Question 1

Name: Zicheng Qu zID: z5092597

(a) $\text{cov}(X,Y) = \dfrac{\sum_i (X_i - \bar{X})(Y_i - \bar{Y})}{n-1}$

$\text{var}(X) = E\left((X - E(X))^2\right) = \dfrac{\sum_i (X_i - \bar{X})^2}{n-1}$

Hence, $\hat{\beta}_1 = \dfrac{\text{cov}(X,Y)}{\text{var}(X)} = \dfrac{\sum_i (X_i - \bar{X})(Y_i - \bar{Y})}{\sum_i (X_i - \bar{X})^2}$

$\hat{\beta}_0 = \bar{Y} - \hat{\beta}_1 \bar{X}$

Similarly, $\tilde{\beta}_1 = \dfrac{\text{cov}(\tilde{X},Y)}{\text{var}(\tilde{X})} = \dfrac{\sum_i (\tilde{X}_i - \bar{\tilde{X}})(Y_i - \bar{Y})}{\sum_i (X_i - \bar{\tilde{X}})^2}$

$\tilde{\beta}_0 = \bar{Y} - \tilde{\beta}_1 \bar{\tilde{X}}$

From $\tilde{X}_i = c(X_i + d)$, I have $\bar{\tilde{X}} = c(\bar{X} + d)$

Hence, $\tilde{X}_i - \bar{\tilde{X}} = c(X_i + d) - c(\bar{X} + d) = c(X_i - \bar{X})$

$$\tilde{\beta}_1 = \frac{\sum_i (\tilde{X}_i - \bar{\tilde{X}})(Y_i - \bar{Y})}{\sum_i (X_i - \bar{\tilde{X}})^2} = \frac{\sum_i c\,(X_i - \bar{X})(Y_i - \bar{Y})}{\sum_i c^2 (X_i - \bar{X})^2} = \frac{1}{c} \cdot \frac{\sum_i (X_i - \bar{X})(Y_i - \bar{Y})}{\sum_i (X_i - \bar{X})^2} = \frac{1}{c}\hat{\beta}_1$$

Since $\begin{cases} \hat{\beta}_0 = \bar{Y} - \hat{\beta}_1 \bar{X} \\ \tilde{\beta}_0 = \bar{Y} - \tilde{\beta}_1 \bar{\tilde{X}} \end{cases}$, I have $\begin{cases} \hat{\beta}_0 = \bar{Y} - \hat{\beta}_1 \bar{X} \\ \tilde{\beta}_0 = \bar{Y} - \frac{1}{c}\hat{\beta}_1 \cdot c(\bar{X} + d) = \bar{Y} - \hat{\beta}_1 \bar{X} - \hat{\beta}_1 d \end{cases}$

Hence $\tilde{\beta}_0 - \hat{\beta}_0 = \bar{Y} - \hat{\beta}_1 \bar{X} - \hat{\beta}_1 d - (\bar{Y} - \hat{\beta}_1 \bar{X}) = -\hat{\beta}_1 d$

Hence $\tilde{\beta}_0 = \hat{\beta}_0 - \hat{\beta}_1 d$

Since $\begin{cases} \hat{\sigma} = \sqrt{\dfrac{\hat{e}^T \hat{e}}{n-p}} \\ \hat{e}_i = Y_i - \hat{Y}_i \end{cases}$, I have $\hat{\sigma} = \sqrt{\dfrac{\sum_i (Y_i - \hat{Y}_i)^2}{n-p}}$, similarly $\tilde{\sigma} = \sqrt{\dfrac{\sum_i (Y_i - \tilde{Y}_i)^2}{n-p}}$

$Y_i - \hat{Y}_i = Y_i - (\hat{\beta}_0 + \hat{\beta}_1 X_i)$

$Y_i - \tilde{Y}_i = Y_i - (\tilde{\beta}_0 + \tilde{\beta}_1 \tilde{X}_i)$

$= Y_i - \left(\hat{\beta}_0 - \hat{\beta}_1 d + \frac{1}{c}\hat{\beta}_1 \cdot c(X_i + d)\right)$

$= Y_i - (\hat{\beta}_0 - \hat{\beta}_1 d + \hat{\beta}_1 X_i + \hat{\beta}_1 d)$

$= Y_i - (\hat{\beta}_0 + \hat{\beta}_1 X_i)$

Hence $Y_i - \tilde{Y}_i = Y_i - \hat{Y}_i$, $\tilde{\sigma} = \hat{\sigma}$

In summary $\begin{cases} \tilde{\beta}_1 = \frac{1}{c}\hat{\beta}_1 \\ \tilde{\beta}_0 = \hat{\beta}_0 - \hat{\beta}_1 d \\ \tilde{\sigma} = \hat{\sigma} \end{cases}$

(b) From question 1. (a). I know that $\begin{cases} \hat{\beta}_1 = \frac{\sum_i (x_i - \bar{x})(Y_i - \bar{Y})}{\sum_i (x_i - \bar{x})^2} \\ \hat{\beta}_0 = \bar{Y} - \hat{\beta}_1 \bar{x} \end{cases}$

Assuming the number of $x_i = 1$ is a, the number of $x_i = 0$ is b.

Hence $\bar{x} = \frac{a \cdot 1 + b \cdot 0}{a+b} = \frac{a}{a+b}$

$\bar{Y} = \frac{a \cdot \bar{Y}_t + b \cdot \bar{Y}_p}{a+b}$

Hence $\sum_i (x_i - \bar{x})(Y_i - \bar{Y})$

$= \sum_a (1 - \bar{x})(Y_a - \bar{Y}) + \sum_b (0 - \bar{x})(Y_b - \bar{Y})$

$= (1 - \bar{x}) \sum_a (Y_a - \bar{Y}) + (0 - \bar{x}) \sum_b (Y_b - \bar{Y})$

$= (1 - \bar{x}) (a \cdot \bar{Y}_t - a\bar{Y}) - \bar{x} \cdot (b\bar{Y}_p - b\bar{Y})$

$= (a+b)\bar{x}\bar{Y} - (a\bar{Y}_t + b\bar{Y}_p)\bar{x} - a\bar{Y} + a\bar{Y}_t$

$= (a+b) \cdot \frac{a}{a+b} \cdot \frac{a\bar{Y}_t + b\bar{Y}_p}{a+b} - (a\bar{Y}_t + b\bar{Y}_p) \cdot \frac{a}{a+b} + a\left(\bar{Y}_t - \frac{a \cdot \bar{Y}_t + b \cdot \bar{Y}_p}{a+b}\right)$

$= \frac{a}{a+b} \cdot (a\bar{Y}_t + b\bar{Y}_p) - (a\bar{Y}_t + b\bar{Y}_p) \cdot \frac{a}{a+b} + a\left(\bar{Y}_t - \frac{a \cdot \bar{Y}_t + b \cdot \bar{Y}_p}{a+b}\right)$

$= 0 + a\left(\bar{Y}_t - \frac{a \cdot \bar{Y}_t + b \cdot \bar{Y}_p}{a+b}\right)$

$= a \cdot \left(\frac{a\bar{Y}_t + b\bar{Y}_t}{a+b} - \frac{a\bar{Y}_t + b \cdot \bar{Y}_p}{a+b}\right)$

$= \frac{ab}{a+b}(\bar{Y}_t - \bar{Y}_p)$

$\sum_i (x_i - \bar{x})^2$

$= \sum_a (1 - \bar{x})^2 + \sum_b (0 - \bar{x})^2$

$= a(1 - \bar{x})^2 + b\bar{x}^2$

$= (a+b)\bar{x}^2 - 2a\bar{x} + a$

$= (a+b)\frac{a^2}{(a+b)^2} - 2a \cdot \frac{a}{a+b} + a$

$= \frac{a^2}{a+b} - \frac{2a^2}{a+b} + \frac{a(a+b)}{a+b}$

$= \frac{a^2+ab}{a+b} - \frac{a^2}{a+b}$

$= \frac{ab}{a+b}$

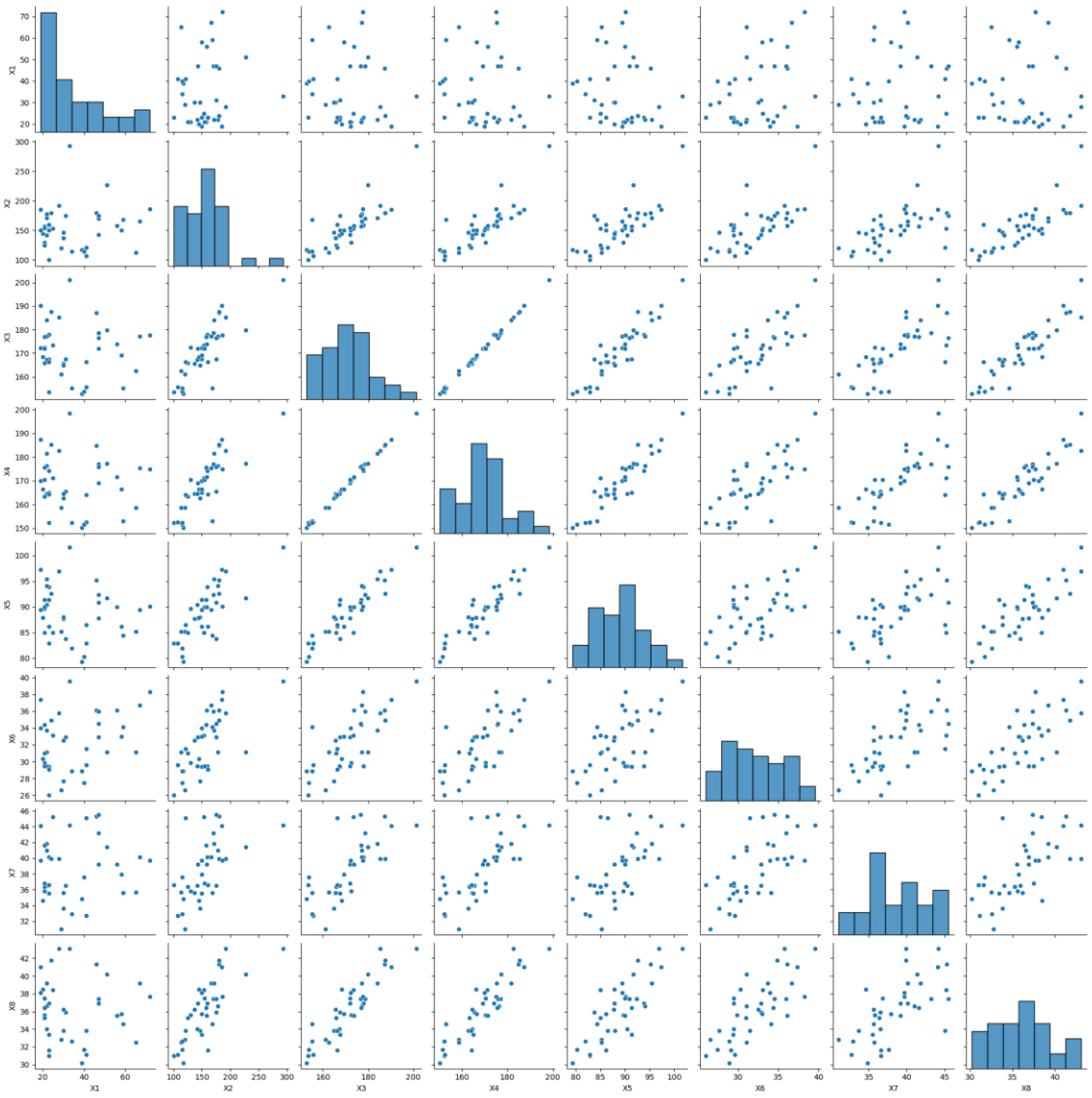
Hence, $\hat{\beta}_1 = \frac{\sum_i (x_i - \bar{x})(Y_i - \bar{Y})}{\sum_i (x_i - \bar{x})^2} = \frac{\frac{ab}{a+b}(\bar{Y}_t - \bar{Y}_p)}{\frac{ab}{a+b}} = \bar{Y}_t - \bar{Y}_p$

$$\begin{aligned}
\hat{\beta}_0 &= \bar{Y} - \hat{\beta}_1 \bar{x} \\
&= \frac{a \cdot \bar{Y}_t + b \cdot \bar{Y}_p}{a+b} - (\bar{Y}_t - \bar{Y}_p) \cdot \frac{a}{a+b} \\
&= \frac{1}{a+b}(a\bar{Y}_t + b\bar{Y}_p - a\bar{Y}_t + a\bar{Y}_p) \\
&= \frac{1}{a+b}(b\bar{Y}_p + a\bar{Y}_p) \\
&= \bar{Y}_p
\end{aligned}$$

In summary $\begin{cases} \hat{\beta}_0 = \bar{Y}_p \\ \hat{\beta}_1 = \bar{Y}_t - \bar{Y}_p \end{cases}$

Question 2

(a)

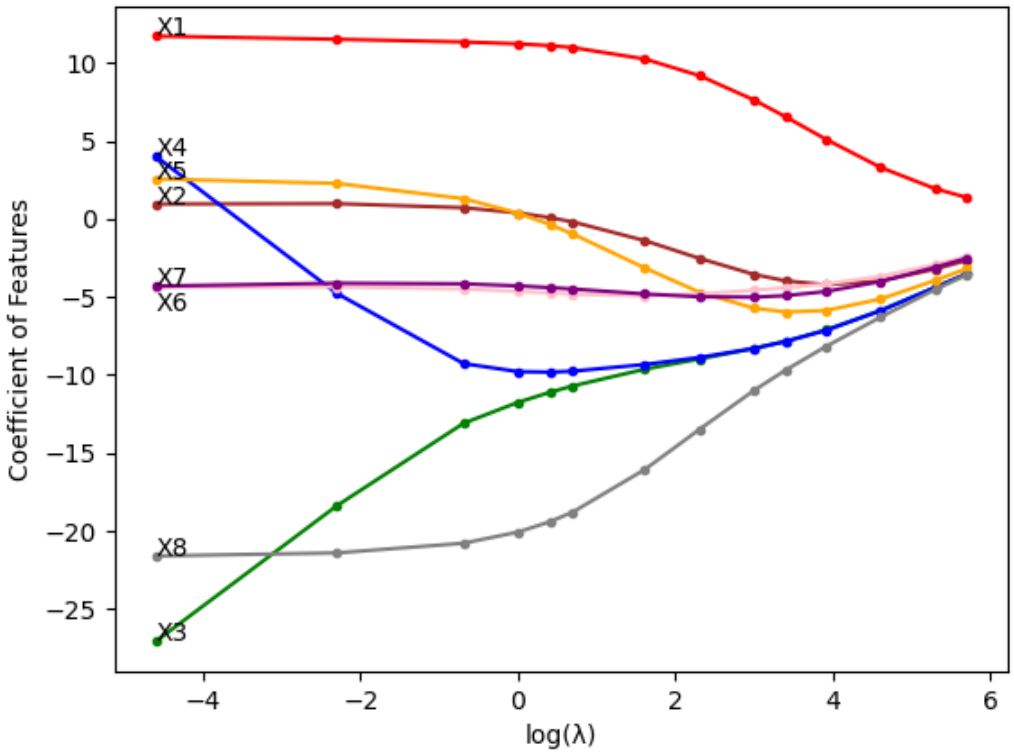


Commentary: Feature X1 might have about 0 correlation with other features. All features besides X1 are positively correlated with each other, among which the correlation between feature X3 and X4 is the highest. Most features have a high correlation, which may cause weights of the model to become unstable and decrease the model's prediction ability.

(b)

```
X1    38.0
X2    38.0
X3    38.0
X4    38.0
X5    38.0
X6    38.0
X7    38.0
X8    38.0
dtype: float64
```

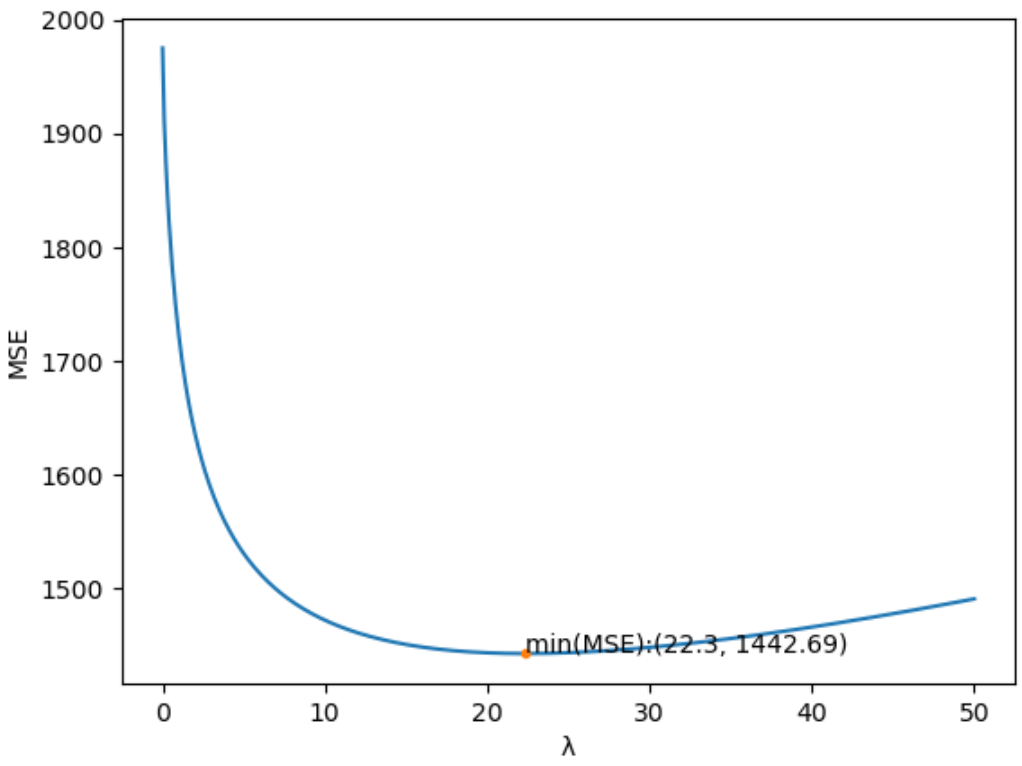
(c)



Commentary: The coefficient of features X6 and X7 are almost the same. The coefficient of all features from X1 to X8 approaches a somehow small value as the penalty coefficient λ increases. X3 and X4 tend to be the same with the rise of λ, but there is no apparent relationship with X5.

```python
# Question 2 (c)
df_Y = df.iloc[:,-1]
coef = list()

alpha_range = [0.01, 0.1, 0.5, 1, 1.5, 2, 5, 10, 20, 30, 50, 100, 200, 300]
colors = ['red', 'brown', 'green', 'blue', 'orange', 'pink', 'purple', 'grey']
for alpha in alpha_range:
    ridge = Ridge(alpha=alpha)
    ridge.fit(df_X, df_Y)
    coef.append(ridge.coef_.tolist())

coef = np.array(coef)

plt.figure()
plt.xlabel("log(λ)")
plt.ylabel("Coefficient of Features")
for i in range(coef.shape[1]):
    plt.plot(np.log(alpha_range), coef[:, i], marker = "o", markersize=3, color= colors[i])
    if i == 5:
        plt.text(np.log(alpha_range[0]), coef[0, i] - 1.5, 'X'+str(i+1)) # Overlap with i == 6
    else:
        plt.text(np.log(alpha_range[0]), coef[0, i], 'X'+str(i+1))
plt.savefig("Question 2 (c).png")
```

(d)



Commentary:

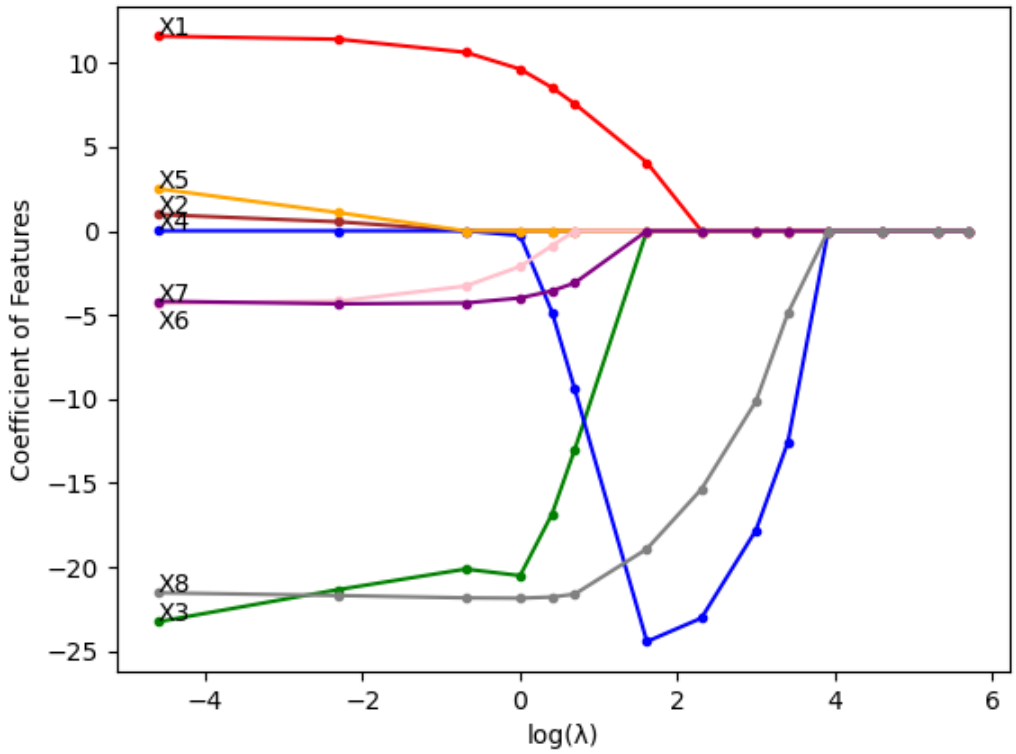
The minimum result for Ridge with different λ is 1442.6982227952913 when λ is 22.3. Hence, it is smaller than the result for OLS, which is 1975.4147393421736. According to the MSE for both models, I can see that the Ridge gives better prediction error. The main reason I think that there should be overfitting in OLS. Thus, the performance of OLS is not better than Ridge.

The larger the penalty coefficient λ is, the smaller the weight is, reducing the degree of overfitting and increasing generalization ability. Therefore, an appropriate increase of λ is helpful to improve the accuracy of the model on the test set (MSE may decrease). However, a significant increase in the value of λ will lead to the weight being too small, resulting in under-fitting, thus increasing MSE again.

```python
# Question 2 (d)
alpha_range = np.arange(0, 50.1, 0.1)
MSEs = list()
for alpha in alpha_range:
    # print("process: " + str(alpha * 100 // 50) + "%")
    MSE_ridge = 0
    for row in range(df.shape[0]): # Leave-One-Out Cross Validation for Ridge
        # Test
        test_X = df_X.loc[row].values.reshape(1, -1) # Convert Series to NumPy and then reshape
        test_Y = df.iloc[row, -1]
        # Train
        train_X = df_X.iloc[:row]
        train_X = train_X.append(df_X.iloc[row+1:])
        train_Y = df.iloc[:row, -1]
        train_Y = train_Y.append(df.iloc[row+1:, -1])

        # Get MSE for Ridge with different λ
        ridge = Ridge(alpha=alpha)
        ridge.fit(train_X, train_Y)
        predicted_Y = ridge.predict(test_X)
        predicted_Y = predicted_Y[0] # From np to integer
        MSE_ridge += pow((test_Y - predicted_Y), 2) # This should be squared error, not mean  squared error
    MSE_ridge /= df.shape[0] # Real MSE
    MSEs.append(MSE_ridge)

alpha = alpha_range[MSEs.index(min(MSEs))] # The λ when taking the min(MSEs)
print("MSE for Ridge: {}, The λ is {}".format(min(MSEs), alpha)) # MSE for Ridge: 1442.6982227952913, The λ is 22.3

plt.figure()
plt.xlabel("λ")
plt.ylabel("MSE")
plt.plot(alpha_range, MSEs)
MSE_min_row = alpha_range[MSEs.index(min(MSEs))] # x for the min(MSEs)
MSE_min_col = int(min(MSEs) * 100) / 100 # y for the min(MSEs)
plt.plot(MSE_min_row, MSE_min_col, marker = "o", markersize=3) # label the point of the min(MSEs)
plt.text(MSE_min_row, MSE_min_col, "min(MSE):("+str(MSE_min_row)+", "+str(MSE_min_col)+")")
plt.savefig("Question 2 (d).png")
```

```python
MSE_OLS = 0
for row in range(df.shape[0]): # Leave-One-Out Cross Validation for OLS
    # Test
    test_X = df_X.loc[row].values.reshape(1, -1) # Convert Series to NumPy and then reshape
    test_Y = df.iloc[row, -1]
    # Train
    train_X = df_X.iloc[:row]
    train_X = train_X.append(df_X.iloc[row+1:])
    train_Y = df.iloc[:row, -1]
    train_Y = train_Y.append(df.iloc[row+1:, -1])

    # Get squared error for OLS
    OLS = LinearRegression()
    OLS.fit(train_X, train_Y)
    predicted_Y = OLS.predict(test_X)
    predicted_Y = predicted_Y[0] # From np to integer
    MSE_OLS += pow((test_Y - predicted_Y), 2) # This should be squared error, not mean squared error

MSE_OLS /= df.shape[0] # Real MSE
print("MSE for LinearRegression: ", MSE_OLS) # MSE for LinearRegression:  1975.4147393421736
```

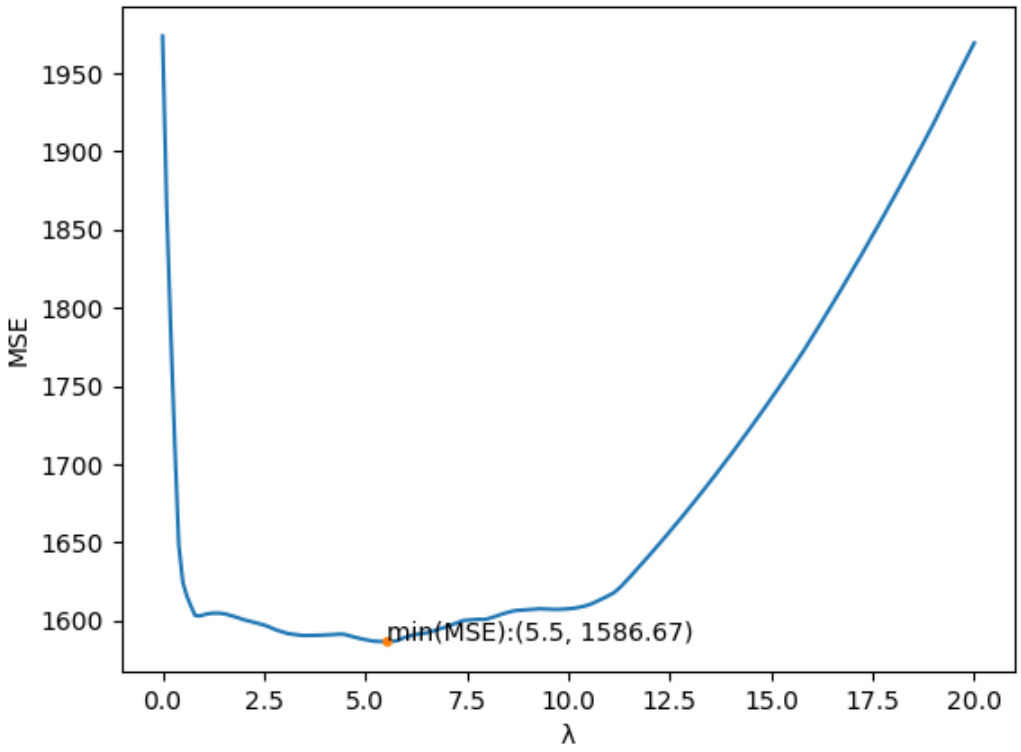
(e)



Commentary: The coefficient of all features from X1 to X8 approaches 0, as the penalty coefficient λ increases. As for X3, X4 and X5, I think X4 and X5 are similar initially, but gradually are different from each other with the increase of λ. X3 is different from X4 and X5. Eventually, they become 0.

```python
# Question 2 (e)
df_Y = df.iloc[:,-1]
coef = list()

alpha_range = [0.01, 0.1, 0.5, 1, 1.5, 2, 5, 10, 20, 30, 50, 100, 200, 300]
colors = ['red', 'brown', 'green', 'blue', 'orange', 'pink', 'purple', 'grey']
for alpha in alpha_range:
    lasso = Lasso(alpha=alpha)
    lasso.fit(df_X, df_Y)
    coef.append(lasso.coef_.tolist())

coef = np.array(coef)

plt.figure()
plt.xlabel("log(λ)")
plt.ylabel("Coefficient of Features")
for i in range(coef.shape[1]):
    plt.plot(np.log(alpha_range), coef[:, i], marker = "o", markersize=3, color= colors[i])
    if i == 5:
        plt.text(np.log(alpha_range[0]), coef[0, i] - 1.5, 'X'+str(i+1)) # Overlap with i == 6
    else:
        plt.text(np.log(alpha_range[0]), coef[0, i], 'X'+str(i+1))
plt.savefig("Question 2 (e).png")
```

(f)



Commentary:

The minimum result for Lasso with different λ is 1586.6715081806426 when λ is 5.5, which is better than OLS with the MSE at 1975.4147393421736.

The MSE for Lasso decreases first and then increases with the increase of λ.

```
# Question 2 (f)
alpha_range = np.arange(0, 20.1, 0.1)
MSEs = list()
for alpha in alpha_range:
    # print("process: " + str(alpha * 100 // 20) + "%")
    MSE_lasso = 0
    for row in range(df.shape[0]): # Leave-One-Out Cross Validation
        # Test
        test_X = df_X.loc[row].values.reshape(1, -1) # Convert Series to NumPy and then reshape
        test_Y = df.iloc[row, -1]
        # Train
        train_X = df_X.iloc[:row]
        train_X = train_X.append(df_X.iloc[row+1:])
        train_Y = df.iloc[:row, -1]
        train_Y = train_Y.append(df.iloc[row+1:, -1])

        # Get MSE for Lasso with different λ
        lasso = Lasso(alpha=alpha)
        lasso.fit(train_X, train_Y)
        predicted_Y = lasso.predict(test_X)
        predicted_Y = predicted_Y[0] # From np to integer
        MSE_lasso += pow((test_Y - predicted_Y), 2) # This should be squared error, not mean  squared error
    MSE_lasso /= df.shape[0] # Real MSE
    MSEs.append(MSE_lasso)

plt.figure()
plt.xlabel("λ")
plt.ylabel("MSE")
plt.plot(alpha_range, MSEs)
MSE_min_row = alpha_range[MSEs.index(min(MSEs))] # x for the min(MSEs)
MSE_min_col = int(min(MSEs) * 100) / 100 # y for the min(MSEs)
plt.plot(MSE_min_row, MSE_min_col, marker = "o", markersize=3) # label the point of the min(MSEs)
plt.text(MSE_min_row, MSE_min_col, "min(MSE):("+str(MSE_min_row)+", "+str(MSE_min_col)+")")
plt.savefig("Question 2 (f).png")

alpha = alpha_range[MSEs.index(min(MSEs))] # The λ when taking the min(MSEs)
print("MSE for LASSO: {}, The λ is {}".format(min(MSEs), alpha)) # MSE for LASSO: 1586.6715081806426, The λ is 5.5
```

(g)

The main difference between LASSO and Ridge is that Lasso stands for L1 regularization and Ridge for L2 regularization. L1 will compress some of the weights to zero, L2 will not. Hence, in (c) and (e), the plot of weights for Ridge approaches to a relatively small value, but the ones for LASSO are 0 eventually. Therefore, in some cases, L1 can be used for feature selection.

L1 has stronger robustness, and it is less sensitive to some outliers. L1 is more robust than L2. L2 is also not sensitive to outliers, and in the meanwhile, there may exist only one solution.

Both the models can be used in different cases. For instance, if I want to do some work relevant to feature selection, I prefer L1 (Lasso). In other cases, L2 is more suitable for common issues.

Question 3.

(a) From the "Homework 1 Forum: Q3a", I know that $|\langle Y, X\beta\rangle|$ is the absolute dot product of Y and $X\beta$.

$$|\langle Y, X\beta\rangle| = \left|\sum_i^n \sum_j^p Y_i \cdot X_{ij}\beta_j\right|$$
$$= \left|\sum_j^p \sum_i^n Y_i \cdot X_{ij}\beta_j\right|$$
$$= \left|\sum_j^p X_j^T Y \cdot \beta_j\right|$$

There will always exist a constant that $\max_j |X_j^T Y| \geq |X_j^T Y|$, where $j \in p$

Hence $|\langle Y, X\beta\rangle| \leq \left|\sum_j^p \max_j X_j^T Y \cdot \beta_j\right|$, where $\max_j X_j^T Y$ is a constant.

Hence $|\langle Y, X\beta\rangle| \leq \max_j |X_j^T Y| \cdot \sum_j |\beta_j|$

(b) The shape of Y is $n \times 1$, shape of X is $n \times p$, shape of β is $p \times 1$

Hence $(Y - X\beta)$ is a matrix of shape $n \times 1$.

$$\frac{1}{2}\|Y - X\beta\|_2^2 + \lambda\|\beta\|_1$$
$$= \frac{1}{2}(Y - X\beta)^T(Y - X\beta) + \lambda\sum_j |\beta_j|$$
$$= \frac{1}{2}(Y^T - \beta^T X^T)(Y - X\beta) + \lambda\sum_j |\beta_j|$$
$$= \frac{1}{2}(Y^T Y - Y^T X\beta - \beta^T X^T Y + \beta^T X^T \cdot X\beta) + \lambda\sum_j |\beta_j|$$

$\beta^T X^T Y = ((\beta^T X^T)^T)^T Y = (X\beta)^T Y$, where $(X\beta)^T$ shape $= 1 \times n$

Hence the shape of $(X\beta)^T Y$ is $1 \times n \cdot n \times 1 = 1 \times 1$

Hence $\beta^T X^T Y = (X\beta)^T Y = ((X\beta)^T Y)^T = Y^T X\beta$

Hence $\frac{1}{2}\|Y - X\beta\|_2^2 + \lambda\|\beta\|_1$

$$= \frac{1}{2}(Y^T Y - Y^T X\beta - \beta^T X^T Y + \beta^T X^T \cdot X\beta) + \lambda\sum_j |\beta_j|$$
$$= \frac{1}{2}(Y^T Y - 2Y^T X\beta + \beta^T X^T X\beta) + \lambda\sum_j |\beta_j|$$
$$= \frac{1}{2}(Y^T Y + \beta^T X^T X\beta) + \lambda\sum_j |\beta_j| - Y^T X\beta$$
$$= \frac{1}{2}(Y^T Y + \beta^T X^T X\beta) + \lambda\sum_j |\beta_j| - |\langle Y, X\beta\rangle|$$
$$\geq \frac{1}{2}(Y^T Y + \beta^T X^T X\beta) + \lambda\sum_j |\beta_j| - \max_j |X_j^T Y| \sum_j |\beta_j|$$
$$\geq \frac{1}{2}(Y^T Y + \beta^T X^T X\beta) + (\lambda - \max_j |X_j^T Y|)\sum_j |\beta_j|$$

$Y^TY \geq 0$, $\beta^T X^T X\beta = (X\beta)^T X\beta \geq 0$, $\lambda - \max_j |X_j^T Y| \geq 0$, $\sum_j |\beta_j| \geq 0$

Hence let $\hat{\beta} = 0_p$ to make $\beta^T X^T X\beta = (\lambda - \max_j |X_j^T Y|) \sum_j |\beta_j| = 0$

Hence the minimum result that $\frac{1}{2}\|Y - X\beta\|_2^2 + \lambda\|\beta\|_1$ can take is $\frac{1}{2}Y^TY$, when $\hat{\beta} = 0_p$.

(c) From Question 3(b), I know that

$$\frac{1}{2}\|Y - X\beta\|_2^2 + \lambda\|\beta\|_1 = \frac{1}{2}(Y^TY + \beta^T X^T X\beta) + \lambda\sum_j |\beta_j| - Y^TX\beta$$
$$= \frac{1}{2}Y^TY, \text{ when } \hat{\beta} = 0_p$$

If $\beta \neq 0_p$, there are two cases for $X\beta$. $\begin{cases} \|X\beta\|_2 = 0 \\ \|X\beta\|_2 \neq 0 \end{cases}$

Hence if $\beta \neq 0_p$ and $\|X\beta\|_2 = 0$, $\frac{1}{2}\|Y - X\beta\|_2^2 + \lambda\|\beta\|_1 = \frac{1}{2}Y^TY + \lambda\sum_j |\beta_j| > \frac{1}{2}Y^TY$

If $\beta \neq 0_p$ and $\|X\beta\|_2 \neq 0$, then $\beta^T X^T X\beta > 0$, $\lambda\sum_j |\beta_j| - Y^TX\beta > 0$.

Hence, $\frac{1}{2}\|Y - X\beta\|_2^2 + \lambda\|\beta\|_1 = \frac{1}{2}(Y^TY + \beta^T X^T X\beta) + \lambda\sum_j |\beta_j| - Y^TX\beta \geq \frac{1}{2}Y^TY$

Hence, only if $\hat{\beta} = 0_p$, the minimum loss of $L(\beta)$ can be obtained.