Artificial Intelligence

AI-04: Explanation of linear_reg_adv.py

brute_force_aic()

model: Regression model. Currently, linear regression, so smf.ols is assigned.
exog: List of explanatory variables.
endog: List of objective variable(s) (only one in this case).
**kwards: Arguments for model.fit().

```
def brute_force_aic(model, exog, endog, **kwargs):
    min_score = None
    min_selected = None
    formula_head = ' + '.join(endog) + ' ~ '
    exog_ar = np.array(exog)
    n = len(exog_ar)
    …
```

min_score: AIC value of the best formula (formula with the smallest AIC) so far.

min_selected: List of explanatory variables for the best formula (formula with the smallest AIC) so far.

The formula is given by the string "objective_variable ~ explanatory_variable1 + explanatory_variable2 + ..." where "formula_head" is "objective_variable ~" part. If there is more than one objective variable, "formula_head" becomes "objective variable1 + objective_variable2 + ... ~".

exog_ar: convert list of explanatory variables to ndarray.
n: Number of explanatory variables

Examine every combination of the n explanatory variables. To do this, move i from 1 to 2**n -1, and let nb be the binary representation of i.

nb:
0 0 … 0 0 1 ← Use only No.0 explanatory variable.
0 0 … 0 1 0 ← Use only No.1 explanatory variable.
0 0 … 0 1 1 ← Use No.0 and No.1 explanatory variables.
…
1 1 … 1 1 1 ← Use all explanatory variables.

```
…
for i in range(1, 2**n):
    ib = str(bin(i))[2:]; m = len(ib)
    nb = np.array(list('0'*(n-m)+ib))
    selected = exog_ar[ nb=='1' ]
    formula_tail = ' + '.join(selected)
    formula = formula_head + formula_tail
    aic = model(formula=formula, **kwargs).fit().aic
    print('AIC: {:.3f}, formula: {}'.format(aic, formula))
    …
```

"selected" is the list of selected (= corresponds to 1 of nb) explanatory variables.

formula_tail: "objective_variable ~ explanatory_variable1 + explanatory_variable2 + …", and formula_tail is the right-hand side of formula.

The AIC of a linear multiple regression with the formula using the current combination of explanatory variables is assigned in aic and displayed with the formula.

```
        ...
        if min_score == None or min_score > aic:
                min_score = aic
                min_selected = selected.copy()

    formula = formula_head + ' + '.join(min_selected)
    print('The best formula: {}'.format(formula))
    print('Minimum AIC: {:.3f}'.format(min_score))
    return model(formula, **kwargs).fit()
```

The explanatory variables (selected) used in the first equation or in the equation with the smallest aic so far are stored with their aic value.

When the for loop finishes, display the formula for the smallest aic along with the aic value, and return the results of the linear multiple regression.

Execution example

abalone_modified.csv, explanatory variables: 'len', 'd', 'h', 'w_all' columns, objective variable: 'ring'

```
results = brute_force_aic(smf.ols, exog, endog, data=df_scaled)
```

```
AIC: 10406.185, formula: ring ~ w_all
AIC: 10293.403, formula: ring ~ h
AIC: 10164.339, formula: ring ~ h + w_all
AIC: 10175.294, formula: ring ~ d
AIC: 10173.998, formula: ring ~ d + w_all
AIC: 10048.947, formula: ring ~ d + h
AIC: 10050.341, formula: ring ~ d + h + w_all
AIC: 10299.876, formula: ring ~ len
AIC: 10274.645, formula: ring ~ len + w_all
AIC: 10117.269, formula: ring ~ len + h
AIC: 10115.510, formula: ring ~ len + h + w_all
AIC: 10151.690, formula: ring ~ len + d
AIC: 10145.454, formula: ring ~ len + d + w_all
AIC: 10017.609, formula: ring ~ len + d + h
AIC: 10019.510, formula: ring ~ len + d + h + w_all
The best formula: ring ~ len + d + h
Minimum AIC: 10017.609
```

Try all combinations at once.
Output the formula with the smallest AIC.

The best formula: ring ~ len + d + h (AIC: 10017.609)

step_aic()

step_aic_forward() corresponds to step_aic() with direction = 'forward'.

model: Regression model. Currently, linear regression, so smf.ols is assigned.
exog: List of explanatory variables.
endog: List of objective variable(s) (only one in this case).
direction: 'forward' for forward selection (#explanatory variables are increased), 'backward' for backward selection (#explanatory variables are decreased).
**kwards: Arguments for model.fit().

```
def step_aic(model, exog, endog, direction='forward', **kwargs):
    exog = np.r_[[exog]].flatten()
    endog = np.r_[[endog]].flatten()
    remaining = set(exog)
    selected = []  # Selected exogenous variables
    ...
```

Since we don't know what shape the arguments exog and endog are, we'll convert them to 1-d ndarray anyway

In the case of the step-forward method: the set of explanatory variables that have not yet been used is "remaining" (initial value: all explanatory variables), and the list of selected explanatory variables is "selected" (initial value: empty).
In the case of the step-backward method: the list of remaining explanatory variables is "remaining" (initial value: all explanatory variables), and the list of selected explanatory variables is selected (initially empty).

```
    formula_head = ' + '.join(endog) + ' ~ '
    if direction == 'forward':
        formula = formula_head + '1'
    elif direction == 'backward':
        formula = formula_head + ' + '.join(remaining)
        selected = remaining.copy()
    aic = model(formula=formula, **kwargs).fit().aic
    print('AIC: {:.3f}, formula: {}'.format(aic, formula))

    current_score = aic

...
```

First,
step-forward: create a formula with only constant term ("objective_variable ~ 1").
step-backward: create a formula with all explanatory variables and set "selected" to all explanatory variables.

Calculate the AIC for this formula, store it in the variable "aic", and display it with the formula. Set "current_score" (AIC of the current best (=lowest AIC) formula) to the value of "aic".

```
...
while True:
    score_with_candidates = []
    for candidate in remaining:
        # Calculate AIC for adding an exog one by one
        if direction == 'forward':
            formula_tail = ' + '.join(selected + [candidate])
        elif direction == 'backward':
            picked = remaining.copy()
            picked.remove(candidate)
            formula_tail = ' + '.join(picked)
        formula = formula_head + formula_tail
        aic = model(formula=formula, **kwargs).fit().aic
        print('AIC: {:.3f}, formula: {}'.format(aic, formula))

        score_with_candidates.append((aic, candidate))
    ...
```

score_with_candidates is the list with members of (aic, one explanatory variable)

Extract an explanatory variable from "remaining" to "candidate" one by one.

In the case of step-forward method:
Create formula_tail by adding "candidate" to the determined selected variables ("selected").

In the case of step-backward method:
Create formula_tail by deleting "candidate" from remaining variables ("remaining").

Calculate AIC and store it to "aic", and display it with the corresponding formula.

Append the following to score_with_candidates:
- step-forward: (aic, the corresponding added explanatory variable)
- step-backward: (aic, the corresponding deleted explanatory variable).

```
...
# Select best_candidate with minimum AIC
 score_with_candidates.sort()
 best_score, best_candidate = score_with_candidates[0]

 # select best_candidate if AIC is improved
 improved = False
 if best_score < current_score:
     remaining.remove(best_candidate)
     if direction == 'forward':
         selected.append(best_candidate)
     else:
         selected = remaining.copy()
     current_score = best_score
     improved = True

 if not remaining or not improved: break
 ...
```

Sort score_with_candidates.
(in ascending order of aic)

Store the smallest aic to "best_score",
and the corresponding explanatory
variable to "best_candidate".

If "best_score" is smaller than "current_score" (the current minimum AIC), remove the corresponding explanatory variable (best_candidate) from remaining and
step-forward method: add it to "selected".
(decided to be used)
step-backward method: copy "remaining" to "selected"
(decided to be deleted)
and then, set the value of "improved" to True.

If there are no explanatory variables left, or if adding/removing variables does not update the smallest value of the AIC, break out of while True.

When while True finishes …

```
…
formula = formula_head + ' + '.join(selected)
print('Direction:', direction)
print('The best formula: {}'.format(formula))
aic = model(formula=formula, **kwargs).fit().aic
print('Minimum AIC: {:.3f}'.format(aic))
return model(formula, **kwargs).fit()
```

Display the formula for the smallest aic along with the aic value, and return the results of the linear multiple regression.

Execution example (forward)

abalone_modified.csv, explanatory variables: 'len', 'd', 'h', 'w_all' columns, objective variable: 'ring'

```
results = step_aic(smf.ols, exog, endog, data=df_scaled, direction='forward')
```

```
AIC: 11847.299, formula: ring ~ 1
AIC: 10299.876, formula: ring ~ len
AIC: 10293.403, formula: ring ~ h
AIC: 10406.185, formula: ring ~ w_all
AIC: 10175.294, formula: ring ~ d
AIC: 10151.690, formula: ring ~ d + len
AIC: 10048.947, formula: ring ~ d + h
AIC: 10173.998, formula: ring ~ d + w_all
AIC: 10017.609, formula: ring ~ d + h + len
AIC: 10050.341, formula: ring ~ d + h + w_all
AIC: 10019.510, formula: ring ~ d + h + len + w_all
Direction: forward
The best formula: ring ~ d + h + len
Minimum AIC: 10017.609
```

Only constant

Try one explanatory variable
→ Best score is obtained for 'd' (determined to be used)

Try to add one explanatory variable with 'd'
→ Best score is obtained for 'h' (determined to be used)

Try to add one explanatory variable with 'd + h'
→ Best score is obtained for 'len' (determined to be used)

Try to add remaining 'w_all' to 'd + h + len'
→ The best score so far is not be updated. Also, we ran out of explanatory variables, so calculation is finished.

The best formula: ring ~ d + h + len (AIC: 10017.609)

Execution example (backward)

abalone_modified.csv, explanatory variables: 'len', 'd', 'h', 'w_all' columns, objective variable: 'ring'

```
results = step_aic(smf.ols, exog, endog, data=df_scaled, direction='backward')
```

AIC: 10019.510, formula: ring ~ h + len + w_all + d
AIC: 10145.454, formula: ring ~ len + w_all + d
AIC: 10050.341, formula: ring ~ h + w_all + d
AIC: 10017.609, formula: ring ~ h + len + d
AIC: 10115.510, formula: ring ~ h + len + w_all
AIC: 10151.690, formula: ring ~ d + len
AIC: 10048.947, formula: ring ~ d + h
AIC: 10117.269, formula: ring ~ len + h
Direction: backward
The best formula: ring ~ d + len + h
Minimum AIC: 10017.609

All explanatory variables (h + len + w_all + d)

Try to delete one explanatory variable from 'h + len + w_all + d'
→ The best score is obtained for 'w_all' (i.e. h + len + d).

Try to delete one explanatory variable from 'h + len + d'
→ The best score so far is not be updated, so calculation is finished.

The best formula: ring ~ d + len + h (AIC: 10017.609)