hw6nk_ipynb

October 5, 2024

[6]:
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import statsmodels.api as sm
import statsmodels.formula.api as smf
from itertools import combinations
```

[ ]:
```
df_hw6 = pd.read_csv('../data/HW6Q2.txt', delimiter=' ')
df_hw6.sample(10)
```

[21]:
```
y = df_hw6['Y']
X = df_hw6[['X1', 'X2', 'X3']]

X = sm.add_constant(X)

def best_subset_selection(X, y):
    n, p = X.shape
    models = []

    for k in range(1, p):
        for combo in combinations(range(1, p), k):
            combo = (0,) + combo
            X_subset = X.iloc[:, list(combo)]
            model = sm.OLS(y, X_subset).fit()
            models.append((model, combo))

    return models
```

[22]:
```
def calculate_metrics(model, X, y):
    n = len(y)
    k = model.df_model

    # AIC
    aic = model.aic

    # BIC
    bic = model.bic
```

```
    # PRESS (Prediction Sum of Squares)
    X_np = X.values
    try:
        inv_term = np.linalg.inv(X_np.T @ X_np)
    except np.linalg.LinAlgError:
        inv_term = np.linalg.pinv(X_np.T @ X_np)
    hat_matrix = X_np @ inv_term @ X_np.T
    residuals = model.resid
    press = np.sum((residuals / (1 - np.diag(hat_matrix))) ** 2)

    # Adjusted R-squared
    r2 = model.rsquared
    adj_r2 = 1 - (1 - r2) * (n - 1) / (n - k - 1)

    # Mallow's CP
    rss = np.sum(residuals ** 2)
    variance = np.var(y, ddof=1)
    cp = rss / variance - (n - 2 * (k + 1))

    return aic, bic, press, adj_r2, cp, int(k)
```

[24]:
```
models = best_subset_selection(X, y)

results = []
for model, combo in models:
    X_subset = X.iloc[:, list(combo)]
    aic, bic, press, adj_r2, cp, num_predictors = calculate_metrics(model,␣
 ↪X_subset, y)
    predictor_names = [X.columns[i] for i in combo if X.columns[i] != 'const']
    results.append({
        'Predictors': predictor_names,
        'n_Predictors': num_predictors,
        'AIC': aic,
        'BIC': bic,
        'PRESS': press,
        'Adjusted R^2': adj_r2,
        'Mallows CP': cp
    })

results_df = pd.DataFrame(results)
results_df = results_df.sort_values(by='n_Predictors').reset_index(drop=True)
pd.set_option('display.max_columns', None)
print(results_df)
```

```
    Predictors  n_Predictors         AIC         BIC       PRESS  \
0         [X1]             1  195.396205  199.220251  144.845842
1         [X2]             1  194.892016  198.716062  146.754946
2         [X3]             1  199.123971  202.948017  157.779166
```

```
3      [X1, X2]             2  192.844955  198.581024  139.805055
4      [X1, X3]             2  159.762207  165.498276   71.752846
5      [X2, X3]             2  196.591744  202.327813  152.066910
6  [X1, X2, X3]             3  159.865676  167.513768   72.565916

   Adjusted R^2  Mallows CP
0      0.056228   -0.698965
1      0.065697   -1.153475
2     -0.016824    2.807558
3      0.120008   -2.640387
4      0.545927  -22.658577
5      0.051532    0.578009
6      0.553324  -21.452911
```

[30]:
```
y = df_hw6['Y']
X = df_hw6[['X1', 'X3']]

X = sm.add_constant(X)

model = sm.OLS(y, X).fit()

influence = model.get_influence()
cooks_d, p_values = influence.cooks_distance

n = len(df_hw6)
threshold = 4 / n

influential_points = df_hw6[cooks_d > threshold]

print("Influential Points based on Cook's Distance (D > 4/n):")
print(influential_points)
```

```
Influential Points based on Cook's Distance (D > 4/n):
        Y     X1     X2     X3
2   10.50  10.96  11.32  -1.50
46  14.50  -3.77   6.39  14.30
48  16.75  -3.65  10.02  16.77
```

[31]:
```
iccdata = pd.read_csv('../data/IceCreamConsumption.csv')
iccdata.sample(10)
```

[31]:

	cons	income	price	temp	time
11	0.298	85	0.270	26	12
28	0.437	91	0.268	64	29
6	0.327	82	0.275	61	7
26	0.376	94	0.265	41	27
8	0.269	76	0.265	32	9
29	0.548	90	0.260	71	30

```
19  0.342      86  0.277    60    20
5   0.344      78  0.262    65     6
14  0.381      84  0.277    55    15
17  0.443      78  0.277    72    18
```

[32]:
```
y = iccdata['cons']
X = iccdata[['income', 'price', 'temp']]

predictor_names = ['income', 'price', 'temp']
all_models = []

for k in range(1, len(predictor_names) + 1):
    for combo in combinations(predictor_names, k):
        all_models.append(list(combo))

for idx, model in enumerate(all_models, 1):
    print(f"Model {idx}: Predictors = {model}")
```

```
Model 1: Predictors = ['income']
Model 2: Predictors = ['price']
Model 3: Predictors = ['temp']
Model 4: Predictors = ['income', 'price']
Model 5: Predictors = ['income', 'temp']
Model 6: Predictors = ['price', 'temp']
Model 7: Predictors = ['income', 'price', 'temp']
```

[33]:
```
y = iccdata['cons']
X = iccdata[['income', 'price', 'temp']]

predictor_names = ['income', 'price', 'temp']
all_models = []

for k in range(1, len(predictor_names) + 1):
    for combo in combinations(predictor_names, k):
        all_models.append(list(combo))

def calculate_cp(rss, sigma_squared, p, n):
    return (rss / sigma_squared) - (n - 2 * p)

results = []

n = len(y)
sigma_squared = np.var(y, ddof=1)

for model_predictors in all_models:
    X_model = sm.add_constant(iccdata[list(model_predictors)])
    model = sm.OLS(y, X_model).fit()
    adj_r2 = model.rsquared_adj
```

```
    rss = np.sum(model.resid ** 2)
    p = len(model_predictors)
    cp = (rss / sigma_squared) - (n - 2 * p)
    results.append({
        'Number of Predictors': p,
        'Adjusted R^2': adj_r2,
        "Mallow's C_p": cp,
        'Predictors': model_predictors
    })
summary_df = pd.DataFrame(results)

print("\nSummary:")
print(summary_df)
```

```
Summary:
   Number of Predictors  Adjusted R^2  Mallow's C_p              Predictors
0                     1     -0.033334      0.933364                [income]
1                     1      0.034082     -0.954282                 [price]
2                     1      0.587365    -16.446210                  [temp]
3                     2     -0.001257      1.033942         [income, price]
4                     2      0.679989    -17.359708          [income, temp]
5                     2      0.605619    -15.351717           [price, temp]
6                     3      0.686570    -15.850822  [income, price, temp]
```

[34]:
```
y_model1 = iccdata['cons']
X_model1 = iccdata[['income', 'price', 'temp']]
X_model1 = sm.add_constant(X_model1)

model1 = sm.OLS(y_model1, X_model1).fit()

y_model2 = iccdata['cons']
X_model2 = iccdata[['income']]
X_model2 = sm.add_constant(X_model2)

model2 = sm.OLS(y_model2, X_model2).fit()

aic_model1 = model1.aic
bic_model1 = model1.bic

aic_model2 = model2.aic
bic_model2 = model2.bic

summary_table = pd.DataFrame({
    'Model': ['[income, price, temp]', '[income]'],
    'Number of Predictors': [3, 1],
    'AIC': [aic_model1, aic_model2],
```

```
    'BIC': [bic_model1, bic_model2]
})

print("\nSummary Table")
print(summary_table)
```

```
Summary Table
                    Model  Number of Predictors         AIC         BIC
0  [income, price, temp]                     3 -109.238872 -103.634082
1               [income]                     1  -75.226523  -72.424129
```