HW3

20200639 Chae Woojin

Chapter 6: 4, 5, 7, 9(excluding (e), (f)), 10

Chapter 7: 1, 2, 4, 10, 11

Chapter 6

4.

(a) iii. As λ increases, model flexibility decreases. This leads to the increase of training RSS.

(b) ii. Test RSS will decrease until decreasing variance overwhelms increasing bias. Then, it will bounce back as effect of increasing bias wins the effect of decreasing variance.

(c) iv. As λ increases, the model becomes less flexible, so the variance steadily decreases.

(d) iii. However, in terms of bias, it steadily increases as model becomes less flexible,

(e) v. The irreducible error is unchanged.

5.

(a)

$$\sum_{i=1}^{2}\left(y_i-\beta_0-\sum_{j=1}^{2}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{2}\beta_j^2$$

(b)

Let $f(\beta_1,\beta_2)=\sum_{i=1}^{2}\left(y_i-\beta_0-\sum_{j=1}^{2}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{2}\beta_j^2$. Then,

$$\frac{df}{d\beta_1}=-2x_{11}\left(y_1-\beta_0-\beta_1x_{11}-\beta_2x_{12}\right)-2x_{21}\left(y_2-\beta_0-\beta_1x_{21}-\beta_2x_{22}\right)+2\lambda\beta_1$$

$$\frac{df}{d\beta_2}=-2x_{12}\left(y_1-\beta_0-\beta_1x_{11}-\beta_2x_{12}\right)-2x_{22}\left(y_2-\beta_0-\beta_1x_{21}-\beta_2x_{22}\right)+2\lambda\beta_2.$$

A minimum can be found when these are set to 0.

As $x_{11}=x_{12},x_{21}=x_{22},\widehat{\beta_0}=0$, solution of $\frac{df}{d\beta_1}=0$, and $\frac{df}{d\beta_2}=0$ is same.

$$\frac{df}{d\beta_1}=0\Leftrightarrow\lambda\beta_1=y_1x_{11}+y_2x_{21}-\left(\beta_1+\beta_2\right)\left(x_{11}^2+x_{21}^2\right)$$

$$\frac{df}{d\beta_2} = 0 \Leftrightarrow \lambda\beta_2 = y_1x_{11} + y_2x_{21} - (\beta_1 + \beta_2)(x_{11}^2 + x_{21}^2)$$

$$\lambda\beta_1 = \lambda\beta_2 \iff \widehat{\beta_1} = \widehat{\beta_2}$$

(c)

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{2}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{2}|\beta_j|$$

(d)

Let $g(\beta_1, \beta_2) = \sum_{i=1}^{2}(y_i - \beta_0 - \sum_{j=1}^{2}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{2}|\beta_j|$. Then, as $\widehat{\beta_0} = 0$, we can simplify partial derivative in regard to each coefficient as

$$\frac{df}{d\beta_1} = -2x_{11}(y_1 - \beta_1x_{11} - \beta_2x_{12}) - 2x_{21}(y_2 - \beta_1x_{21} - \beta_2x_{22}) + \lambda\partial|\beta_1|,$$

$$\frac{df}{d\beta_2} = -2x_{11}(y_1 - \beta_0 - \beta_1x_{11} - \beta_2x_{12}) - 2x_{21}(y_2 - \beta_1x_{21} - \beta_2x_{22}) + \lambda\partial|\beta_2|,$$

where $\partial|\beta_1|$, $\partial|\beta_2|$ denotes subdifferential of $|\beta_1|$, and $|\beta_2|$ respectively.

Then, as partial derivatives contain the subgradient in its term, it is well known that

$$\widehat{\beta_1} = \begin{cases} \frac{1}{2}(L - \lambda)\ if\ L > \lambda, \\ 0\ if\ \lambda > L > -\lambda, \\ \frac{1}{2}(L + \lambda)\ if\ -\lambda > L \end{cases}$$

where $L = -2x_{11}(y_1 - \beta_0 - \beta_1x_{11} - \beta_2x_{12}) - 2x_{21}(y_2 - \beta_0 - \beta_1x_{21} - \beta_2x_{22})$

$\widehat{\beta_2}$ is evaluated similarly.

From this point, one can conclude that we cannot denote its solution uniquely.

7.

(a)

$$L = \prod_{i=1}^{n} N(0,\ \sigma^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\varepsilon_i^2}{2\sigma^2}\right)$$

$$= \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n} \varepsilon_i^2\right)$$

(b)

The posterior probability can be calculated by multiplying the prior and likelihood.

$$p(\beta|X,Y) \propto p(X,Y|\beta)p(\beta)$$

$$\propto \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n} \varepsilon_i^2\right) \prod_{j=1}^{p} \frac{1}{2b} \exp\left(-\frac{|\beta_j|}{b}\right)$$

$$\propto \left(\frac{1}{2b}\right)^p \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n} \varepsilon_i^2 - \sum_{j=1}^{p} \frac{|\beta_j|}{b}\right)$$

(c)

Mode of β can be achieved by finding β which maximizes the likelihood of posterior probability.

$$\log(p(\beta|X,Y)) \propto \log\left[\left(\tfrac{1}{2b}\right)^p \left(\tfrac{1}{\sqrt{2\pi}\sigma}\right)^n\right] - \tfrac{1}{2\sigma^2}\textstyle\sum_{i=1}^{n} \varepsilon_i^2 - \sum_{j=1}^{p} \frac{|\beta_j|}{b}.$$

As first term is independent of β, our solution will be when we maximize the second term.

$$argmax_\beta \left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n} \varepsilon_i^2 - \sum_{j=1}^{p} \frac{|\beta_j|}{b}\right)$$

$$= argmin_\beta \left(\frac{1}{2\sigma^2}\sum_{i=1}^{n} \varepsilon_i^2 + \sum_{j=1}^{p} \frac{|\beta_j|}{b}\right)$$

$$= argmin_\beta \left(\sum_{i=1}^{n} \varepsilon_i^2 + \frac{2\sigma^2}{b}\sum_{j=1}^{p} |\beta_j|\right)$$

As $RSS = \sum_{i=1}^{n} \varepsilon_i^2$ and if we set $\lambda = \frac{2\sigma^2}{b}$, the mode corresponds to lasso optimization.

$$argmin_\beta \left(RSS + \lambda\sum_{j=1}^{p} |\beta_j|\right)$$

(d)

$$p(\beta|X,Y) \propto \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}\varepsilon_i^2\right)\prod_{j=1}^{p}\frac{1}{\sqrt{2\pi c}}\exp\left(-\frac{\beta_j^2}{2c}\right)$$

$$\propto \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n \left(\frac{1}{\sqrt{2\pi c}}\right)^{\mathrm{p}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}\varepsilon_i^2 - \frac{1}{2c}\sum_{j=1}^{p}\beta_j^2\right)$$

(e)

To show that the ridge estimate is the mode, we can again find the maximum by maximizing the log of the posterior probability. The log is

$$\log(\beta|X,Y) \propto \log\left[\left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n \left(\frac{1}{\sqrt{2\pi c}}\right)^{\mathrm{p}}\right] - \left(\frac{1}{2\sigma^2}\sum_{i=1}^{n}\varepsilon_i^2 + \frac{1}{2c}\sum_{j=1}^{p}\beta_j^2\right)$$

As the first term is independent of β, it suffices to consider finding β that minimizes $\frac{1}{2\sigma^2}\sum_{i=1}^{n}\varepsilon_i^2 + \frac{1}{2c}\sum_{j=1}^{p}\beta_j^2$.

$$argmin_\beta\left(\frac{1}{2\sigma^2}\sum_{i=1}^{n}\varepsilon_i^2 + \frac{1}{2c}\sum_{j=1}^{p}\beta_j^2\right)$$

$$= argmin_\beta\left(\sum_{i=1}^{n}\varepsilon_i^2 + \frac{\sigma^2}{c}\sum_{j=1}^{p}\beta_j^2\right)$$

As $RSS = \sum_{i=1}^{n}\varepsilon_i^2$, if we set $\lambda = \frac{\sigma^2}{c}$, this term becomes $argmin_\beta\left(RSS + \lambda\sum_{j=1}^{p}\beta_j^2\right)$, which is ridge estimation. In sum, mode corresponds to the ridge regression estimate for β under this posterior distribution.

Next, to show that ridge estimate is the mean of the posterior probability, lets note that posterior probability is also gaussian function, because both likelihood and prior are also gaussian.

$$p(\beta|X,Y) \propto p(X,Y|\beta)p(\beta)$$

It is well known fact that mode=maximum=mean in gaussian. Therefore, mean also corresponds to the ridge regression estimate for β under this posterior distribution.

9.

(a)

```
[131] x_train, x_test, y_train, y_test = train_test_split(
          np.array(df.loc[:, ~df.columns.isin(['Apps'])]),
          np.array(df.loc[:, 'Apps']),
          random_state=0
      )

      print(len(x_train))
      print(len(x_test))
      print(len(y_train))
      print(len(y_test))
```

```
582
195
582
195
```

(b)

```
[132] lr = LinearRegression()
      lr.fit(x_train, y_train)

      # print Test MSE
      print(np.mean(((lr.predict(x_test)-y_test)**2)))
      print(lr.score(x_test, y_test))
```

```
1022430.0889255599
0.9002392990734506
```

Test error was about 1022430.

(c)

```
[137] alpha_cands = np.linspace(.01, 100, 1000)

      rcv = RidgeCV(alphas=alpha_cands,  cv=10)
      rcv.fit(x_train, y_train)

      print(np.mean(((rcv.predict(x_test)-y_test)**2)))
      print(rcv.score(x_test, y_test))
```

```
1018023.2320525252
0.9006692855686631
```

Optimal lambda chosen by cross validation was 39.345, and test error was about 1018023.

```
[139] print('optimal alpha:', rcv.alpha_)
```

```
optimal alpha: 39.3454054054054
```

(d)

```
[138] alpha_cands = np.linspace(.01, 100, 1000)

      lcv = LassoCV(alphas=alpha_cands,  cv=10)
      lcv.fit(x_train, y_train)

      print(np.mean(((lcv.predict(x_test)-y_test)**2)))
      print(lcv.score(x_test, y_test))
```

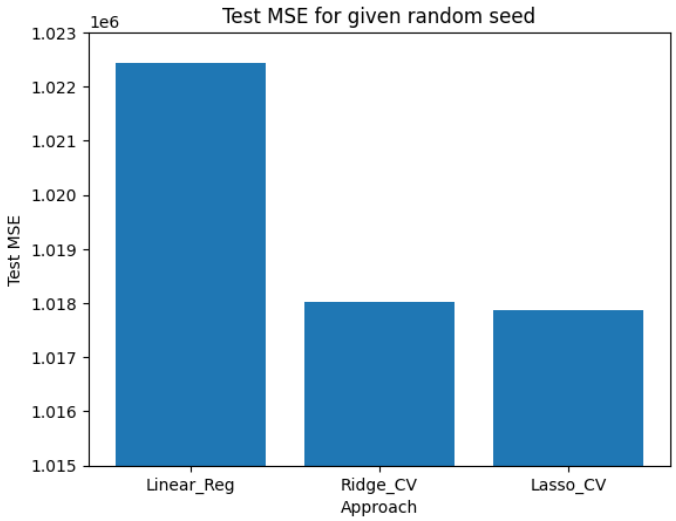
```
1017878.7762435534
0.9006833804323726
```

```
print('optimal alpha:', lcv.alpha_)
print('number of non-zero features: {0} (out of {1}): '
      .format(len([item for item in lcv.coef_ if item !=0]), x_test.shape[1]))
```

```
optimal alpha: 14.623153153153151
number of non-zero features: 16 (out of 18):
```

Optimal lambda chosen by cross validation was 14.623, and test error was about 1017878. Also, the number of non-zero coefficient estimates were 16 out of 18.

(g)



Smallest test error was obtained by lasso model in this given random seed. Ridge model also gave comparable performance against normal linear regression. I think this is because these shrinkage models were known to reduce variance of the estimates by leaving explanatory terms (removing colinear estimates or noise terms), so that they are likely to work better on the new data.

10.

(a)

```
[2]  np.random.seed(0)

     # Dataframe with random numbers and the specified dimensions
     n = 1000
     p = 20
     X = pd.DataFrame(np.random.normal(size=(n, p)))

     # Epsilon
     epsilon = np.random.normal(size=n)

     # Coefficient b1
     b1 = [5, 4, 3, 2, 0, 0, 0, 0, 0, 0,
           0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

     # Final expression
     # y must be a vector with 1000 rows.
     y = np.dot(X, b1) + epsilon
```

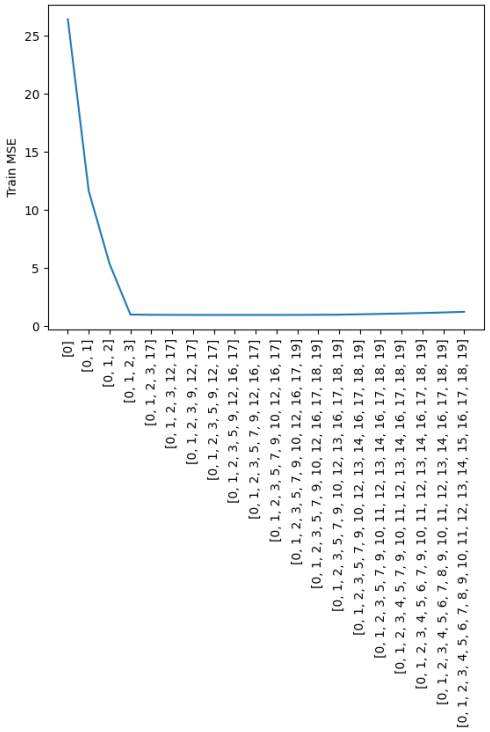
(b)

```
[ ]  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .9)
```

```
[ ]  len(X_test)
```

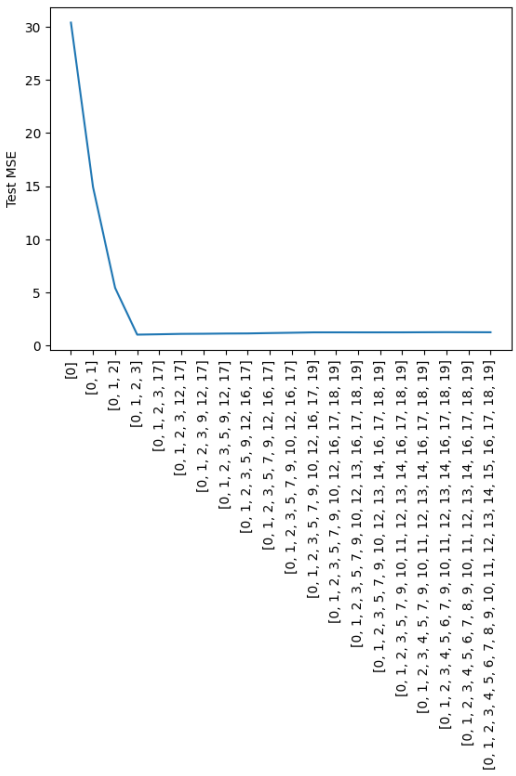
```
900
```

(c)



X-tick represents the selected best model.

(d)



X-tick represents the selected best model.

(e)

```
pd.Series(test_mse).idxmin()+1
```

```
4
```

The minimum test MSE is found when model size is 4. This corresponds to the true data which has 4 non-zero values. (I added one in the result because index starts with 0.)

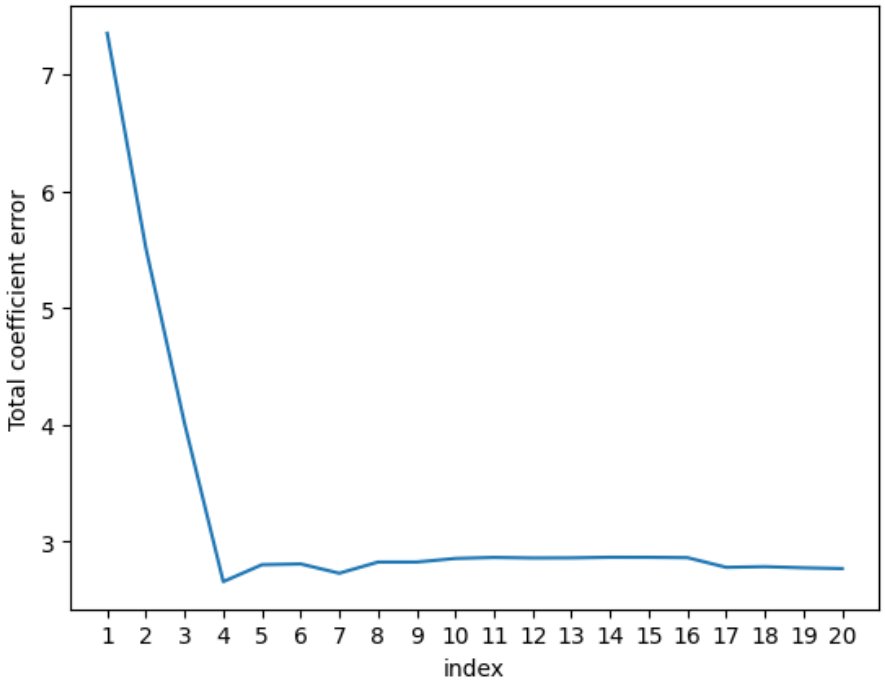
(f)

```
lr = LinearRegression()
lr.fit(X_train[best_subset[3]], y_train)
print(lr.coef_)
print(b1)
```

```
[5.0074071  3.89420905 2.9945604  2.0882942 ]
[5, 4, 3, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

Coefficient values between two model (best model's coefficients vs real coefficients) resembles clearly.

(g)



The total error of coefficient estimates is minimized when model size is 4. It corresponds to the size where test MSE is minimized.

Chapter 7

1. $f_1(x) = a_1 + b_1x + c_1x^2 + d_1x^3$

(a)

$$f(x) = \beta_0 + \beta_1x + \beta_2x^2 + \beta_3x^3 + \beta_4(x-\zeta)_+^3 \text{ with } (x-\zeta)_+^3 = \begin{cases} (x-\zeta)_+^3 \; if \; x > \zeta \\ 0 \; otherwise \end{cases}$$

For $x \leq \zeta$, set $a_1 = \beta_0,\ b_1 = \beta_1,\ c_1 = \beta_2,\ d_1 = \beta_3$, then $f_1(x) = f(x)$

(b)

For $x > \zeta$, set $a_1 = (\beta_0 - \beta_4\zeta^3),\ b_1 = (\beta_1 + 3\zeta^2\beta_4),\ c_1 = (\beta_2 - 3\zeta\beta_4),\ d_1 = (\beta_3 + \beta_4)$, then $f_1(x) = f(x)$

(c)

$f_1(\zeta) = \beta_0 + \beta_1\zeta + \beta_2\zeta^2 + \beta_3\zeta^3,$

$f_2(\zeta) = (\beta_0 - \beta_4\zeta^3) + (\beta_1 + 3\zeta^2\beta_4)\zeta + (\beta_2 - 3\zeta\beta_4)\zeta^2 + (\beta_3 + \beta_4)\zeta^3 = \beta_0 + \beta_1\zeta + \beta_2\zeta^2 + \beta_3\zeta^3$

$\therefore f_1(\zeta) = f_2(\zeta),\ f(x)$ is continuous at ζ.

(d)

$f_1'(x) = \beta_1 + 2\beta_2x + 3\beta_3x^2,$

$f_1'(\zeta) = \beta_1 + 2\beta_2\zeta + 3\beta_3\zeta^2$

$f_2'(x) = (\beta_1 + 3\zeta^2\beta_4) + 2(\beta_2 - 3\zeta\beta_4)x + 3(\beta_3 + \beta_4)x^2,$

$f_2'(\zeta) = (\beta_1 + 3\zeta^2\beta_4) + 2(\beta_2 - 3\zeta\beta_4)\zeta + 3(\beta_3 + \beta_4)\zeta^2 = \beta_1 + 2\beta_2\zeta + 3\beta_3\zeta^2$

$\therefore f_1'(\zeta) = f_2'(\zeta),\ f'(x)$ is continuous at ζ.

(e)

$f_1''(x) = 2\beta_2 + 6\beta_3x,$

$f_1''(\zeta) = 2\beta_2 + 6\beta_3\zeta$

$f_2''(x) = 2(\beta_2 - 3\zeta\beta_4) + 6(\beta_3 + \beta_4)x,$

$f_2''(\zeta) = 2(\beta_2 - 3\zeta\beta_4) + 6(\beta_3 + \beta_4)\zeta = 2\beta_2 + 6\beta_3\zeta$

$\therefore f_1''(\zeta) = f_2''(\zeta),\ f''(x)$ is continuous at ζ.

2. $\hat{g} = argmin_g \left(\sum_{i=1}^{n} \left(y_i - g(x_i) \right)^2 + \lambda \int \left[g^{(m)}(x) \right]^2 dx \right)$

(a) $\lambda = \infty, m = 0$

As penalty term of original function dominates, $\hat{g}$ should be zero.

(b) $\lambda = \infty, m = 1$

Still penalty term of first derivative dominates, so that slope of g should be zero. Hence, $\hat{g}$ is a constant function that minimizes $\sum_{i=1}^{n} \left(y_i - g(x_i) \right)^2$ (i.e, $\hat{g} = \frac{1}{n} \sum_{i=1}^{n} y_i$)

(c) $\lambda = \infty, m = 2$

Still penalty term of second derivative dominates, so that slope of $g^{(1)}$ should be zero. Hence $\hat{g}$ is a linear function that minimizes $\sum_{i=1}^{n} \left(y_i - g(x_i) \right)^2$

(d) $\lambda = \infty, m = 3$

Similarly, penalty term of third derivative of function dominates, so that slop of $g^{(2)}$ should be zero. Hence, we can estimate that $\hat{g}$ will be a quadratic function or a linear function.

(e) $\lambda = 0, m = 3$

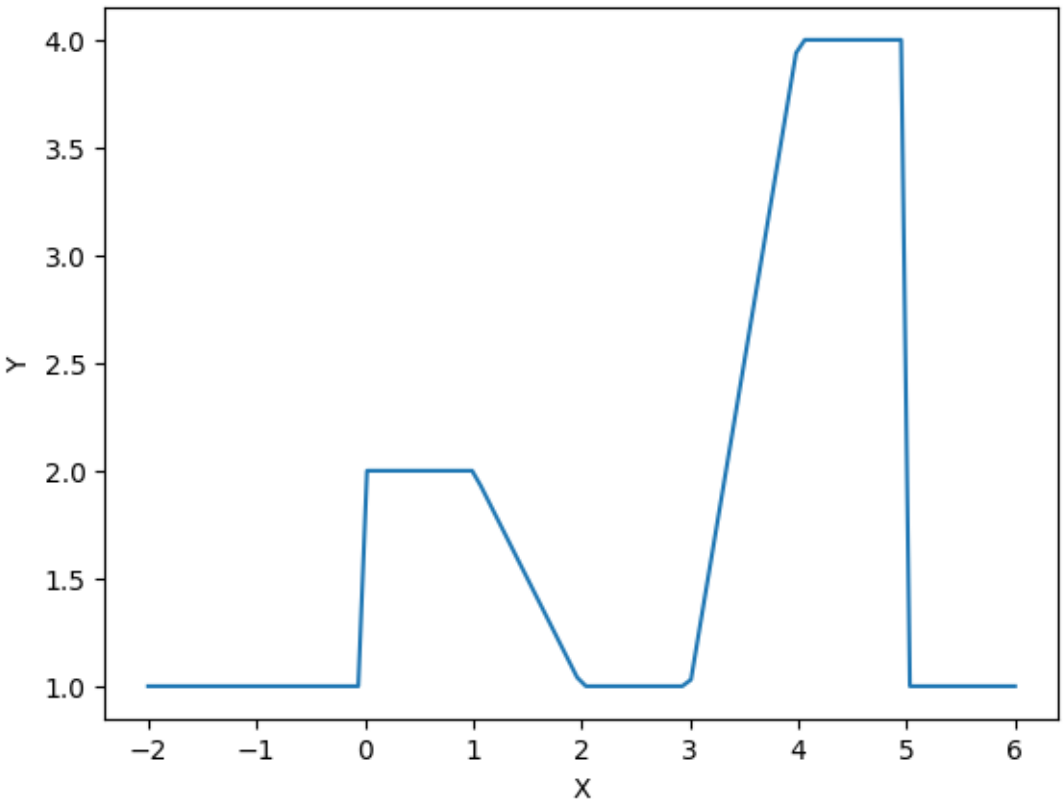
As we do not penalize third derivative of function, it is same with finding optimal function that minimizes RSS. Therefore, $\hat{g}$ will be able to interpolate all points.

4.

$$Y = \beta_0 + \beta_1 b_1(X) + \beta_2 b_2(X) + \varepsilon, \text{ (Assume } \varepsilon \sim N(0, 1^2) \text{ for simplicity)}$$

$$\text{Then, } \hat{Y} = \widehat{\beta_0} + \widehat{\beta_1} b_1(X) + \widehat{\beta_2} b_2(X) = 1 + b_1(X) + 3b_2(X) \text{ with}$$

$$\begin{cases} b_1(X) = I(0 \le X \le 2) - (X-1)I(1 \le X \le 2) \\ b_2(X) = (X-3)I(3 \le X \le 4) + I(4 < X \le 5) \end{cases}$$



10.

(a)

Before splitting given data into train data and test data set, I converted every qualitative predictor into dummy variable (In this data, 'Private' was that case.)

	Private	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	Outstate	Room.Board	Books	Personal	PhD	Terminal	S.F.Ratio	perc.alumni	Expend	Grad.Rate
Abilene Christian University	Yes	1660	1232	721	23	52	2885	537	7440	3300	450	2200	70	78	18.1	12	7041	60
Adelphi University	Yes	2186	1924	512	16	29	2683	1227	12280	6450	750	1500	29	30	12.2	16	10527	56
Adrian College	Yes	1428	1097	336	22	50	1036	99	11250	3750	400	1165	53	66	12.9	30	8735	54
Agnes Scott College	Yes	417	349	137	60	89	510	63	12960	5450	450	875	92	97	7.7	37	19016	59
Alaska Pacific University	Yes	193	146	55	16	44	249	869	7560	4120	800	1500	76	72	11.9	2	10922	15

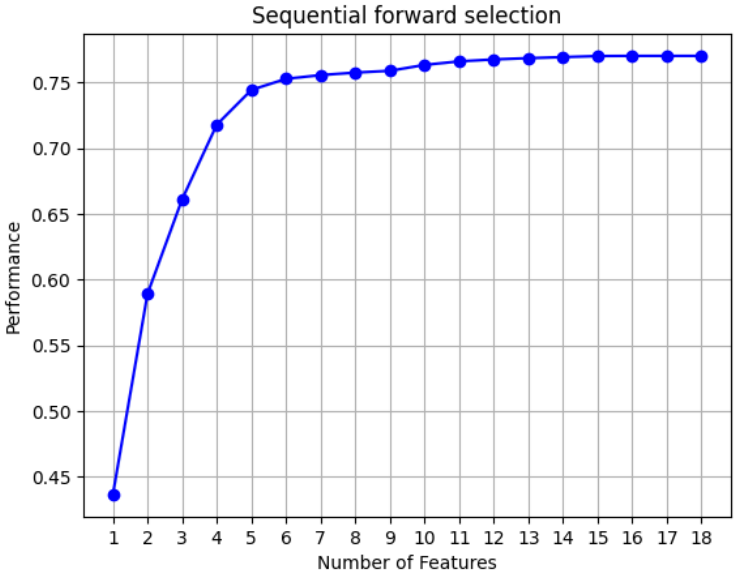
```
[ ] df = pd.get_dummies(df)
```

```
[ ] df.head()
```

	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	Outstate	Room.Board	Books	Personal	PhD	Terminal	S.F.Ratio	perc.alumni	Expend	Grad.Rate	Private_No	Private_Yes
Abilene Christian University	1660	1232	721	23	52	2885	537	7440	3300	450	2200	70	78	18.1	12	7041	60	0	1
Adelphi University	2186	1924	512	16	29	2683	1227	12280	6450	750	1500	29	30	12.2	16	10527	56	0	1
Adrian College	1428	1097	336	22	50	1036	99	11250	3750	400	1165	53	66	12.9	30	8735	54	0	1
Agnes Scott College	417	349	137	60	89	510	63	12960	5450	450	875	92	97	7.7	37	19016	59	0	1
Alaska Pacific University	193	146	55	16	44	249	869	7560	4120	800	1500	76	72	11.9	2	10922	15	0	1

Then, I split given data set into train, and test data set.

```
[ ]  X = df.loc[:, df.columns != 'Outstate']
     Y = df.loc[:, 'Outstate']

     X_train, X_test, Y_train, Y_test = train_test_split(
         X, Y, test_size = .3, random_state=1
     )
```



The picture above describes the R-square value as number of feature increases using forward selection. It seems that taking subset consists up to 6th features (['Room.Board', 'perc.alumni', 'Expend', 'Private_No', 'PhD', 'Grad.Rate']) work well. However, lets look further using some criteria (Adjusted R-square, Mallow's Cp, AIC) below.

According to 'Adjusted R-square' criteria, ['Room.Board', 'perc.alumni', 'Expend', 'Private_No', 'PhD', 'Grad.Rate', 'Personal', 'Top25perc', 'Accept', 'Apps', 'Enroll', 'Top10perc', 'S.F.Ratio', 'Terminal', 'Books'] was the best subset of predictors for estimating out-of-state tuition.

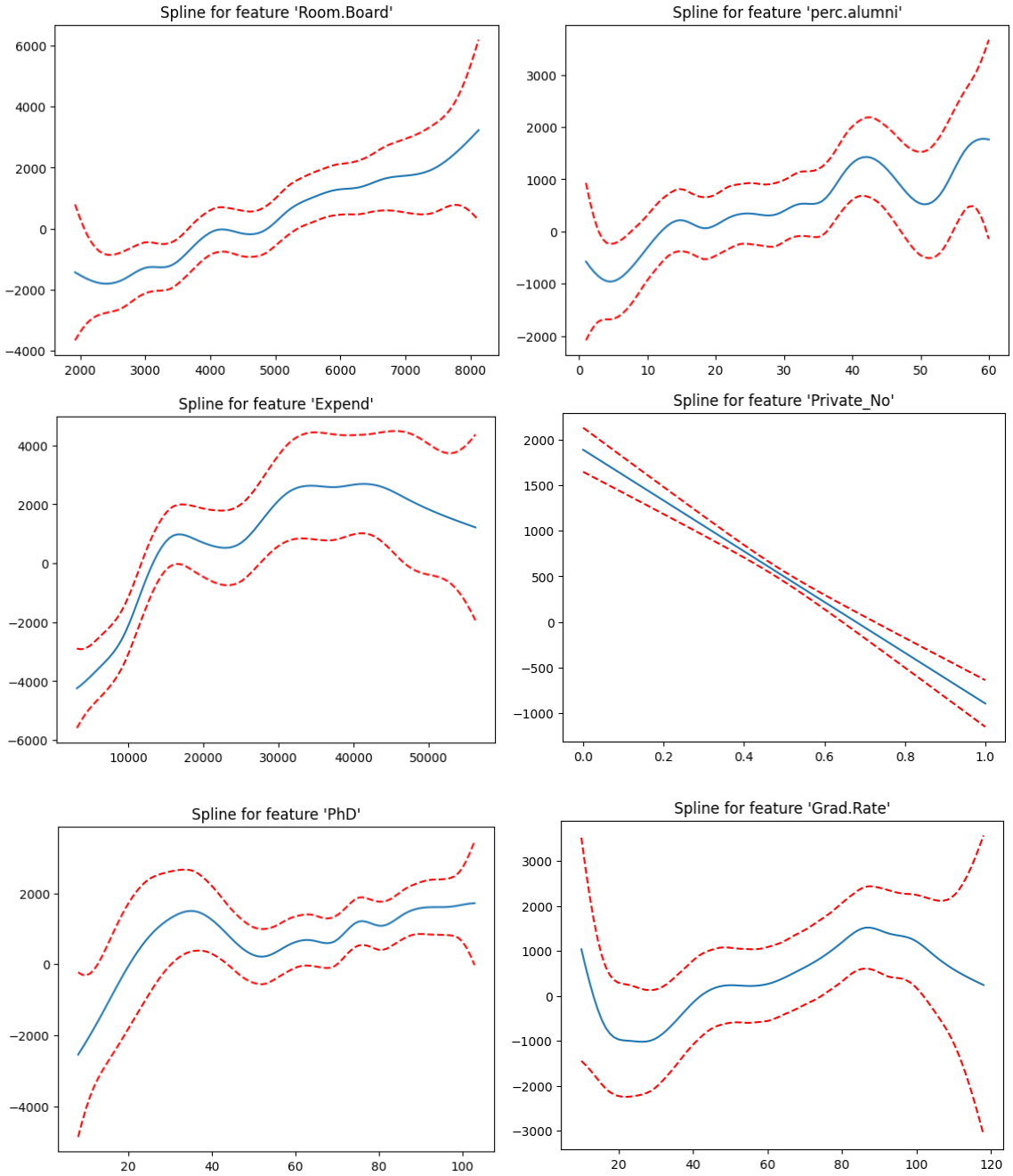
According to 'Mallow's Cp' criteria, ['Room.Board'] was the best subset of predictors for estimating out-of-state tuition.

According to 'AIC' criteria, ['Room.Board', 'perc.alumni', 'Expend', 'Private_No', 'PhD', 'Grad.Rate', 'Personal', 'Top25perc', 'Accept', 'Apps', 'Enroll', 'Top10perc', 'S.F.Ratio'] was the best subset of predictors for estimating out-of-state tuition.

One can easily see that different model was selected based on each criterion.

For the sake of simplicity, I will use top 6 features as a training data set (['Room.Board', 'perc.alumni', 'Expend', 'Private_No', 'PhD', 'Grad.Rate']), according in the further problem.

(b)


Spline for feature 'Room.Board'
Spline for feature 'perc.alumni'
Spline for feature 'Expend'
Spline for feature 'Private_No'
Spline for feature 'PhD'
Spline for feature 'Grad.Rate'

(c)

```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

Y_pred = gam.predict(X_test[selected_features])

# Evaluate the model using Mean Squared Error
mse = mean_squared_error(Y_test, Y_pred)
print(f"Mean Squared Error on Test Set: {mse}")

# Evaluate the model using R^2 score
r2 = r2_score(Y_test, Y_pred)
print(f"R^2 Score on Test Set: {r2:.2f}")
```

```
Mean Squared Error on Test Set: 3481100.408144848
R^2 Score on Test Set: 0.79
```

Test MSE was about 3481100, and R-square value was 0.79.

(d)

This picture summarizes the statistics of GAM when every predictor is fitted as spline (non-parametrically).

```
[25] from pygam import LinearGAM, s

     selected_features = ['Room.Board', 'perc.alumni', 'Expend', 'Private_No', 'PhD', 'Grad.Rate']
     gam = LinearGAM(s(0)+s(1)+s(2)+s(3)+s(4)+s(5))

     gam.fit(X_train[selected_features], Y_train)
     print(gam.summary())

     # Evaluate the model using Mean Squared Error
     Y_pred = gam.predict(X_test[selected_features])
     mse = mean_squared_error(Y_test, Y_pred)
     print(f"Mean Squared Error on Test Set: {mse}")

     # Evaluate the model using R^2 score
     r2 = r2_score(Y_test, Y_pred)
     print(f"R^2 Score on Test Set: {r2:.2f}")
```

```
LinearGAM
============================================= ==========================================================
Distribution:                       NormalDist Effective DoF:                                    53.5653
Link Function:                     IdentityLink Log Likelihood:                                -8679.2047
Number of Samples:                          543 AIC:                                           17467.5399
                                                AICc:                                          17479.9803
                                                GCV:                                         4232277.0771
                                                Scale:                                       3488086.0916
                                                Pseudo R-Squared:                                  0.8034
============================================================================================================
Feature Function                  Lambda               Rank         EDoF         P > x        Sig. Code
================================= ==================== ============ ============ ============ ============
s(0)                              [0.6]                20           12.7         1.51e-08     ***
s(1)                              [0.6]                20           11.7         1.65e-02     *
s(2)                              [0.6]                20           8.6          1.11e-16     ***
s(3)                              [0.6]                20           1.7          1.11e-16     ***
s(4)                              [0.6]                20           10.6         3.49e-02     *
s(5)                              [0.6]                20           8.3          8.02e-04     ***
intercept                                              1            0.0          1.11e-16     ***
============================================================================================================
Significance codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

WARNING: Fitting splines and a linear function to a feature introduces a model identifiability problem
         which can cause p-values to appear significant when they are not.

WARNING: p-values calculated in this manner behave correctly for un-penalized models or models with
         known smoothing parameters, but when smoothing parameters have been estimated, the p-values
         are typically lower than they should be, meaning that the tests reject the null too readily.
None
Mean Squared Error on Test Set: 3481100.408144848
R^2 Score on Test Set: 0.79
<ipython-input-25-ae668f21cfea>:7: UserWarning: KNOWN BUG: p-values computed in this summary are likely much
```

And the six figures below show the results when we substitute each predictor linearly.

1) Linear term on Room.Board

```
[16] from pygam import LinearGAM, l, s

     # Define the model using smooth terms for all predictors
     terms = l(0) + s(1) + s(2) + s(3) + s(4) + s(5)
     gam = LinearGAM(terms).fit(X_train[selected_features], Y_train)

     # Print model summary
     print(gam.summary())

     # Evaluate the model using Mean Squared Error
     Y_pred = gam.predict(X_test[selected_features])
     mse = mean_squared_error(Y_test, Y_pred)
     print(f"Mean Squared Error on Test Set: {mse}")

     # Evaluate the model using R^2 score
     r2 = r2_score(Y_test, Y_pred)
     print(f"R^2 Score on Test Set: {r2:.2f}")
```

```
LinearGAM
============================================= ==========================================================
Distribution:                      NormalDist Effective DoF:                                     43.8537
Link Function:                  IdentityLink Log Likelihood:                                  -8677.0938
Number of Samples:                       543 AIC:                                             17443.895
                                             AICc:                                            17452.1691
                                             GCV:                                           4060177.066
                                             Scale:                                         3474552.3198
                                             Pseudo R-Squared:                                    0.8003
=================================================================================================
Feature Function                  Lambda               Rank         EDoF         P > x        Sig. Code
================================= ==================== ============ ============ ============ ============
l(0)                              [0.6]                1            1.0          4.44e-16     ***
s(1)                              [0.6]                20           12.8         9.17e-03     **
s(2)                              [0.6]                20           8.9          1.11e-16     ***
s(3)                              [0.6]                20           1.6          1.11e-16     ***
s(4)                              [0.6]                20           11.0         3.89e-02     *
s(5)                              [0.6]                20           8.5          6.04e-04     ***
intercept                                              1            0.0          1.11e-16     ***
=================================================================================================
Significance codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

WARNING: Fitting splines and a linear function to a feature introduces a model identifiability problem
         which can cause p-values to appear significant when they are not.

WARNING: p-values calculated in this manner behave correctly for un-penalized models or models with
         known smoothing parameters, but when smoothing parameters have been estimated, the p-values
         are typically lower than they should be, meaning that the tests reject the null too readily.
None
Mean Squared Error on Test Set: 3556019.7289496
R^2 Score on Test Set: 0.79
```

2) Linear term on perc.alumni

```
[17] from pygam import LinearGAM, l, s

     # Define the model using smooth terms for all predictors
     terms = s(0) + l(1) + s(2) + s(3) + s(4) + s(5)
     gam = LinearGAM(terms).fit(X_train[selected_features], Y_train)

     # Print model summary
     print(gam.summary())

     # Evaluate the model using Mean Squared Error
     Y_pred = gam.predict(X_test[selected_features])
     mse = mean_squared_error(Y_test, Y_pred)
     print(f"Mean Squared Error on Test Set: {mse}")

     # Evaluate the model using R^2 score
     r2 = r2_score(Y_test, Y_pred)
     print(f"R^2 Score on Test Set: {r2:.2f}")
```

```
LinearGAM
============================================= ==========================================================
Distribution:                      NormalDist Effective DoF:                                     43.3888
Link Function:                  IdentityLink Log Likelihood:                                  -8679.0704
Number of Samples:                       543 AIC:                                             17446.9184
                                             AICc:                                            17455.0161
                                             GCV:                                           4067776.4995
                                             Scale:                                         3487223.2668
                                             Pseudo R-Squared:                                    0.7994
=================================================================================================
Feature Function                  Lambda               Rank         EDoF         P > x        Sig. Code
================================= ==================== ============ ============ ============ ============
s(0)                              [0.6]                20           12.6         8.45e-09     ***
l(1)                              [0.6]                1            1.0          5.78e-06     ***
s(2)                              [0.6]                20           8.8          1.11e-16     ***
s(3)                              [0.6]                20           1.7          1.11e-16     ***
s(4)                              [0.6]                20           10.9         3.07e-02     *
s(5)                              [0.6]                20           8.4          6.80e-04     ***
intercept                                              1            0.0          1.11e-16     ***
=================================================================================================
Significance codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

WARNING: Fitting splines and a linear function to a feature introduces a model identifiability problem
         which can cause p-values to appear significant when they are not.

WARNING: p-values calculated in this manner behave correctly for un-penalized models or models with
         known smoothing parameters, but when smoothing parameters have been estimated, the p-values
         are typically lower than they should be, meaning that the tests reject the null too readily.
None
Mean Squared Error on Test Set: 3478279.911273383
R^2 Score on Test Set: 0.79
```

3) Linear term on Expend

```
[18] from pygam import LinearGAM, l, s

     # Define the model using smooth terms for all predictors
     terms = s(0) + s(1) + l(2) + s(3) + s(4) + s(5)
     gam = LinearGAM(terms).fit(X_train[selected_features], Y_train)

     # Print model summary
     print(gam.summary())

     # Evaluate the model using Mean Squared Error
     Y_pred = gam.predict(X_test[selected_features])
     mse = mean_squared_error(Y_test, Y_pred)
     print(f"Mean Squared Error on Test Set: {mse}")

     # Evaluate the model using R^2 score
     r2 = r2_score(Y_test, Y_pred)
     print(f"R^2 Score on Test Set: {r2:.2f}")
```

```
LinearGAM
============================================= ==========================================================
Distribution:                        NormalDist Effective DoF:                                     46.2667
Link Function:                     IdentityLink Log Likelihood:                                -8742.6334
Number of Samples:                         543 AIC:                                            17579.8001
                                               AICc:                                           17589.0229
                                               GCV:                                          4623527.2501
                                               Scale:                                        3920286.4574
                                               Pseudo R-Squared:                                   0.7757
==========================================================================================================
Feature Function                  Lambda               Rank         EDoF         P > x        Sig. Code
================================= ==================== ============ ============ ========== ============
s(0)                              [0.6]                20           12.5         2.23e-13     ***
s(1)                              [0.6]                20           11.6         1.19e-03     **
l(2)                              [0.6]                1            0.9          6.66e-16     ***
s(3)                              [0.6]                20           1.7          1.11e-16     ***
s(4)                              [0.6]                20           10.9         1.43e-05     ***
s(5)                              [0.6]                20           8.6          2.84e-03     **
intercept                                              1            0.0          1.11e-16     ***
==========================================================================================================
Significance codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

WARNING: Fitting splines and a linear function to a feature introduces a model identifiability problem
         which can cause p-values to appear significant when they are not.

WARNING: p-values calculated in this manner behave correctly for un-penalized models or models with
         known smoothing parameters, but when smoothing parameters have been estimated, the p-values
         are typically lower than they should be, meaning that the tests reject the null too readily.
None
Mean Squared Error on Test Set: 4055237.929871498
R^2 Score on Test Set: 0.75
<ipython-input-18-03e2199e1632>:8: UserWarning: KNOWN BUG: p-values computed in this summary are likely much
```

4) Linear term on Private_No

```
[19] from pygam import LinearGAM, l, s

     # Define the model using smooth terms for all predictors
     terms = s(0) + s(1) + s(2) + l(3) + s(4) + s(5)
     gam = LinearGAM(terms).fit(X_train[selected_features], Y_train)

     # Print model summary
     print(gam.summary())

     # Evaluate the model using Mean Squared Error
     Y_pred = gam.predict(X_test[selected_features])
     mse = mean_squared_error(Y_test, Y_pred)
     print(f"Mean Squared Error on Test Set: {mse}")

     # Evaluate the model using R^2 score
     r2 = r2_score(Y_test, Y_pred)
     print(f"R^2 Score on Test Set: {r2:.2f}")
```

```
LinearGAM
============================================= ==========================================================
Distribution:                        NormalDist Effective DoF:                                     53.5549
Link Function:                     IdentityLink Log Likelihood:                                -8679.2203
Number of Samples:                         543 AIC:                                            17467.5504
                                               AICc:                                           17479.9858
                                               GCV:                                          4232225.5771
                                               Scale:                                        3488186.1667
                                               Pseudo R-Squared:                                   0.8034
==========================================================================================================
Feature Function                  Lambda               Rank         EDoF         P > x        Sig. Code
================================= ==================== ============ ============ ========== ============
s(0)                              [0.6]                20           12.7         1.35e-08     ***
s(1)                              [0.6]                20           11.7         1.62e-02     *
s(2)                              [0.6]                20           8.6          1.11e-16     ***
l(3)                              [0.6]                1            0.9          1.11e-16     ***
s(4)                              [0.6]                20           10.9         4.37e-02     *
s(5)                              [0.6]                20           8.6          7.73e-04     ***
intercept                                              1            0.0          1.11e-16     ***
==========================================================================================================
Significance codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

WARNING: Fitting splines and a linear function to a feature introduces a model identifiability problem
         which can cause p-values to appear significant when they are not.

WARNING: p-values calculated in this manner behave correctly for un-penalized models or models with
         known smoothing parameters, but when smoothing parameters have been estimated, the p-values
         are typically lower than they should be, meaning that the tests reject the null too readily.
None
Mean Squared Error on Test Set: 3475406.9847101797
R^2 Score on Test Set: 0.79
<ipython-input-19-a056ca36a8b1>:8: UserWarning: KNOWN BUG: p-values computed in this summary are likely much
```

5) Linear term on PhD

```
[20] from pygam import LinearGAM, l, s

     # Define the model using smooth terms for all predictors
     terms = s(0) + s(1) + s(2) + s(3) + l(4) + s(5)
     gam = LinearGAM(terms).fit(X_train[selected_features], Y_train)

     # Print model summary
     print(gam.summary())

     # Evaluate the model using Mean Squared Error
     Y_pred = gam.predict(X_test[selected_features])
     mse = mean_squared_error(Y_test, Y_pred)
     print(f"Mean Squared Error on Test Set: {mse}")

     # Evaluate the model using R^2 score
     r2 = r2_score(Y_test, Y_pred)
     print(f"R^2 Score on Test Set: {r2:.2f}")
```

```
LinearGAM
============================================= ==========================================================
Distribution:                      NormalDist Effective DoF:                                     44.2196
Link Function:                 IdentityLink Log Likelihood:                                  -8684.7306
Number of Samples:                      543 AIC:                                             17459.9003
                                            AICc:                                            17468.3146
                                            GCV:                                           4123430.4265
                                            Scale:                                         3523764.0467
                                            Pseudo R-Squared:                                    0.7976
==========================================================================================================
Feature Function                  Lambda               Rank         EDoF         P > x        Sig. Code
================================= ==================== ============ ============ ============ ============
s(0)                              [0.6]                20           12.6         1.69e-08     ***
s(1)                              [0.6]                20           11.6         1.82e-02     *
s(2)                              [0.6]                20           8.4          1.11e-16     ***
s(3)                              [0.6]                20           1.3          1.11e-16     ***
l(4)                              [0.6]                1            0.9          6.45e-04     ***
s(5)                              [0.6]                20           9.3          1.38e-03     **
intercept                                              1            0.0          1.11e-16     ***
==========================================================================================================
Significance codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

WARNING: Fitting splines and a linear function to a feature introduces a model identifiability problem
         which can cause p-values to appear significant when they are not.

WARNING: p-values calculated in this manner behave correctly for un-penalized models or models with
         known smoothing parameters, but when smoothing parameters have been estimated, the p-values
         are typically lower than they should be, meaning that the tests reject the null too readily.
None
Mean Squared Error on Test Set: 3458390.841745832
R^2 Score on Test Set: 0.79
<ipython-input-20-0bc56a9b1fe3>:8: UserWarning: KNOWN BUG: p-values computed in this summary are likely much
```

6) Linear term on Grad_Rate

```
from pygam import LinearGAM, l, s

# Define the model using smooth terms for all predictors
terms = s(0) + s(1) + s(2) + s(3) + s(4) + l(5)
gam = LinearGAM(terms).fit(X_train[selected_features], Y_train)

# Print model summary
print(gam.summary())

# Evaluate the model using Mean Squared Error
Y_pred = gam.predict(X_test[selected_features])
mse = mean_squared_error(Y_test, Y_pred)
print(f"Mean Squared Error on Test Set: {mse}")

# Evaluate the model using R^2 score
r2 = r2_score(Y_test, Y_pred)
print(f"R^2 Score on Test Set: {r2:.2f}")
```

```
LinearGAM
============================================= ==========================================================
Distribution:                      NormalDist Effective DoF:                                     44.4031
Link Function:                 IdentityLink Log Likelihood:                                  -8677.8968
Number of Samples:                      543 AIC:                                             17446.5997
                                            AICc:                                            17455.0849
                                            GCV:                                           4074714.4303
                                            Scale:                                         3479694.2682
                                            Pseudo R-Squared:                                    0.8002
==========================================================================================================
Feature Function                  Lambda               Rank         EDoF         P > x        Sig. Code
================================= ==================== ============ ============ ============ ============
s(0)                              [0.6]                20           12.6         6.82e-09     ***
s(1)                              [0.6]                20           11.6         6.90e-03     **
s(2)                              [0.6]                20           8.3          1.11e-16     ***
s(3)                              [0.6]                20           1.4          1.11e-16     ***
s(4)                              [0.6]                20           9.7          1.98e-02     *
l(5)                              [0.6]                1            0.9          7.11e-07     ***
intercept                                              1            0.0          1.11e-16     ***
==========================================================================================================
Significance codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

WARNING: Fitting splines and a linear function to a feature introduces a model identifiability problem
         which can cause p-values to appear significant when they are not.

WARNING: p-values calculated in this manner behave correctly for un-penalized models or models with
         known smoothing parameters, but when smoothing parameters have been estimated, the p-values
         are typically lower than they should be, meaning that the tests reject the null too readily.
None
Mean Squared Error on Test Set: 3531713.8310200856
R^2 Score on Test Set: 0.79
<ipython-input-27-553f1fd3816O>:8: UserWarning: KNOWN BUG: p-values computed in this summary are likely much
```

Model	GCV	Test MSE	AIC
s(0)+ s(1)+ s(2)+ s(3)+ s(4)+ s(5)	4232277	3481100	17467
l(0)+ s(1)+ s(2)+ s(3)+ s(4)+ s(5)	4060177	3556019	17443
s(0)+ l(1)+ s(2)+ s(3)+ s(4)+ s(5)	4067776	3478279	17446
s(0)+ s(1)+ **l(2)**+ s(3)+ s(4)+ s(5)	4623527	4055237	17579
s(0)+ s(1)+ s(2)+ l(3)+ s(4)+ s(5)	4232225	3475406	17467
s(0)+ s(1)+ s(2)+ s(3)+ l(4)+ s(5)	4123430	3458390	17459
s(0)+ s(1)+ s(2)+ s(3)+ s(4)+ l(5)	4074714	3531713	17446

From the statistics, one can conclude that 3rd feature 'Expend' has the most significant non-linear relationship with the response, because the model performance worsens the most when we tried to substitute its non-parametric approximation into parametric approximation (linear regression in this case).

11.

(a)

```
[36] x1 = np.random.normal(size=100)
     x2 = np.random.normal(size=100)
     eps = np.random.normal(size=100)
     y = 16 + 5.1 * x1 - 7.3 * x2 + eps
```

$$X_1 \sim N(0,1),\ X_2 \sim N(0,1),\ \varepsilon \sim N(0,1),\ Y = 16 + 5.1X_1 - 7.3X_2 + \varepsilon$$

(b)

```
[37] beta0 = np.nan
     beta1 = 1
     beta2 = np.nan

     print(beta0, beta1, beta2)

     nan 1 nan
```

(c)

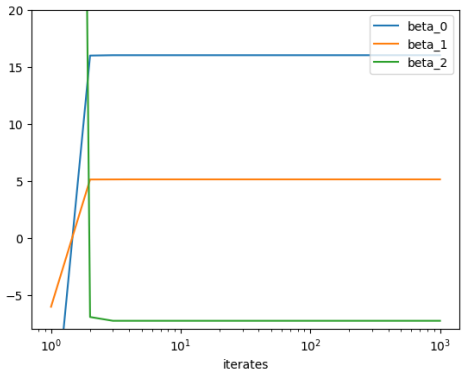
```
[41] lr = LinearRegression()
     lr.fit(x2.reshape(-1 ,1), (y-beta1*x1))

     print(lr.coef_)
     print(lr.intercept_)
     beta2 = lr.coef_[0]

     [-7.15994424]
     15.99776872061943
```

(d)

```
[42] lr = LinearRegression()
     lr.fit(x1.reshape(-1, 1), y-beta2*x2)

     print(lr.coef_)
     print(lr.intercept_)
     beta1 = lr.coef_[0]

     [5.11046383]
     15.987520004769923
```

(e)

```
[44] from tqdm import tqdm

     beta0_list = []
     beta1_list = []
     beta2_list = []

     lr = LinearRegression()
     for i in tqdm(range(1000)):
       lr.fit(x2.reshape(-1, 1), y-beta1*x1)
       beta2 = lr.coef_[0]

       lr.fit(x1.reshape(-1, 1), y-beta2*x2)
       beta1 = lr.coef_[0]

       beta0 = lr.intercept_
       beta0_list.append(beta0)
       beta1_list.append(beta1)
       beta2_list.append(beta2)

     print(beta0, beta1, beta2)

100%|██████████| 1000/1000 [00:05<00:00, 173.83it/s]15.999633643611695 5.114287788429456 -7.282065348875617
```



After 1000 iterates, we obtained $\widehat{\beta_0} = 15.9996, \widehat{\beta_1} = 5.1143,\ \widehat{\beta_2} = -7.2820$

(f)

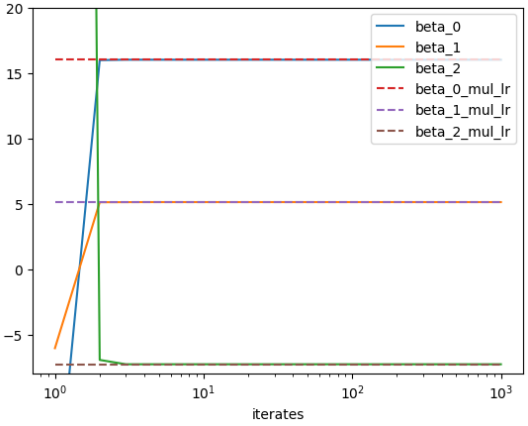
```
[85] reg = smf.ols('y ~ x1 + x2', data=df).fit()
     reg.summary()
```

OLS Regression Results

Dep. Variable:	y	R-squared:	0.988
Model:	OLS	Adj. R-squared:	0.987
Method:	Least Squares	F-statistic:	3905.
Date:	Sun, 29 Oct 2023	Prob (F-statistic):	2.01e-93
Time:	04:42:05	Log-Likelihood:	-134.67
No. Observations:	100	AIC:	275.3
Df Residuals:	97	BIC:	283.2
Df Model:	2		
Covariance Type:	nonrobust		

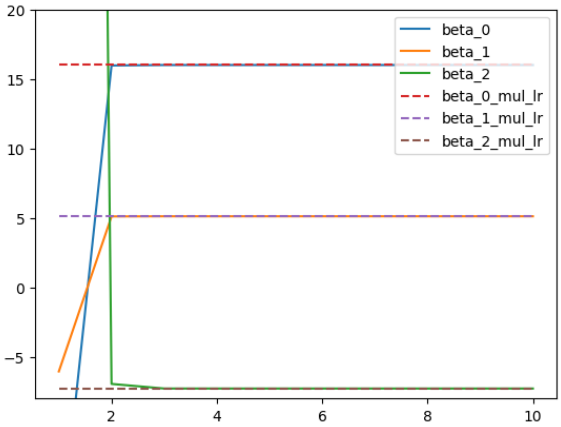
	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	15.9996	0.095	168.485	0.000	15.811	16.188
x1	5.1143	0.101	50.584	0.000	4.914	5.315
x2	-7.2821	0.098	-73.978	0.000	-7.477	-7.087

Omnibus:	3.023	Durbin-Watson:	1.906
Prob(Omnibus):	0.221	Jarque-Bera (JB):	2.798
Skew:	0.148	Prob(JB):	0.247
Kurtosis:	3.765	Cond. No.	1.13



From multiple linear regression, we obtained, $\widehat{\beta_0} = 15.9996, \widehat{\beta_1} = 5.1143,\ \widehat{\beta_2} = -7.2820$

(g)



From the graph visualizing performance of backfitting algorithm up to 10^{th} iterates, it seems that 3~4 iterations are enough to get a good coefficient estimate.