Chapters 6 and 7: Python Workbook

Ahmad Tahmid

Contents

1 Chapter 6 Solutions

1.1 Exercise 6.6.1

Goal

Perform best subset selection (or stepwise selection) on a dataset like `Hitters` to predict `Salary`, and compare different models.

Solution

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
from itertools import combinations

# Example: using the Hitters data from ISLP or a local CSV
Hitters = pd.read_csv("Hitters.csv").dropna()

# We'll do a basic best subset selection for up to a certain number
   of predictors
def best_subset_selection(data, response, max_features=8):
    """
    Exhaustively search all combinations of predictors up to
       max_features in size.
    Returns a dictionary with the best model (lowest AIC) for each
       subset size.
    """
    y = data[response]
    X_all = data.drop(columns=[response])
    predictors = X_all.columns

    results = {}

    for k in range(1, max_features+1):
        best_aic = np.inf
        best_combo = None
        best_model = None

        for combo in combinations(predictors, k):
            Xk = X_all[list(combo)]
            # Add intercept
            Xk_const = sm.add_constant(Xk)
            model = sm.OLS(y, Xk_const).fit()
            if model.aic < best_aic:
                best_aic = model.aic
                best_combo = combo
                best_model = model
```

```
        results[k] = {"predictors": best_combo, "model": best_model}
        print(f"Best subset of size {k}: {best_combo}, AIC={
            best_model.aic:.2f}")
    return results

subset_results = best_subset_selection(Hitters, "Salary",
    max_features=8)
# We now have the best models (by AIC) for k=1..8
# We can inspect them or pick the best overall model.
```

1.2 Exercise 6.6.2

Goal

Apply ridge and lasso regression to a dataset (e.g., `Hitters`), select the best λ via cross-validation, and examine coefficient shrinkage.

Solution

```
import pandas as pd
import numpy as np
from sklearn.linear_model import Ridge, Lasso
from sklearn.model_selection import GridSearchCV

Hitters = pd.read_csv("Hitters.csv").dropna()

y = Hitters["Salary"].values
X = Hitters.drop(columns=["Salary", "Name"])  # or exclude any non-
    numeric columns
# Convert categoricals if present
X = pd.get_dummies(X, drop_first=True)

# Ridge
alpha_values = np.logspace(-2, 4, 50)  # range of lambdas
ridge = Ridge()
param_grid = {"alpha": alpha_values}
ridge_cv = GridSearchCV(ridge, param_grid, scoring='
    neg_mean_squared_error', cv=10)
ridge_cv.fit(X, y)

print("Best alpha (Ridge):", ridge_cv.best_params_)
print("MSE:", -ridge_cv.best_score_)

# Retrieve best model
ridge_best = ridge_cv.best_estimator_
print("Ridge coefficients:\n", ridge_best.coef_)
```

```
# Lasso
lasso = Lasso(max_iter=10000)
param_grid_lasso = {"alpha": alpha_values}
lasso_cv = GridSearchCV(lasso, param_grid_lasso, scoring='
   neg_mean_squared_error', cv=10)
lasso_cv.fit(X, y)

print("Best alpha (Lasso):", lasso_cv.best_params_)
print("MSE:", -lasso_cv.best_score_)

# Retrieve best model
lasso_best = lasso_cv.best_estimator_
print("Lasso coefficients:\n", lasso_best.coef_)
```

1.3 Exercise 6.6.3

Goal

Perform PCR (Principal Components Regression) and PLS on a dataset, compare test errors for different numbers of components.

Solution

```
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import KFold
from sklearn.cross_decomposition import PLSRegression

Hitters = pd.read_csv("Hitters.csv").dropna()
y = Hitters["Salary"].values
X = Hitters.drop(columns=["Salary", "Name"])
X = pd.get_dummies(X, drop_first=True).values

# PCR approach with k-fold CV
def pcr_cv(X, y, max_components=10):
    n_splits = 10
    kf = KFold(n_splits=n_splits, shuffle=True, random_state=1)

    for n_comp in range(1, max_components+1):
        fold_mse = []
        for train_idx, test_idx in kf.split(X):
            X_train, X_test = X[train_idx], X[test_idx]
            y_train, y_test = y[train_idx], y[test_idx]
```

```
            pca = PCA(n_components=n_comp)
            X_train_pca = pca.fit_transform(X_train)
            X_test_pca  = pca.transform(X_test)

            lm = LinearRegression().fit(X_train_pca, y_train)
            preds = lm.predict(X_test_pca)
            fold_mse.append(np.mean((y_test - preds)**2))
        print(f"PCR: n_components={n_comp}, MSE={np.mean(fold_mse)
            :.2f}")

pcr_cv(X, y, max_components=10)

# PLS approach with k-fold CV
def pls_cv(X, y, max_components=10):
    n_splits = 10
    kf = KFold(n_splits=n_splits, shuffle=True, random_state=2)

    for n_comp in range(1, max_components+1):
        fold_mse = []
        for train_idx, test_idx in kf.split(X):
            X_train, X_test = X[train_idx], X[test_idx]
            y_train, y_test = y[train_idx], y[test_idx]

            pls = PLSRegression(n_components=n_comp)
            pls.fit(X_train, y_train)
            preds = pls.predict(X_test).ravel()
            fold_mse.append(np.mean((y_test - preds)**2))
        print(f"PLS: n_components={n_comp}, MSE={np.mean(fold_mse)
            :.2f}")

pls_cv(X, y, max_components=10)
```

2 Chapter 7 Solutions

2.1 Exercise 7.9.1

Goal

Fit polynomial regression and piecewise step functions to a dataset (e.g., `Wage`), compare results, and visualize.

Solution

```
import pandas as pd
import numpy as np
import statsmodels.formula.api as smf
```

```
import matplotlib.pyplot as plt

# Suppose we have the Wage data
Wage = pd.read_csv("Wage.csv")

# 1. Polynomial regression of wage on age (degree=4)
poly_model = smf.ols("wage ~ age + I(age**2) + I(age**3) + I(age**4)
   ", data=Wage).fit()
print(poly_model.summary())

# Plot
age_grid = np.linspace(Wage["age"].min(), Wage["age"].max(), 100)
pred_df = pd.DataFrame({"age": age_grid})
pred_df["pred_wage"] = poly_model.predict(pred_df)

plt.scatter(Wage["age"], Wage["wage"], alpha=0.5)
plt.plot(age_grid, pred_df["pred_wage"], color='red', linewidth=2)
plt.xlabel("Age")
plt.ylabel("Wage")
plt.title("Polynomial Regression of Wage ~ Age")
plt.show()

# 2. Piecewise Step Function (example with cutpoints at 25, 40, 60)
Wage["age_cut"] = pd.cut(Wage["age"], bins=[Wage["age"].min()
   ,25,40,60,Wage["age"].max()])
step_model = smf.ols("wage ~ age_cut", data=Wage).fit()
print(step_model.summary())
```

2.2 Exercise 7.9.2

Goal

Fit splines for age and interpret the results. Plot the spline fit.

Solution

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
from patsy import dmatrix
import matplotlib.pyplot as plt

Wage = pd.read_csv("Wage.csv")

# Create a spline basis with patsy
# Let's specify 3 knots at ages 25, 40, 60
age = Wage["age"]
```

```
knots = [25, 40, 60]

transformed_x = dmatrix(
    "bs(age, knots=(25,40,60), degree=3, include_intercept=False)",
    {"age": age}, return_type='dataframe'
)
spline_model = sm.OLS(Wage["wage"], transformed_x).fit()
print(spline_model.summary())

# Predict on a grid
age_grid = np.linspace(age.min(), age.max(), 100)
X_grid = dmatrix(
    "bs(age, knots=(25,40,60), degree=3, include_intercept=False)",
    {"age": age_grid}, return_type='dataframe'
)
pred = spline_model.predict(X_grid)

plt.scatter(age, Wage["wage"], alpha=0.3, label="Data")
plt.plot(age_grid, pred, color='red', linewidth=2, label="Spline Fit
   ")
plt.xlabel("Age")
plt.ylabel("Wage")
plt.title("Cubic Spline of Wage ~ Age")
plt.legend()
plt.show()
```

2.3 Exercise 7.9.3

Goal

Use local regression or smoothing splines. Compare to polynomial or splines. Possibly consider a GAM with age and year or another variable as smooth functions.

Solution

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.nonparametric.smoothers_lowess import lowess

Wage = pd.read_csv("Wage.csv")
age = Wage["age"].values
wage = Wage["wage"].values

# 1. Local regression (LOWESS)
lowess_fit = lowess(wage, age, frac=0.3)  # frac controls smoothing
x_lowess = lowess_fit[:,0]
```

```
y_lowess = lowess_fit[:,1]

plt.scatter(age, wage, alpha=0.3, label="Data")
plt.plot(x_lowess, y_lowess, color='red', linewidth=2, label="LOWESS
    ")
plt.xlabel("Age")
plt.ylabel("Wage")
plt.title("Local Regression (LOWESS)")
plt.legend()
plt.show()

# 2. Example: a simple GAM using 'year' as well (if 'gam' or 'pyGAM'
     is installed)
# This snippet requires the 'pyGAM' library. If not available, skip
    or adapt.
"""
from pygam import LinearGAM, s

X = Wage[["year","age"]]
y = Wage["wage"]

gam = LinearGAM(s(0) + s(1)).fit(X, y)
XX = pd.DataFrame({
    "year": np.linspace(X["year"].min(), X["year"].max(), 100),
    "age": np.linspace(X["age"].min(), X["age"].max(), 100)
})
# We'll do partial dependence or just pick one variable to vary
"""
```

3 My Learnings

- performed best subset selection, ridge, lasso, and partial/complete cross-validation to compare model fits
- explored dimension-reduction methods (PCR, PLS) to handle potentially high-dimensional settings
- worked with polynomial expansions, step functions, and spline bases to move beyond linear regression
- used local regression (LOWESS) for non-parametric smoothing
- recognized how these methods can capture complex relationships while balancing interpretability