COMP9417 - Machine Learning and Data Mining
T2-2021

Homework 1: Linear Regression & Friends

Daksh Mukhra
[redacted]

Q1)

a) we shall derive $\tilde{\beta}_0$ $\tilde{\beta}_1$ from the Least Squares minimization criterion.

→ note we have univariate model.

$$L(\tilde{\beta}_0, \tilde{\beta}_1) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{\beta}_0 - \tilde{\beta}_1 \tilde{x}_i)^2$$

$$= \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{\beta}_0 - \tilde{\beta}_1 (c(x_i+d)))^2 \quad // \text{ as } \tilde{x} = c(x_i+d)$$

$$= \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{\beta}_0 - \tilde{\beta}_1 c(x_i+d))^2$$

→ differentiate wrt $\tilde{\beta}_0$

$$\frac{\partial L}{\partial \tilde{\beta}_0} = \frac{\partial}{\partial \tilde{\beta}_0} \cdot \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{\beta}_0 - \tilde{\beta}_1 c(x_i+d))^2$$

$$= \frac{1}{n}\sum_{i=1}^{n} -2(y_i - \tilde{\beta}_0 - \tilde{\beta}_1 c(x_i+d)).$$

$$= -2\left[\frac{1}{n}\sum_{i=1}^{n} y_i - \frac{\tilde{\beta}_0}{n}\sum_{i=1}^{n} 1 - \frac{\tilde{\beta}_1 c}{n}\sum_{i=1}^{n}(x_i+d)\right]$$

$$= -2\left[\bar{y} - \tilde{\beta}_0 - \tilde{\beta}_1 c\bar{x} - \tilde{\beta}_1 cd\right]$$

→ set =0 & solve.

$$\bar{y} - \tilde{\beta}_0 - \tilde{\beta}_1 c\bar{x} - \tilde{\beta}_1 cd = 0$$

$$\tilde{\beta}_0 = \bar{y} - \tilde{\beta}_1 c(\bar{x} + d)$$

→ differentiate wrt $\tilde{\beta}_1$

$$\frac{\partial L}{\partial \tilde{\beta}_1} = \frac{\partial}{\partial \tilde{\beta}_1} \cdot \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{\beta}_0 - \tilde{\beta}_1 c(x_i+d))^2 \longrightarrow$$

$$= \frac{1}{n}\sum_{i=1}^{n} -2c(x_i+d)(y_i - \tilde{\beta}_0 - \tilde{\beta}_1 c(x_i+d))$$

$$= -2\left[\frac{1}{n}\sum_{i=1}^{n} c(x_i+d)y_i - \tilde{\beta}_0 c(x_i+d) - \tilde{\beta}_1 c^2(x_i+d)^2\right]$$

$$= -2\left[\frac{1}{n}\sum_{i=1}^{n} cx_iy_i + cdy_i - \tilde{\beta}_0 cx_i + \tilde{\beta}_0 cd - \tilde{\beta}_1 c^2 x_i^2 - \tilde{\beta}_1 c^2 d^2 - 2\tilde{\beta}_1 c^2 x_i d\right]$$

$$\frac{\partial L}{\partial \tilde{\beta}_1} = -2\left[\frac{1}{n}\sum_{i=1}^{n} c x_i y_i + c d y_i - \tilde{\beta}_0 c d - \tilde{\beta}_0 c x_i - \tilde{\beta}_1 c^2 x_i^2 - \tilde{\beta}_1 c^2 d^2 - 2\tilde{\beta}_1 c^2 x_i d\right]$$

$$\frac{\partial L}{\partial \tilde{\beta}_1} = -2\left[c\overline{xy} + cd\bar{y} - \tilde{\beta}_0 cd - \tilde{\beta}_0 c\bar{x} - \tilde{\beta}_1 c^2 \overline{x^2} - \tilde{\beta}_1 c^2 d^2 - 2\tilde{\beta}_1 c^2 d\bar{x}\right]$$

→ equate to 0 & solve

$$\rightarrow c d\bar{y} + c\overline{xy} - \tilde{\beta}_0 cd - \tilde{\beta}_0 c\bar{x} - \tilde{\beta}_1 c^2 \overline{x^2} - \tilde{\beta}_1 c^2 d^2 - 2\tilde{\beta}_1 c^2 d\bar{x} = 0$$

$$\rightarrow -\tilde{\beta}_1(+c^2\overline{x^2} + c^2 d^2 + 2c^2 d\bar{x}) + c(d\bar{y} + \overline{xy}) - \tilde{\beta}_0(cd + c\bar{x}) = 0$$

$$\tilde{\beta}_1 = \left[\frac{\tilde{\beta}_0 c(d+\bar{x}) - c(d\bar{y} + \overline{xy})}{+c^2\overline{x^2} + c^2 d^2 + 2c^2 d\bar{x}}\right] \times -1.$$

Sub in $\tilde{\beta}_0 = \bar{y} - \tilde{\beta}_1 c(\bar{x} + d)$

$$\tilde{\beta}_1 = \frac{c(d\bar{y} + \overline{xy}) - [\bar{y} - \tilde{\beta}_1 c(\bar{x}+d)]c(d+\bar{x})]}{+c^2\overline{x^2} + c^2 d^2 + 2c^2 d\bar{x}}$$

$$\tilde{\beta}_1 = \frac{c(d\bar{y} + \overline{xy}) - c(d+\bar{x})\bar{y} + \tilde{\beta}_1 c^2(\bar{x}+d)^2}{c^2\overline{x^2} + c^2 d^2 + 2c^2 d\bar{x}}$$

$$\tilde{\beta}_1[+c^2\overline{x^2} + c^2 d^2 + 2c^2 d\bar{x}] = c(d\bar{y} + \overline{xy}) - c\bar{y}(d+\bar{x}) + \tilde{\beta}_1 c^2(\bar{x}+d)^2$$

$$\tilde{\beta}_1 c^2[\overline{x^2} + d^2 + 2d\bar{x}] - \tilde{\beta}_1 c^2(\bar{x}+d)^2 = c(d\bar{y} + \overline{xy}) - c\bar{y}(d+\bar{x})$$

$$\tilde{\beta}_1 c^2[\overline{x^2} + d^2 + 2d\bar{x} - \bar{x}^2 - d^2 - 2\bar{x}d] = c(d\bar{y} + \overline{xy}) - c\bar{y}(d+\bar{x})$$

next pg →

$$\tilde{\beta}_1 c^2[\overline{x^2} - \bar{x}^2] = c(d\bar{y} + \overline{xy}) - c\bar{y}(d+\bar{x})$$

$$\tilde{\beta}_1 = \frac{d\bar{y} + \overline{xy} - \bar{y}d - \bar{y}\bar{x}}{c[\overline{x^2} - \bar{x}^2]} = \frac{\overline{xy} - \bar{x}\bar{y}}{c[\overline{x^2} - \bar{x}^2]}$$

$\therefore \boxed{\tilde{\beta}_1 = \frac{1}{c}\left[\hat{\beta}_1\right]}$ where $\hat{\beta}_1$ is the OLS estimator of β_1 for $Y \sim X$.

from earlier

$$\rightarrow \tilde{\beta}_0 = \bar{y} - \frac{1}{c}\left[\hat{\beta}_1\right] c(\bar{x} + d)$$

$$\tilde{\beta}_0 = \bar{y} - \hat{\beta}_1(\bar{x} + d)$$

$$\boxed{\tilde{\beta}_0 = \hat{\beta}_0 + d}$$ where $\hat{\beta}_0$ is the OLS estimator of β_0 for $Y \sim X$.

→ we now find error variance:

$$\tilde{\sigma} = \sqrt{\frac{\tilde{e}^T\tilde{e}}{n-2}} \quad \& \quad \hat{\sigma} = \sqrt{\frac{\hat{e}^T e}{n-2}}$$

$$\tilde{\sigma}^2 = \frac{\tilde{e}^T\tilde{e}}{n-2} = \frac{(y_i - \tilde{y}_i)^T(y_i - \tilde{y}_i)}{n-2}$$

$$\tilde{\sigma}^2 = \frac{\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2}{n-2} = \frac{\sum_{i=1}^{n}(y_i - (\tilde{\beta}_0 + \tilde{\beta}_1 c x_i))^2}{n-2}$$

$$\rightarrow \frac{\sum_{i=1}^{n}(y_i - (\tilde{\beta}_0 + \tilde{\beta}_1(c(x_i + d))))^2}{n-2} = \tilde{\sigma}^2$$

// sub in $\tilde{\beta}_0, \tilde{\beta}_1$

$$\rightarrow \frac{\sum_{i=1}^{n}(y_i - \tilde{\beta}_0 - \tilde{\beta}_1 c(x_i + d))^2}{n-2} = \frac{\sum_{i=1}^{n}(y_i - (\hat{\beta}_0 + d) - \left[\frac{1}{c}\hat{\beta}_1\right] c(x_i + d))^2}{n-2}$$

$$\rightarrow \frac{\sum_{i=1}^{n}(y_i - \hat{\beta}_0 - d - \hat{\beta}_1 x_i - \hat{\beta}_1 d)^2}{n-2} = \frac{\sum_{i=1}^{n}(y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i) - d(\hat{\beta}_1 + 1))^2}{n-2}$$

$$\tilde{\sigma}^2 = \frac{\sum_{i=1}^{n}(y_i - \hat{y} - d(\hat{\beta}_1 + 1))^2}{n-2} = \boxed{\frac{\sum_{i=1}^{n}(\hat{e}_i - d(\hat{\beta}_1 + 1))^2}{n-2}}$$

Q(1) b)

→ let $x_i = \begin{cases} 1 & \text{dosage if } i = n_1+1, \dots, n \\ 0 & \text{no dosage if } i = 1, \dots, n_1 \end{cases}$

→ $\bar{Y}_T$ is the ~~sample~~ mean of treatment group

$\bar{Y}_P$ is the mean of placebo group.

→ let $n_2 + n_1 = n \quad \therefore \quad n_2 = (n - n_1)$

↳ Hence let n_2 be the number of patients that took dosage.

n_1 be the number of patients that took placebo.

→ we have the regression model $Y = \beta_0 + \beta_1 X + \epsilon$ where $\epsilon \sim N(0,$

→ From the usual minimization proceedure we obtain the OLS estimates:

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}\bar{x} \quad \& \quad \hat{\beta}_1 = \frac{\overline{xy} - \bar{x}\bar{y}}{\overline{x^2} - \bar{x}^2}$$

→ Note that the group means are.

$$\bar{y}_P = \frac{\sum_{i=1}^{n_1} y_i}{n_1} \qquad \bar{y}_T = \frac{\sum_{i=1}^{n} y_i}{n_2}$$

→ Also Note the following derivations:

① $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{1}{n} \cdot n_2 = \frac{n_2}{n_1 + n_2}$ // as only people who are treated = 1.

② $(\bar{x})^2 = \left(\frac{n_2}{n_1 + n_2}\right)^2$

③ $\sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} 1^2 + 0 + 1^2 \dots = n_2.$

④ $\bar{x}\bar{y} = \frac{n_2}{n_1 + n_2}\left(\frac{1}{n}\sum_{i=1}^{n} y_i\right)$

⑤ $\sum_{i=1}^{n} x_i y_i = \sum_{i=1}^{n_1} (0)\, y_i + \sum_{i=n_1+1}^{n} (1)\, y_i = \sum_{i=1}^{n_2} y_i$

// as $n_2 = (n - n_1)$

→ see nxt pg

→ we now find $\hat{\beta}_1$ interms of group mean.

→ $$\hat{\beta}_{1,OLS} = \frac{\frac{1}{n}\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i \frac{1}{n^2}}{\frac{1}{n}\sum_{i=1}^{n} x^2 - \left(\frac{1}{n}\sum_{i=1}^{n} x\right)^2}$$

// where $n = n_1 + n_2$

using ①, ③, ④, ⑤ as descr earlier

$$\hat{\beta}_1 = \frac{\frac{1}{n_1+n_2}\sum_{i=1}^{n_2} y_i - \left(\frac{n_2}{n_1+n_2}\right)\cdot \sum_{i=1}^{n} y_i \left(\frac{1}{n_1+n_2}\right)}{\frac{n_2}{n_1+n_2} - \left(\frac{n_2}{n_1+n_2}\right)^2}$$

(from ①; from ③; from ②)

→ as $n = n_1 + n$

$$\hat{\beta}_1 = \frac{\frac{1}{n_1+n_2}\sum_{i=1}^{n_2} y_i - \frac{n_2}{n_1+n_2}\left(\frac{1}{n_1+n_2}\cdot\left(\sum_{i=1}^{n_1} y_i + \sum_{i=1}^{n_2} y_2\right)\right)}{\frac{n_2(n_1+n_2) - n_2^2}{(n_1+n_2)^2}}$$

$$\hat{\beta}_1 = \left(\frac{1}{n_1+n_2}\right)\left[\sum_{i=1}^{n_2} y_i - \frac{n_2}{n_1+n_2}\left(\sum_{i=1}^{n_1} y_i + \sum_{i=1}^{n_2} y_2\right)\right] \Big/ \frac{n_1 n_2 + n_2^2 - n_2^2}{(n_1+n_2)^2}$$

see nxt pg for $\hat{\beta}_0$ →

$$\hat{\beta}_1 = \frac{(n_1+n_2)}{(n_1+n_2)}\frac{1}{} \cdot \left[1 - \frac{n_2}{n_1+n_2}\right]\sum_{i=1}^{n_2} y_i - \frac{n_2}{n_1+n_2}\sum_{i=1}^{n_1} y_1 \Big/ \frac{n_1 n_2}{(n_1+n_2)}$$

$$= \frac{1}{n_2}\sum_{i=1}^{n_2} y_i - \frac{1}{n_1}\sum_{i=1}^{n_1} y$$

$$\hat{\beta}_1 = \frac{n_1 + n_2 - n_2}{n_1+n_2}\sum_{i=1}^{n_2} y_i - \frac{n_2}{n_1+n_2}\sum_{i=1}^{n_1} y_i \Big/ \frac{n_1 n_2}{n_1+n_2}$$

$$\hat{\beta}_1 = \bar{y}_2 - \bar{y}_1$$

where 2 = , 1 = i

$$\hat{\beta}_1 = \frac{n_1}{n_1+n_2} \times \frac{n_1+n_2}{n_1 n_2} * \sum_{i=1}^{n_2} y_i - \frac{n_1+n_2}{n_1 n_2} \times \frac{n_2}{n_1+n_2}\sum_{i=1}^{n_1} y_i$$

To find $\hat{\beta}_0$ we do a similar procedure

$$\hookrightarrow \hat{\beta}_{0\,OLS} = \bar{y} - b_1 \bar{x}$$

$$\hat{\beta}_0 = \frac{1}{n}\sum_{i=1}^{n} y_i - \left(\bar{y}_2 - \bar{y}_1\right)\frac{n_2}{n_1+n_2}$$

$$// n = n_1 + n_2$$

$$\hat{\beta}_0 = \left(\frac{1}{n_1+n_2}\right)\left[\sum_{i=1}^{n_1} y_i + \sum_{i=1}^{n_2} y_i\right] - \left[\frac{n_2}{n_1+n_2}\right]\left(\bar{y}_2 - \bar{y}_1\right)$$

$$\hat{\beta}_0 = \left(\frac{1}{n_1+n_2}\right)\left[\sum_{i=1}^{n_1} y_i + \sum_{i=1}^{n_2} y_i - n_2\left(\frac{1}{n_2}\sum_{i=1}^{n_2} y_i - \frac{1}{n_1}\sum_{i=1}^{n_1} y_i\right)\right]$$

$$\hat{\beta}_0 = \left(\frac{1}{n_1+n_2}\right)\left[\sum_{i=1}^{n_1} y_i + \frac{n_2}{n_1}\sum_{i=1}^{n_1} y_i + \cancel{\sum_{i=1}^{n_2} y_i} - \cancel{\sum_{i=1}^{n_2} y_i}\right]$$

$$\hat{\beta}_0 = \left(\frac{1}{n_1+n_2}\right)\left[\left(1 + \frac{n_2}{n_1}\right)\sum_{i=1}^{n_1} y_i\right]$$

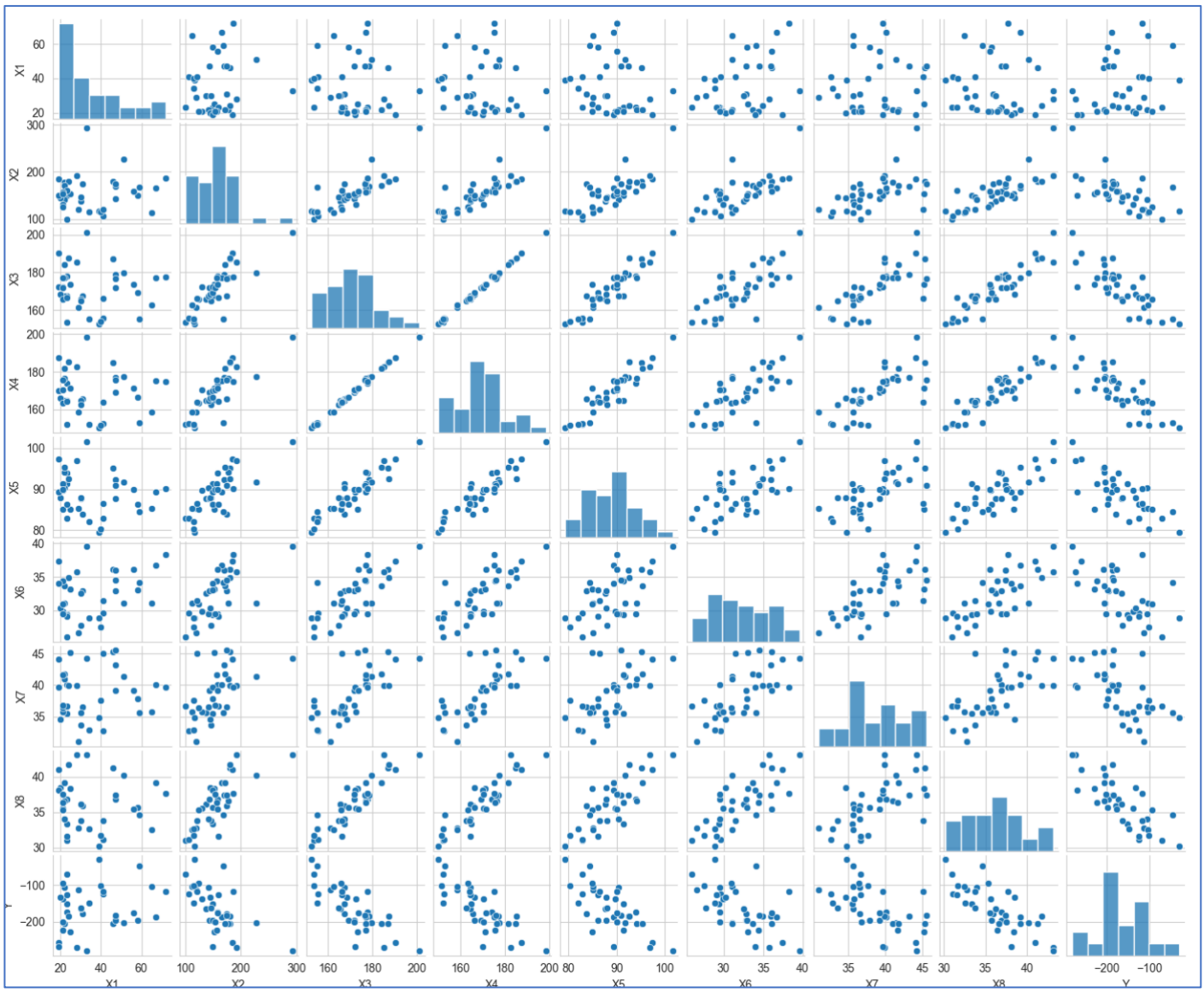
$$\hat{\beta}_0 = \frac{n_1+n_2}{n_1} \times \frac{1}{n_1+n_2} \times \sum_{i=1}^{n_1} y_i$$

$$\boxed{\hat{\beta}_0} = \frac{\cancel{n_1+n_2}}{\cancel{n_1+n_2}} \times \frac{1}{n_1}\sum_{i=1}^{n_1} y_i = \boxed{\bar{y}_1} \quad \checkmark$$

$$\therefore \left[\begin{array}{l} \hat{\beta}_0 = \bar{y}_P \\ \hat{\beta}_1 = \bar{y}_T - \bar{y}_P \end{array}\right]$$

Question 2)

Q2 a) Pairs plot to assess Multicollinearity



From the plot above we can see that many of the features are highly correlated e.g. (x4,x3), (x8,x3) and (x4,x5). Typically, a feature's coefficient represents the effect of a 1-unit change in that variable on the response while holding all other variables constant. However as seen above a very strong correlation between variables (e.g. x3,x4) it becomes difficult for the linear model to estimate the relationship between each feature and the response *independently* because the features tend to change in unison. So we have an imprecise estimate of the effect of independent changes in the features. **A model with multicollinear data has very high variance but very low bias which results in overfitting.**

Q2 b)

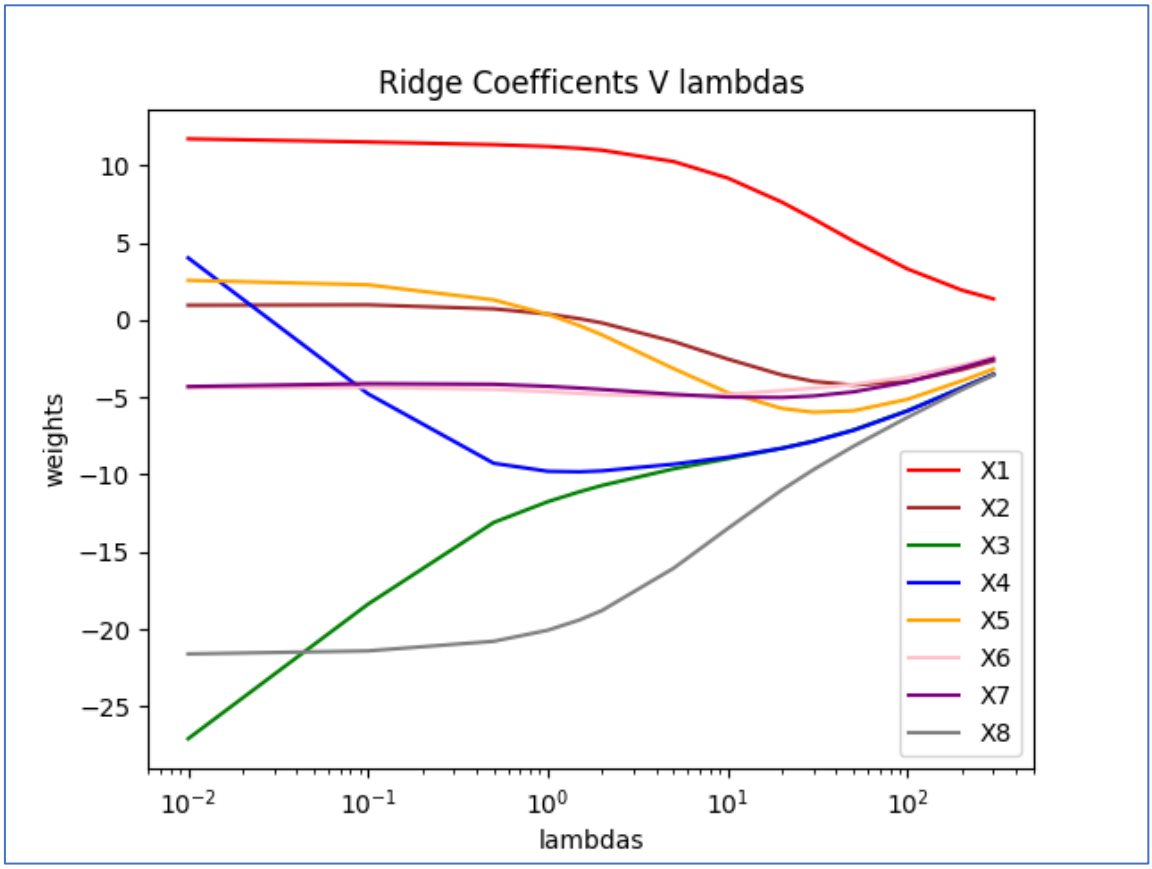
As we can see below the sum of squares is ≈ 38. With mean ≈ 0 and var 1 for each factor.

```
----mean----
X1   -2.921640e-17
X2    2.859555e-16
X3   -1.602519e-15
X4    1.804112e-16
X5    1.845015e-15
X6   -1.051790e-16
X7    9.714451e-17
X8   -1.234393e-16
Y    -1.648849e+02
dtype: float64
```

```
----variance----
X1     1.000000
X2     1.000000
X3     1.000000
X4     1.000000
X5     1.000000
X6     1.000000
X7     1.000000
X8     1.000000
Y     58.857318
dtype: float64
```

```
----sum of sq's----
X1 - 38.0
X2 - 38.00000000000001
X3 - 38.00000000000002
X4 - 37.99999999999998
X5 - 38.0
X6 - 38.0
X7 - 38.000000000000014
X8 - 37.999999999999986
```

Q2 c) Graph produced mapping the change in coefficient values as Lambda values change

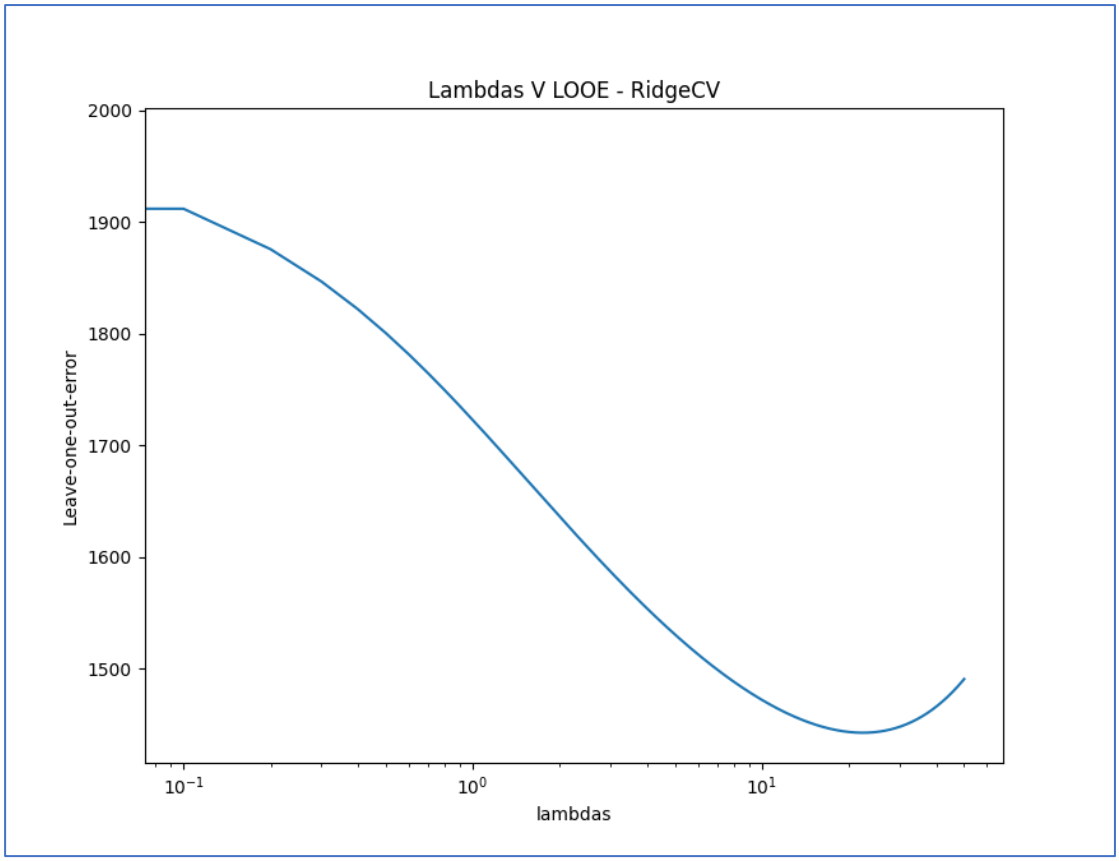


From the plot as alpha increases the coefficients start to converge to 0. This is the power of ridge regression, making the coefficients smaller to limit the collinearity between predictors. For regressor 3 it completely flipped the sign on the coefficient, and it converged to a smaller value. Although regressor 4 remained the same sign the change in magnitude for higher lambda values was significant. Regressor 5 remained similar. Also, the coefficient of regressor X1 continues to remain important even with increasing penalty.

Code used to produce plot in Q2 c)

```python
# part c - Ridge regression Plots
# summarize shape
print(normalized_df.shape)
y = normalized_df['Y']
X = normalized_df.drop('Y',axis=1)

# get our coefficents for each lambda
lambdas = np.array([0.01, 0.1, 0.5, 1, 1.5, 2, 5, 10, 20, 30, 50, 100, 200, 300])
ridge = Ridge()
coefs = []
for a in lambdas:
    ridge.set_params(alpha = a)
    ridge.fit(X, y)
    coefs.append(ridge.coef_)

print(coefs)

# plot
ax = plt.gca()
labels = ['X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X7', 'X8']
color_cycle= ['red', 'brown', 'green', 'blue', 'orange', 'pink', 'purple', 'grey']
for i in range(len(coefs[0])):
    ax.plot(lambdas,[pt[i] for pt in coefs], color=color_cycle[i],label = f"{labels[i]}")

ax.set_xscale('log')
ax.autoscale()
plt.xlabel('lambdas')
plt.ylabel('weights')
plt.title('Ridge Coefficents V lambdas')
plt.legend()
plt.show()
```

Q2 d) Graph produced through LOOCV -Ridge



The LOOE from OLS = 1975.414739342173
Minimum LOOE from ridge = 1442.6982227952915
Best lambda = 22.3

We can see from the graph that the Error gradually reduces to a minimum at lambda = 22.3 and then slowly rises again. Compared to OLS which has a MSE of 1975.4 the ridge regression produces an error of 1442.69 at its best. This matches what we know about ridge regression as it produces estimators that have inherently less variance then OLS but are a slightly biased

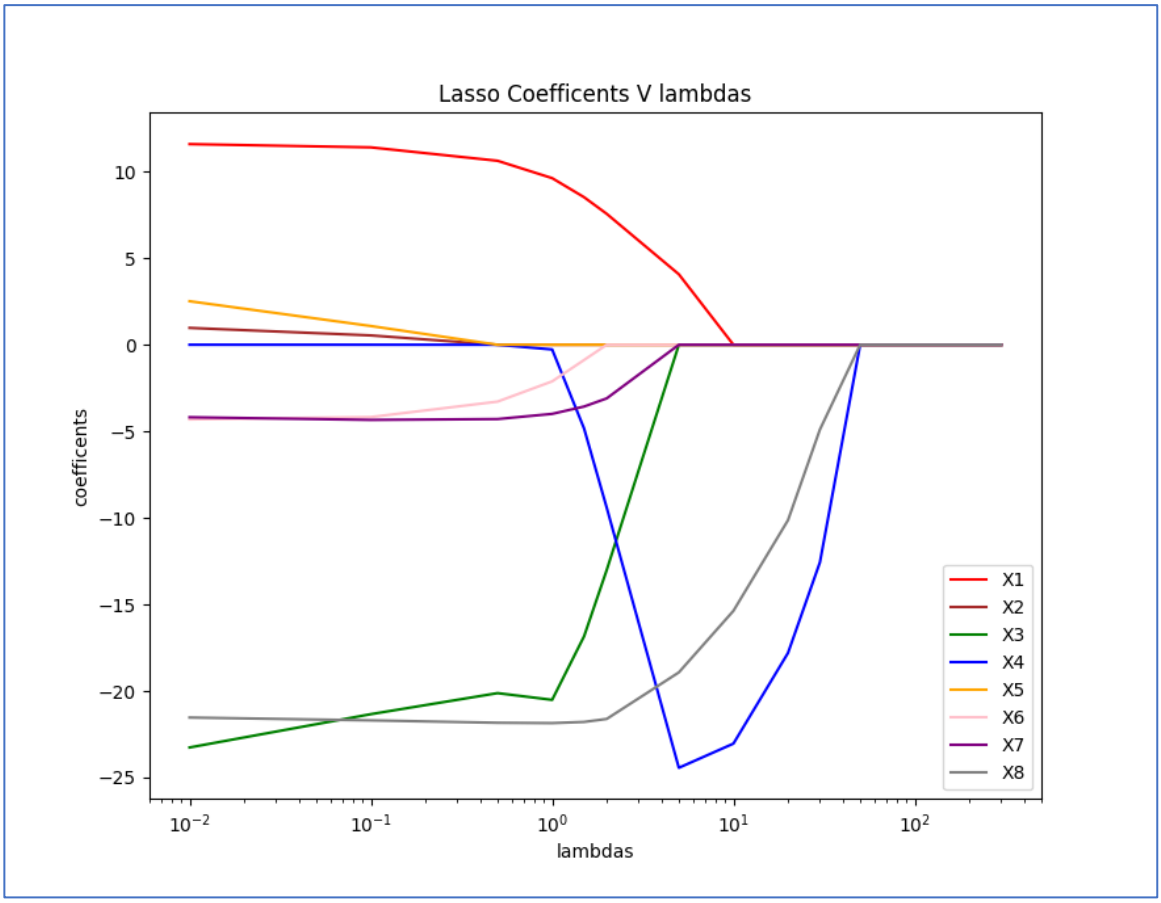
Code used to produce the above results in Q2d) :

```
#part d LOOCV from scratch
#find optimal lambda
lambdas = np.arange(start=0, stop=50.1, step=0.1)
copy_X = normalized_df.drop('Y',axis=1)
copy_y = normalized_df['Y']

ridge_d = Ridge()
row_it = 0
mse_for_lambdas = []

for a in lambdas:
    sq_errors = []
    while row_it < 38:  # Note : one observation is equal to one row per defintion of MLR
        dropped_row_X = copy_X.loc[row_it]
        dropped_y = copy_y.iloc[row_it]

        #remove the ith obs
        ith_removed_X = copy_X.drop(row_it, axis = 0)
        ith_removed_y = copy_y.drop(row_it, axis = 0)
        ridge_d.set_params(alpha = a) # set to current lambda
        ridge_d.fit(ith_removed_X, ith_removed_y)
        dropped_row_X = dropped_row_X.values.reshape(1, -1)

        #get prediction error
        prediction = ridge_d.predict(dropped_row_X)
        prediction_error = pow((dropped_y - prediction[0]),2)
        sq_errors.append(prediction_error)
        row_it= row_it + 1

    row_it = 0
    mse = (sum(sq_errors))/38 # find the Leave one out error average for lambda = a
    mse_for_lambdas.append(mse)

#find best lambda
minMSE = min(mse_for_lambdas)
minMSE_index = mse_for_lambdas.index(min(mse_for_lambdas))
best_lambda = lambdas[minMSE_index]
print(f"MIN mse =  {minMSE} --------------------------")
print(f"best lambda =  {best_lambda} --------------------------")

#plot
ax = plt.gca()
ax.set_xscale('log')
ax.autoscale()
plt.xlabel('lambdas')
plt.ylabel('Leave-one-out-error')
plt.title('Lambdas V LOOE - RidgeCV')
plt.plot(lambdas, mse_for_lambdas, color ="tab:blue")
plt.show()
```

```python
ols = LinearRegression()
sq_errors = []
# LOOCV for OLS
while row_it < 38:
    dropped_row_X = copy_X.loc[row_it]
    dropped_y = copy_y.iloc[row_it]
    ith_removed_X = copy_X.drop(row_it, axis = 0)
    ith_removed_y = copy_y.drop(row_it, axis = 0)

    ols.fit(ith_removed_X, ith_removed_y)
    dropped_row_X = dropped_row_X.values.reshape(1, -1)

    prediction = ols.predict(dropped_row_X)
    prediction_error = pow((dropped_y - prediction[0]),2)
    sq_errors.append(prediction_error)
    row_it= row_it + 1

mse = (sum(sq_errors))/38
print(f"the LOOE from OLS is {mse}")
```

Code used to produce results for OLS comparison in Q2d)

Q2e) Graph produced mapping the change in coefficient values as Lambda values change

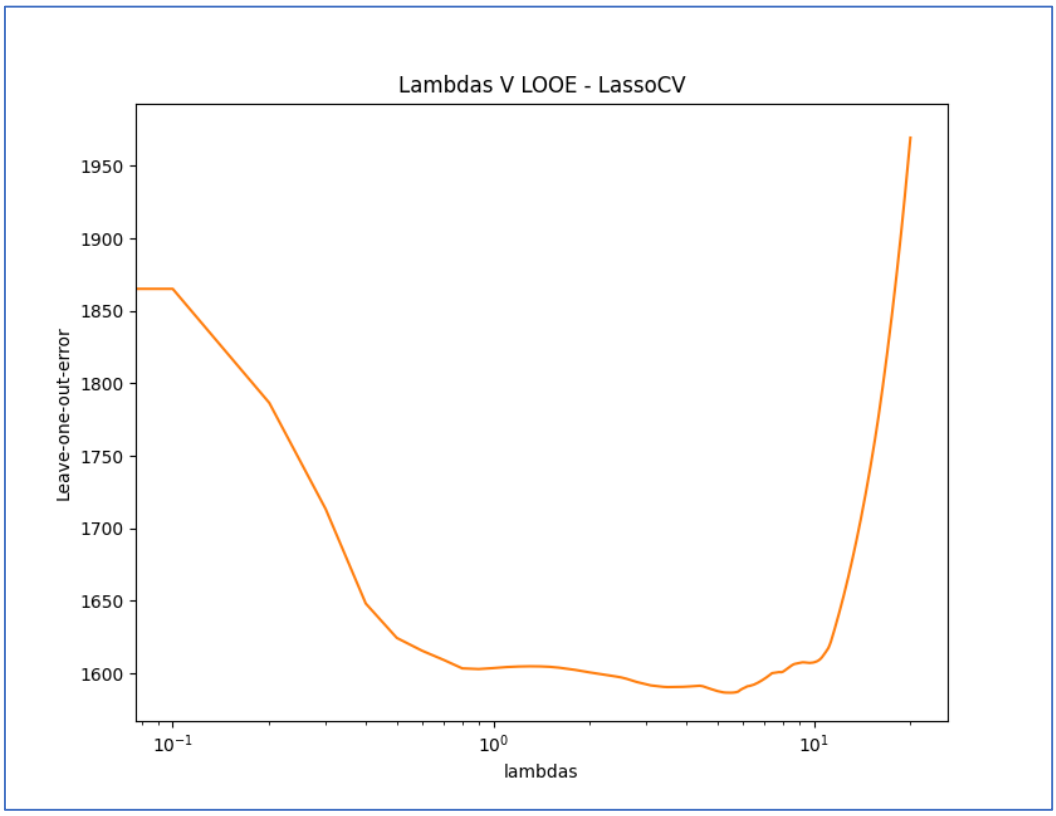


As seen the above graph compared to the Ridge Regression done in part c) there a stark difference present between the path coefficient's take with respect to high penalties. From the graph all coefficients go to 0 at high enough lambdas. The coefficient on X1 takes an expect path and remains important While the coefficient on X4 is initially 0 then a has a significant negative affect while rising to zero again. Indicating further investigation between X4 and y is needed.

Code used to produce above results for Q2e)

```
#Q2 part e
#get our lambdas
lambdas = np.array([0.01, 0.1, 0.5, 1, 1.5, 2, 5, 10, 20, 30, 50, 100, 200, 300])

copy_X = normalized_df.drop('Y',axis=1)
copy_y = normalized_df['Y']
lasso = Lasso()
coefs = []

for a in lambdas:
    lasso.set_params(alpha = a)
    lasso.fit(copy_X, copy_y)
    coefs.append(lasso.coef_)
#sanity check
print(coefs)
#plot
ax = plt.gca()
labels = ['X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X7', 'X8']
# ax.plot(lambdas, coefs)
color_cycle= ['red', 'brown', 'green', 'blue', 'orange', 'pink', 'purple', 'grey']
for i in range(len(coefs[0])):
    ax.plot(lambdas,[pt[i] for pt in coefs], color=color_cycle[i],label = f"{labels[i]}")

ax.set_xscale('log')
ax.autoscale()
plt.axis('tight')
plt.xlabel('lambdas')
plt.ylabel('coefficents')
plt.title('Lasso Coefficents V lambdas')
plt.legend()
plt.show()
```

Q2 f) Graph produced from LOOCV – Lasso



MIN LOOE = 1586.6715081806428
best lambda = 5.5

From the graph we can see that like Ridge regression the error drops significantly for 'medium' levels of lambda but dramatically increases after a certain point.

Code used to the above results for Q2 f)

```
#Q2 part f
lambdas = np.arange(start=0, stop=20.1, step=0.1)
copy_X = normalized_df.drop('Y',axis=1)
copy_y = normalized_df['Y']

lasso = Lasso()
row_it = 0
mse_for_lambdas = []

for a in lambdas:
    sq_errors = []

    while row_it < 38:
        dropped_row_X = copy_X.loc[row_it]
        dropped_y = copy_y.iloc[row_it]
        #remove the ith obs
        ith_removed_X = copy_X.drop(row_it, axis = 0)
        ith_removed_y = copy_y.drop(row_it, axis = 0)

        #Predict
        lasso.set_params(alpha = a) # set to current lambda
        lasso.fit(ith_removed_X, ith_removed_y)
        dropped_row_X = dropped_row_X.values.reshape(1, -1)
        prediction = lasso.predict(dropped_row_X)
        prediction_error = pow((dropped_y - prediction[0]),2)
        sq_errors.append(prediction_error)

        row_it= row_it + 1

    row_it = 0
    mse = (sum(sq_errors))/38
    mse_for_lambdas.append(mse)
# Find minimum LOOE for a lambda
minMSE = min(mse_for_lambdas)
minMSE_index = mse_for_lambdas.index(min(mse_for_lambdas))
best_lambda = lambdas[minMSE_index]
print(f"MIN mse =  {minMSE} --------------------------")
print(f"best lambda =  {best_lambda} --------------------------")
#plot graph
ax = plt.gca()
ax.set_xscale('log')
ax.autoscale()
plt.xlabel('lambdas')
plt.ylabel('Leave-one-out-error')
plt.title('Lambdas V LOOE - LassoCV')
plt.plot(lambdas, mse_for_lambdas, color ="tab:orange")
plt.show()
```

Q2 g)

The differences between ridge and lasso regression lie in the minimisation criterion and specifically we can observe how the different criteria have affected our predictions. For our case the minimum error for ridge regression produced was 1442 compared to 1586 for lasso found through out of sample prediction. So, Ridge regression would be the preferred model for prediction.

Also, our data was shown to be highly correlated and gave little reason to choose between different linear combinations of colinear predictors. Ridge regression will perform better as it tends to shrink the "group" (the correlated variables) proportionately reducing variance. Whereas Lasso doesn't since it promotes setting individual regression coefficients to zero to reduce model size.

Q 3)

a) show $|\langle Y, XB\rangle| \leq \max_j |X_j^T Y| \sum_j |B_j|$

LHS: $|\langle Y, XB\rangle| = |\langle XB, Y\rangle|$

$= |\langle B, X^T Y\rangle|$

$= \langle |(X^T Y)|, |B| \rangle$

→ let $X^T Y = \alpha$ where α is a vector.

hence LHS $= \langle |\alpha|, |\beta| \rangle$

// note $\beta \in \mathbb{R}^p$
$X^T Y \in \mathbb{R}^p$
as $X^T \in \mathbb{R}^{p \times n}$
$Y \in \mathbb{R}^n$

$$= \left\langle \begin{vmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_p \end{vmatrix}, \begin{vmatrix} \beta_1 \\ \vdots \\ \beta_p \end{vmatrix} \right\rangle$$

$= |\beta_1 \alpha_1 + \beta_2 \alpha_2 + \cdots + \beta_p \alpha_p|$

$\leq |\beta_1 \alpha_j + \beta_2 \alpha_j + \cdots + \beta_p \alpha_j|$

where $\alpha_j = \max_j (x^T y)$

$= |\alpha_j (\beta_1 + \beta_2 + \cdots \beta_p)|$

$= |\alpha_j \Sigma_j \beta_j|$

$= \max_j |x^T y| \leq_j |B_j|$

$=$ RHS.

→ Hence LHS $\leq$ RHS

b) assume $\lambda \geq \max_j |x^T y|$