Homework 2

Author: Boqian Wang

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Question 1

Median sale prices data for Los Angeles County Housing in Aug 2013 from the Los Angeles Times were compiled into the file LAhousingpricesaug2013.txt. Let Y = sales of single family homes in August, X1 = median price of a single family residence (SFR) in thousands of dollars, X2 = median price of a condo in thousands of dollars, and X3 = median home price per square foot, in dollars. Each of these 4 vectors initially has length 269. If any row has an “n/a” in it for any of these 4 variables, then remove this entire row. Now each vector will have length 217. Please report your code of reading and cleaning the data.

In [ ]:
```
data = pd.read_csv('LAhousingpricesaug2013.txt',sep='\t')
data = data.dropna(subset=['SalesofSingleFamilyHomes','PriceMedianSFR(1000)','Pr
```

In [ ]:
```
Y = np.array(data['SalesofSingleFamilyHomes'])
X1 = np.array(data['PriceMedianSFR(1000)'])
X2 = np.array(data['PriceMedianCondos(1000)'])
X3 = np.array(data['MedianHomePrice/Sq.Ft'])
assert(Y.shape==(217,))
assert(X1.shape==(217,))
assert(X2.shape==(217,))
assert(X3.shape==(217,))
```

Question 2

Perform regression (with intercept) of Y on X = {X1, X2, X3} to compute a vector of parameter estimates, β = (β0, β1, β2, β3), where β0 is the estimated intercept and for i = 1,2,3, βi is the slope corresponding to explanatory variable Xi. Please report βˆ. Note: you need to implement two methods from the lecture for linear regression: Vectorized Gauss-Jordan Elimination wtih pivoting, and Sweep Operator.

We need to apply $\beta = (X^T X)^{-1}(X^T Y)$ to get the estimated β. When calculating the inverse, we can apply Gauss-Jordan Elimination with pivoting and Sweep Operator.

In [ ]:
```
X = np.concatenate((X1.reshape(-1, 1), X2.reshape(-1, 1), X3.reshape(-1, 1)), ax
ones = np.ones((X.shape[0], 1), dtype=int)
X = np.hstack((ones, X))
X_T = np.transpose(X)
product1 = np.dot(X_T,X) # product1 is the matrix that we need to apply Gauss-Jo
product2 = np.dot(X_T,Y) # X transpose times Y
```

In [ ]:
```
# Algorithm 1: Gauss-Jordan Elimination with pivoting
I = np.identity(4)
B = np.concatenate((product1,I),axis=1)
for i in range(4):
    pivot_row = i + np.argmax(np.abs(B[i:, i])) # Pivoting
    if pivot_row != i:
        B[[i, pivot_row]] = B[[pivot_row, i]]
    B[i] /= B[i][i]
    for j in range(4):
        if i != j:
            B[j] -= B[j][i] * B[i]
product1_1 = B[:,4:]
assert(product1_1.any() == np.linalg.inv(product1).any()) # Verify the correctne
```

In [ ]:
```
# Algorithm 2: Sweep operator
product1_2 = product1.copy()

for k in range(4):
    for i in range(4):
        for j in range(4):
            if i!=k and j!=k:
                product1_2[i][j] = product1_2[i][j] - product1_2[i][k]*product1_
    for i in range(4):
        if i!=k:
            product1_2[i][k] = product1_2[i][k]/product1_2[k][k]
    for j in range(4):
        if j!=k:
            product1_2[k][j] = product1_2[k][j]/product1_2[k][k]
    product1_2[k][k] = -1/product1_2[k][k]

product1_2 = -product1_2
assert(product1_2.any() == np.linalg.inv(product1).any()) # Verify the correctne
```

Now that we have used two different algorithms to calculate $(X^TX)^{-1}$, and β can therefore be calculated easily.

In [ ]:
```
product3 = product1_1 # X transpose times X.
product4 = np.dot(product3, product2)
beta0_hat = product4[0]
beta1_hat = product4[1]
beta2_hat = product4[2]
beta3_hat = product4[3]

print("So the beta hat is given by")
print("beta0_hat = ", beta0_hat)
print("beta1_hat = ", beta1_hat)
print("beta2_hat = ", beta2_hat)
print("beta3_hat = ", beta3_hat)
```

```
So the beta hat is given by
beta0_hat =  32.78844026750748
beta1_hat =  0.007823430546415142
beta2_hat =  0.0019521958359425828
beta3_hat =  -0.04255347469932469
```

Question 3

Let i = 1 Perform regression with intercept of Y on X with row i removed from the dataset. Let (−i) denote your resulting vector of parameter estimates, so that $\beta^{(-i)}$ is your estimate of the slope with ith row dropped. Please record $\beta^{(-i)}$.

In [ ]:
```
X_new = X.copy()
Y_new = Y.copy()
X_new = np.delete(X_new,0,axis=0)
Y_new = np.delete(Y_new,0)
```

Where X_new and Y_new are the new matrix and vector for us to play with. To start with, we write a function to calculate the final matrix to make our life easier (as there are loops in Question 4).

In [ ]:
```
def MatrixProcessing(X_new,Y_new):
    X_new_T = np.transpose(X_new)
    product1 = np.dot(X_new_T,X_new)
    product2 = np.dot(X_new_T,Y_new)
    for k in range(4):
        for i in range(4):
            for j in range(4):
                if i!=k and j!=k:
                    product1[i][j] = product1[i][j] - product1[i][k]*product1[k]
        for i in range(4):
            if i!=k:
                product1[i][k] = product1[i][k]/product1[k][k]
        for j in range(4):
            if j!=k:
                product1[k][j] = product1[k][j]/product1[k][k]
        product1[k][k] = -1/product1[k][k]
    product1 = -product1
    product4 = np.dot(product1,product2)
    return product4
```

In [ ]:
```
print("So the beta -1 is given by")
np.set_printoptions(suppress=True)
print(MatrixProcessing(X_new,Y_new))
```

```
So the beta -1 is given by
[33.35487414  0.00851346  0.00169906 -0.0446435 ]
```

Question 4

Repeat step 3 for i = 2, 3, $\cdots$, 217 and record $\beta^{(-i)}$

In [ ]:
```
beta = np.array([])
for i in range(1,217):
    X_temp = X.copy()
    Y_temp = Y.copy()
    X_temp = np.delete(X_temp,i,axis=0)
    Y_temp = np.delete(Y_temp,i)
    beta = np.append(beta,MatrixProcessing(X_temp,Y_temp))
beta = beta.reshape((216,4))
print(beta[:5]) # First five rows
```

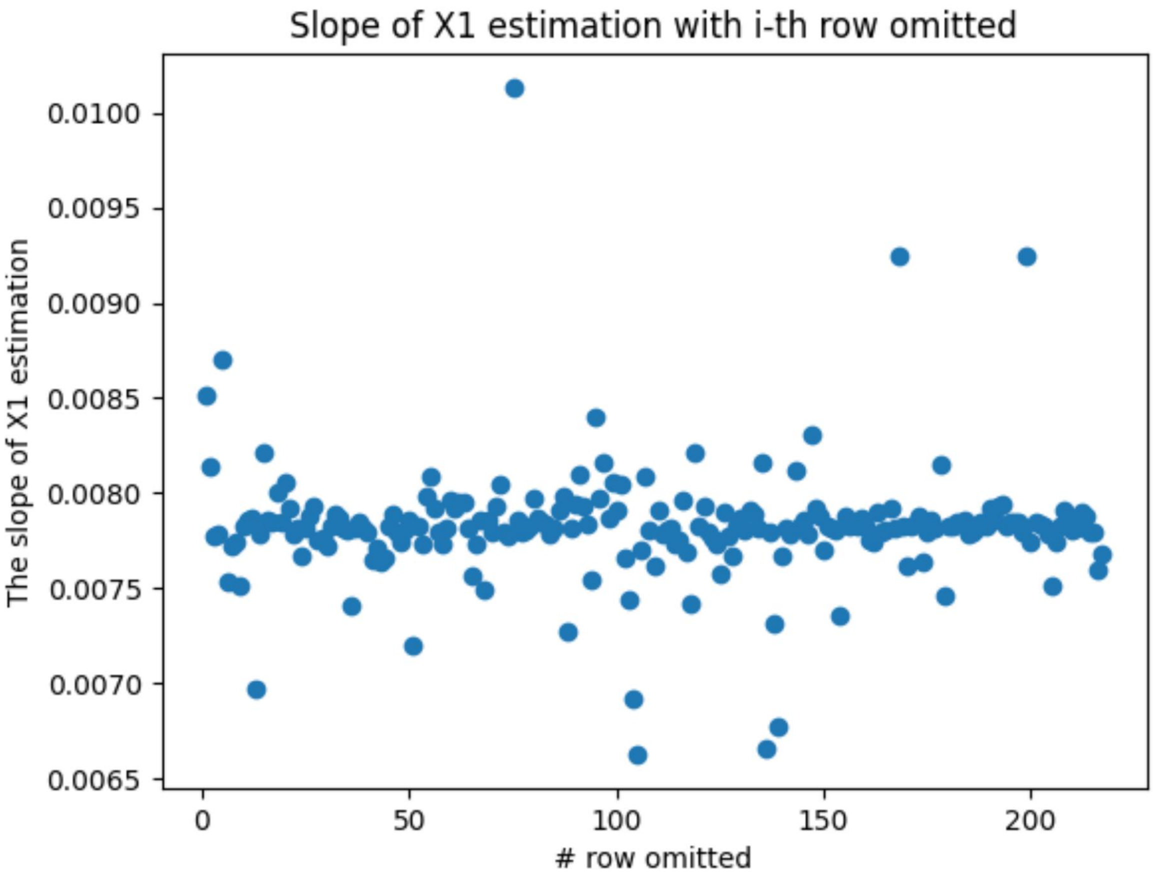
```
[[32.93716819  0.00813582  0.00226625 -0.04366388]
 [32.79049957  0.00777242  0.00192741 -0.04239459]
 [32.80722775  0.00778487  0.0019321  -0.04246719]
 [33.11309556  0.00870048 -0.0026299  -0.04044993]
 [32.83634459  0.00753104  0.00100914 -0.04148516]]
```

In this way, beta contains all of the $\beta^{(-i)}$ that we need.

Question 5

Plot the influences of $\beta_1^{(-i)}$, versus i. That is, the x-axis will span from i = 1 to 217, and the y-axis will be $\beta_1^{(-1)}, \ldots, \beta_1^{(-217)}$ which indicates the influence of observation i on the estimated slope. Please briefly describe your observation.

In [ ]:
```
data = np.array([])
data = np.append(data,MatrixProcessing(X_new,Y_new)[1])
for i in range(0,216):
    data = np.append(data,beta[i][1])
x = range(1,218)
plt.scatter(x=x, y=data)
plt.title('Slope of X1 estimation with i-th row omitted')
plt.xlabel('# row omitted')
plt.ylabel('The slope of X1 estimation')
plt.show()
```



Comment: The slope of X_1 estimation β_1 is around 0.008 if an arbitary row is omitted. However, there exists some outliers that the estimated slope approaches high to 0.0100 or low to 0.0065. In this way, we can conclude that these observations have significant contributions to the estimation of slope of X_1.