model\_prior.score: model\_prior = model\_high

        else: break

    return model\_prior

stepwise\_selection(features, target)



Let us check if the score above matches with the score calculated manually:

features\_stepwise = pd.DataFrame(features[stepwise\_selection(features, target).columns])

model\_stepwise = LinearRegression().fit(features\_stepwise, target)

yhat = model\_stepwise.predict(features\_stepwise)

rSquare(features\_stepwise, target, yhat)

output:

{'r2': 0.6413378529502689, 'adjr2': 0.6411689813608158}