MATH 448

CH 6 HW

ANDREW DAHLSTROM

4/25/24

1)

a)

Best subset selection typically has the smallest training RSS because it evaluates all possible combinations of predictors which allows it to find the combination that minimizes the training RSS for each k predictor.

b)

The model that has the smallest RSS for the test set depends largely on the structure of the data. Best subset selection has more of a tendency to overfit the training set than forward and backward stepwise selection meaning that it has a greater chance to perform worse on the testing set that the other two. c)

i. True, predictors are added stepwise based on which will contribute the most improvement in the model so the set of predictors in the k-variable model will always be a subset of the predictors in the k+1 model.

ii. True, the model begins with all predictors which are then removed stepwise based on the predictor that contributes the least improvement so any k-variable model will be a subset of the predictors in the k+1 variable model.

iii. False, the selection process for forward and backward stepwise are unique so the predictors selected can be entirely different because the different methods consider different criteria in the model development.

iv. False, there is no guarantee best subset will select the k predictors of a model that are a subset of the k+1 predictors of another model. The model considers all possible combinations of k predictors independently.

2)

a) iii.

Lasso is less flexible so when it is underfitting the training data its increase in bias is less that its decrease in variance from the reduced model complexity.

b) iii.

Ridge is less flixible and will increase the prediction accurancy when the increase in bias is less than the decrease in variance. It benefits from lowering variance significantly and increasing bias only a small amount.

c) i.

The non-linear methods are much more flexible relative to least squares.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_model import LinearRegression, RidgeCV, LassoCV
from sklearn.preprocessing import scale
from sklearn.decomposition import PCA
from sklearn.cross_decomposition import PLSRegression
from sklearn.metrics import mean_squared_error
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler


# Load dataset
college_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/MATH 448/College.csv')
print(college_df.head())
```

```
                        Unnamed: 0 Private  Apps  Accept  Enroll  Top10perc  \
0  Abilene Christian University     Yes  1660    1232     721         23
```

```
1           Adelphi University     Yes  2186    1924     512          16
2              Adrian College     Yes  1428    1097     336          22
3          Agnes Scott College     Yes   417     349     137          60
4     Alaska Pacific University     Yes   193     146      55          16

   Top25perc  F.Undergrad  P.Undergrad  Outstate  Room.Board  Books  Personal  \
0         52         2885          537      7440        3300    450      2200
1         29         2683         1227     12280        6450    750      1500
2         50         1036           99     11250        3750    400      1165
3         89          510           63     12960        5450    450       875
4         44          249          869      7560        4120    800      1500

   PhD  Terminal  S.F.Ratio  perc.alumni  Expend  Grad.Rate
0   70        78       18.1           12    7041         60
1   29        30       12.2           16   10527         56
2   53        66       12.9           30    8735         54
3   92        97        7.7           37   19016         59
4   76        72       11.9            2   10922         15
```

```
# Drop non numerical columns
college_df = college_df.drop(['Unnamed: 0', 'Private'], axis=1)
college_df.head()
```

	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	Outstate	Room.Board	Books	Personal	PhD	Terminal	S.I
0	1660	1232	721	23	52	2885	537	7440	3300	450	2200	70	78	
1	2186	1924	512	16	29	2683	1227	12280	6450	750	1500	29	30	
2	1428	1097	336	22	50	1036	99	11250	3750	400	1165	53	66	
3	417	349	137	60	89	510	63	12960	5450	450	875	92	97	
4	193	146	55	16	44	249	869	7560	4120	800	1500	76	72	

Next steps: Generate code with college_df   View recommended plots

```
# Create predictors and response
X = college_df.drop('Apps', axis=1)
y = college_df['Apps']
# scale predictors
scaler = StandardScaler()
X = scaler.fit_transform(X)
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=2)


# Fit a linear model
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)

# Predict and calculate the test error
y_pred_linear = linear_model.predict(X_test)
error_linear = mean_squared_error(y_test, y_pred_linear)
print("Linear Model MSE", error_linear)
```

```
Linear Model MSE 1760383.4479113745
```

```
# Fit a ridge regression model with cross-validation
ridge_model = RidgeCV(alphas=np.logspace(-6, 6, 13))
ridge_model.fit(X_train, y_train)

# Predict and calculate the test error
y_pred_ridge = ridge_model.predict(X_test)
error_ridge = mean_squared_error(y_test, y_pred_ridge)
print("Ridge Regression Model MSE", error_ridge)
```

```
Ridge Regression Model MSE 1760381.3967166764
```

```
print("Alpha used", ridge_model.alpha_)
```

```
Alpha used 1e-06
```

```
# Fit a lasso model with cross-validation
lasso_model = LassoCV(cv=5)
lasso_model.fit(X_train, y_train)

# Predict and calculate the test error
y_pred_lasso = lasso_model.predict(X_test)
error_lasso = mean_squared_error(y_test, y_pred_lasso)
print("Lasso Model MSE", error_lasso)
print("Number of Non-Zero Coefficients", np.sum(lasso_model.coef_ != 0))
```

```
Lasso Model MSE 1804853.0814868803
Number of Non-Zero Coefficients 14
```

```
# Combine PCA and linear regression to create PCR

# Define a range for the number of PCA components
n_components = np.arange(1, min(X_train.shape[1], X_train.shape[0]) + 1)
mse_scores = []

# Loop over possible numbers of components
for n in n_components:
    pca = PCA(n_components=n)
    linear_reg = LinearRegression()
    pcr_model = make_pipeline(pca, linear_reg)

    # Perform cross-validation
    scores = cross_val_score(pcr_model, X_train, y_train, cv=5, scoring='neg_mean_squared_error')
    mse_scores.append(-np.mean(scores))

# Find the number of components with the lowest MSE
optimal_n = n_components[np.argmin(mse_scores)]
print("Optimal number of components", optimal_n)

# Fit PCR model on training data using the optimal number of components
pca = PCA(n_components=optimal_n)
linear_reg = LinearRegression()
pcr_model = make_pipeline(pca, linear_reg)
pcr_model.fit(X_train, y_train)

# Predict and calculate the test error
y_pred_pcr = pcr_model.predict(X_test)
error_pcr = mean_squared_error(y_test, y_pred_pcr)
print("PCR Model MSE", error_pcr)
```

```
Optimal number of components 16
PCR Model MSE 1760383.447911378
```

```
# Use PLS and find optimal number of PLS components
# Define a range for the number of PLS components
n_components = np.arange(1, min(X_train.shape[1], X_train.shape[0]) + 1)
mse_scores = []

# Loop over possible numbers of components
for n in n_components:
    pls = PLSRegression(n_components=n)

    # Perform cross-validation
    scores = cross_val_score(pls, X_train, y_train, cv=5, scoring='neg_mean_squared_error')
    mse_scores.append(-np.mean(scores))  # Store average MSE (note: scores are negative MSE)

# Find the number of components with the lowest MSE
optimal_n = n_components[np.argmin(mse_scores)]
print("Optimal number of components", optimal_n)

# Fit PLS model on training data using the optimal number of components
pls = PLSRegression(n_components=optimal_n)
pls.fit(X_train, y_train)

# Predict and calculate the test error
y_pred_pls = pls.predict(X_test)
error_pls = mean_squared_error(y_test, y_pred_pls)
print("PLS Model MSE", error_pls)
```

```
Optimal number of components 11
PLS Model MSE 1764873.400419569
```

g)

Both linear and ridge produced very similar MSEs and the alpha value for ridge was very small so it basically functioned as a linear regression. The lasso performed slightly worse than linear and ridge suggesting that some important predictors were shrunk to 0 considering there were 14 non-zero components. The PCR model had a similar MSE to linear and ridge and used 16 components suggesting that most predictors contributed to the response. The PLS model performed the worst and used 11 components optimally. This sugguests again that most of the predictors are useful in this task and that reducing the contributions of the predictors results is worse model performance for this task.