Regression model

Relative location of CT slices on axial axis

The data are available at:

https://archive.ics.uci.edu/dataset/206/relative+location+of+ct+slices+on+axial+axis

The dataset consists of 384 features extracted from CT images. The class variable is numeric and denotes the relative location of the CT slice on the axial axis of the human body.

The data was retrieved from a set of 53500 CT images from 74 different patients (43 male, 31 female).

To exstract the data use the panda routines

```
import zipfile
import pandas as pd

# read the dataset using the compression zip
df = 
pd.read_csv('https://archive.ics.uci.edu/static/public/206/relative+lo
cation+of+ct+slices+on+axial+axis.zip',compression='zip')

# display dataset
print(df.head())
```

```
   patientId  value0  value1  value2  value3  value4  value5  value6
value7  \
0          0     0.0     0.0     0.0     0.0     0.0     0.0   -0.25
-0.25
1          0     0.0     0.0     0.0     0.0     0.0     0.0   -0.25
-0.25
2          0     0.0     0.0     0.0     0.0     0.0     0.0   -0.25
-0.25
3          0     0.0     0.0     0.0     0.0     0.0     0.0   -0.25
-0.25
4          0     0.0     0.0     0.0     0.0     0.0     0.0   -0.25
-0.25

   value8  ...  value375  value376  value377  value378  value379
value380  \
0   -0.25  ...     -0.25  0.980381       0.0       0.0       0.0
0.0
1   -0.25  ...     -0.25  0.977008       0.0       0.0       0.0
0.0
```

```
2   -0.25  ...     -0.25  0.977008       0.0       0.0       0.0
0.0
3   -0.25  ...     -0.25  0.977008       0.0       0.0       0.0
0.0
4   -0.25  ...     -0.25  0.976833       0.0       0.0       0.0
0.0

   value381  value382  value383  reference
0       0.0     -0.25     -0.25  21.803851
1       0.0     -0.25     -0.25  21.745726
2       0.0     -0.25     -0.25  21.687600
3       0.0     -0.25     -0.25  21.629474
4       0.0     -0.25     -0.25  21.571348

[5 rows x 386 columns]
```

We transform the data to a matrix of shape 53500 x 386

```
Aall=df.to_numpy()
print(Aall.shape)

(53500, 386)
```

We add a column of all 1 and we organize the input data by dividing in test set and training set

```
from sklearn.model_selection import train_test_split
import numpy as np
#Add a column of ones at the beginning of the data matrix
Aall = np.column_stack([np.ones(Aall.shape[0]), Aall])
X = Aall
X=np.delete(X,386,1)
y = Aall[:,386]
```

Use the prepared data to solve the regression model with all the studied techniques. Can we use the normal equation and the QR factorization? If the answer is positive compare the condition numbers of the QR methods and the normal equations. What are the results?

Use the funcation scipy.linalg.lstsq and check if all the lapack drivers works. Compare the results changing the initial value cond. The results are the same? What about the execution time?

Analyze the singular values and check if it is possible to use a principal component regression procedure. Compute the solution using the singular value decomposition. Can you observe a relation in the chosen singular value and the value of cond of the routine lstsq?

Perform the same analysis by preprocessing the data in order to have data from a normal distribution with mean zero and compute the singular value decomposition on this matrix.

Check the performance of the method by computing the least square residual for the training set and the testset. The minimum and the maximum values of the predicted error for both, the training set and the testset.

Compute the multiple R-squared: R2_train = 1 - sum((y - yest)**2)/sum((y-mean(y))**2 where y are the value to predict and yest are the estimated values for the training set. Compute the value R2_test for the testset.

A value of R2 near one means that the constructed model is good.

Change the size of the training set and the testing set to 0.7% and 0.3% and repeat the previous steps.

Comment the obtained results.

Normal equation is a method to solve the linear regression problem. It is based on the following formula:

$$\theta = \left(X^T X\right)^{-1} X^T y$$

where θ is the vector of parameters, X is the matrix of input data, and y is the vector of output data. It is important to note that the matrix $X^T X$ must be full rank, otherwise the inverse of $X^T X$ does not exist, so the normal equation cannot be used.

```
def normalEquations(X_train, y_train, X_test, y_test):
    print("Rank of train data:
",np.linalg.matrix_rank(X_train.T@X_train))
    print("Shape of train data: ",(X_train.T @ X_train).shape)
    if np.linalg.matrix_rank(X_train.T@X_train)==X_train.shape[1]:
        theta=np.linalg.solve(X_train.T@X_train,X_train.T@y_train)
        y_train_pred=X_train@theta
        y_train_pred = X_train @ theta
        residuals_train = y_train - y_train_pred
        print("Residuals for train set:
",np.linalg.norm(residuals_train,2))
        print("Maximum error for train set:
",np.max(np.abs(residuals_train)))
        print("Minimum error for train set:
",np.min(np.abs(residuals_train)))
        R2_train  = 1 - np.sum((y_train -
y_train_pred)**2)/np.sum((y_train - np.mean(y_train))**2)
        print("R2 for train set: ",R2_train)

        y_test_pred = X_test @ theta
        residuals_test = y_test - y_test_pred
        print("Residuals for test set:
",np.linalg.norm(residuals_test,2))
        print("Maximum error for test set:
",np.max(np.abs(residuals_test)))
        print("Minimum error for test set:
",np.min(np.abs(residuals_test)))
        R2_test  = 1 - np.sum((y_test -
y_test_pred)**2)/np.sum((y_test - np.mean(y_test))**2)
        print("R2 for test set: ",R2_test)
```

```python
    else:
        print("Matrix is singular")
```

The QR factorization can be used to solve the linear regression problem. It is based on the following formula:

$$X = QR$$

where X is the matrix of input data, Q is an orthogonal matrix, and R is an upper triangular matrix. The solution of the linear regression problem is given by:

$$\theta = R^{-1} Q^T y$$

When X is not full rank, we cannot use the QR factorization to solve the problem. We need to use the Pivot QR factorization, which is based on the following formula:

$$X P = \begin{pmatrix} Q_X & Q_x^- \end{pmatrix} \begin{pmatrix} R_1 & R_2 \\ 0 & 0 \end{pmatrix}$$

where P is a permutation matrix. The solution of the linear regression problem is given by:

$$\hat{\theta} = P \begin{pmatrix} R_1^{-1} Q_X^T y \\ 0 \end{pmatrix}.$$

```python
import scipy.linalg as la
def QRsolver(X_train, y_train, X_test, y_test):
    rank=np.linalg.matrix_rank(X_train)
    #if rank not maximum
    if rank<X_train.shape[1]:
        Q,R,P = la.qr(X_train, pivoting=True, mode='economic')
#economic -> Q is m x k, R is k x n where k=min(m,n)
        print("Condition number of QR factorization matrix:
",np.linalg.cond(R,2))
        #truncate R and Q to rank
        R_trunc = R[:rank, :rank]
        Q_trunc = Q[:, :rank]
        QTb = Q_trunc.T @ y_train
        theta_permuted=np.zeros(X_train.shape[1])
        theta_permuted [:rank]= la.solve_triangular(R_trunc, QTb)
#store the solution in the first rank columns
        #restore original order
        theta = np.zeros_like(theta_permuted)
        theta[P] = theta_permuted #permute the elements of
theta_permuted to get the solution
    else:
        Q,R=la.qr(X_train, mode='economic')
        R_trunc = R[:rank, :rank]
        Q_trunc = Q[:, :rank]
        QTb = Q_trunc.T @ y_train
```

```
        theta = la.solve_triangular(R_trunc, QTb)
    y_train_pred = X_train @ theta
    residuals_train = y_train - y_train_pred
    print("Residuals for train set: ",np.linalg.norm(residuals_train,2))
    print("Maximum error for train set: ",np.max(np.abs(residuals_train)))
    print("Minimum error for train set: ",np.min(np.abs(residuals_train)))
    R2_train  = 1 - np.sum((y_train - y_train_pred)**2)/np.sum((y_train - np.mean(y_train))**2)
    print("R2 for train set: ",R2_train)

    y_test_pred = X_test @ theta
    residuals_test = y_test - y_test_pred
    print("Residuals for test set: ",np.linalg.norm(residuals_test,2))
    print("Maximum error for test set: ",np.max(np.abs(residuals_test)))
    print("Minimum error for test set: ",np.min(np.abs(residuals_test)))
    R2_test  = 1 - np.sum((y_test - y_test_pred)**2)/np.sum((y_test - np.mean(y_test))**2)
    print("R2 for test set: ",R2_test)
```

The Singular Value Decomposition (SVD) can also be used to solve the linear regression problem. It is based on the following formula:

$$X = U \Sigma V^T$$

where X is the matrix of input data, U is an orthogonal matrix, Σ is a rectangular, and V is an orthogonal matrix. The solution of the linear regression problem is given by:

$$theta = V_r D^{-1} U_r^T y$$

where V_r is the matrix containing the first r columns of V, U_r is the matrix containing the first r columns of U, and D is the diagonal matrix containing the first r singular values, which are the non-zeros one

```
def SVDSolver(X_train, y_train, X_test, y_test):
    rank = np.linalg.matrix_rank(X_train)
    U, S, Vt = np.linalg.svd(X_train, full_matrices=False) #U and V are of size m x k and n x k, k=min(m,n) when full_matrices=False
    print("Rank of X_train: ",rank)
    print("Number of non zero singular values: ",np.count_nonzero(S))
    #S contains all singular values. If we consider only the non zero singular values, they should be equal to the rank of X_train. In this case this is not true, probably due to numerical errors.
    # I will use later the PCR and Euckardt-Young theorem to remove noise
```

```
    # So let's consider the first r=rank(X_train) singular values
    U_r = U[:, :rank]
    Vt_r = Vt[:rank, :]
    S_r = np.diag(S)[:rank, :rank]
    S_r_inv = np.linalg.inv(S_r)
    theta = Vt_r.T @ S_r_inv @ U_r.T @ y_train
    y_train_pred = X_train @ theta
    residuals_train = y_train - y_train_pred
    print("Residuals for train set:
",np.linalg.norm(residuals_train,2))
    print("Maximum error for train set:
",np.max(np.abs(residuals_train)))
    print("Minimum error for train set:
",np.min(np.abs(residuals_train)))
    R2_train  = 1 - np.sum((y_train -
y_train_pred)**2)/np.sum((y_train - np.mean(y_train))**2)
    print("R2 for train set: ",R2_train)

    y_test_pred = X_test @ theta
    residuals_test = y_test - y_test_pred
    print("Residuals for test set: ",np.linalg.norm(residuals_test,2))
    print("Maximum error for test set:
",np.max(np.abs(residuals_test)))
    print("Minimum error for test set:
",np.min(np.abs(residuals_test)))
    R2_test  = 1 - np.sum((y_test - y_test_pred)**2)/np.sum((y_test -
np.mean(y_test))**2)
    print("R2 for test set: ",R2_test)
```

The function scipy.linalg.lstsq is a method to solve the least squares problem. It uses different LAPACK drivers to solve the problem. The available drivers are:

1. **`gelsd`**:
   - **Description**: The `gelsd` driver solves the **least squares problem** using **Singular Value Decomposition (SVD)** and Divide-and-Conquer method.
2. **`gelsy`**:
   - **Description**: The `gelsy` driver solves the **least squares problem** using **QR decomposition with column pivoting**.
3. **`gelss`**:
   - **Description**:The `gelss` driver solves the **least squares problem** using **Singular Value Decomposition (SVD)**.

Divide and conquer is an optimization technique used to compute the **Singular Value Decomposition (SVD)** more efficiently.

- **Concept**: Divide and conquer breaks down the SVD problem into smaller, more manageable subproblems. Instead of performing the full SVD computation at once, the algorithm recursively divides the matrix into smaller parts and computes the decomposition in stages, merging results as it progresses.

```
import time
def lstsqSolver(X_train, y_train, X_test, y_test):
    conditions=[1e-1,1e-2,1e-4,1e-8,1e-12,1e-16]
    values={}
    for condition in conditions:
        #gelsd
        times=[]
        for i in range(5):
            start_time = time.time()
            theta_gelsd = la.lstsq(X_train,
y_train,lapack_driver='gelsd',cond=condition)[0] #we took [0] because
lstsq returns a tuple
            times.append(time.time()-start_time)

        values['driver']='gelsd'
        values['condition']=condition
        values['AVGtime']=np.mean(times)
        y_train_pred = X_train @ theta_gelsd
        residuals_train = y_train - y_train_pred
        values['residuals_train']=np.linalg.norm(residuals_train,2)
        values['Min error
residuals_train']=np.min(np.abs(residuals_train))
        values['Max error
residuals_train']=np.max(np.abs(residuals_train))
        R2_train  = 1 - np.sum((y_train -
y_train_pred)**2)/np.sum((y_train - np.mean(y_train))**2)
        values['R2_train']=R2_train
        y_test_pred = X_test @ theta_gelsd
        residuals_test = y_test - y_test_pred
        values['residuals_test']=np.linalg.norm(residuals_test,2)
        values['Min error
residuals_test']=np.min(np.abs(residuals_test))
        values['Max error
residuals_test']=np.max(np.abs(residuals_test))
        R2_test  = 1 - np.sum((y_test -
y_test_pred)**2)/np.sum((y_test - np.mean(y_test))**2)
        values['R2_test']=R2_test
        print(values)
        #gelss
        times=[]
        for i in range(5):
            start_time = time.time()
            theta_gelss = la.lstsq(X_train,
y_train,lapack_driver='gelss',cond=condition)[0] #we took [0] because
lstsq returns a tuple
            times.append(time.time()-start_time)
        values['driver']='gelss'
```

```
        values['condition']=condition
        values['AVGtime']=np.mean(times)
        y_train_pred = X_train @ theta_gelss
        residuals_train = y_train - y_train_pred
        values['residuals_train']=np.linalg.norm(residuals_train,2)
        values['Min error
residuals_train']=np.min(np.abs(residuals_train))
        values['Max error
residuals_train']=np.max(np.abs(residuals_train))
        R2_train  = 1 - np.sum((y_train -
y_train_pred)**2)/np.sum((y_train - np.mean(y_train))**2)
        values['R2_train']=R2_train
        y_test_pred = X_test @ theta_gelss
        residuals_test = y_test - y_test_pred
        values['residuals_test']=np.linalg.norm(residuals_test,2)
        values['Min error
residuals_test']=np.min(np.abs(residuals_test))
        values['Max error
residuals_test']=np.max(np.abs(residuals_test))
        R2_test  = 1 - np.sum((y_test -
y_test_pred)**2)/np.sum((y_test - np.mean(y_test))**2)
        values['R2_test']=R2_test
        print(values)
        #gelsy
        times=[]
        for i in range(5):
            start_time = time.time()
            theta_gelsy = la.lstsq(X_train,
y_train,lapack_driver='gelsy',cond=condition)[0] #we took [0] because
lstsq returns a tuple
            times.append(time.time()-start_time)
        values['driver']='gelsy'
        values['condition']=condition
        values['AVGtime']=np.mean(times)
        y_train_pred = X_train @ theta_gelsy
        residuals_train = y_train - y_train_pred
        values['residuals_train']=np.linalg.norm(residuals_train,2)
        values['Min error
residuals_train']=np.min(np.abs(residuals_train))
        values['Max error
residuals_train']=np.max(np.abs(residuals_train))
        R2_train  = 1 - np.sum((y_train -
y_train_pred)**2)/np.sum((y_train - np.mean(y_train))**2)
        values['R2_train']=R2_train
        y_test_pred = X_test @ theta_gelsy
        residuals_test = y_test - y_test_pred
        values['residuals_test']=np.linalg.norm(residuals_test,2)
        values['Min error
residuals_test']=np.min(np.abs(residuals_test))
```

```
        values['Max error residuals_test']=np.max(np.abs(residuals_test))
        R2_test  = 1 - np.sum((y_test - y_test_pred)**2)/np.sum((y_test - np.mean(y_test))**2)
        values['R2_test']=R2_test
        print(values)
```

PCR can be used to solve the linear regression problem. It is based on the following formula:

$$X_k = U_k \Sigma_k V_k^T$$

where X_k is an approximation of the matrix of input data, U_k contains the first k columns of U, Σ_k contains the first k singular values, and V_k contains the first k columns of V.

The problem is to find the best value of k. There are several methods that we can use like:

1. **Mixed Error**: We can use the mixed error to find the best value of k. The criterion is based on the following formula:

$$\frac{\| X - X_k \|_2}{\| X \|_2 + 1} = \frac{\sigma_{k+1}}{\sigma_1 + 1} \leq \text{tol}$$

When we find k+1 so that the criterion is satisfied, we can use k as the best value of k.

1. **Scree Plot**: We can use the scree plot to find the best value of k. The scree plot shows the singular values in decreasing order. We can find the best value of k by looking at the point where the curve starts to flatten.

2. **Cumulative Percentage of Variance**: We can use the cumulative percentage of variance to find the best value of k. The cumulative percentage of variance is given by:

$$\frac{\sum_{i=1}^{k} \sigma_i^2}{\sum_{i=1}^{r} \sigma_i^2} > ¿ \text{p}$$

where p is the percentage of variance that we want to explain between 0 and 1. When we find k so that the criterion is satisfied, we can use k as the best value of k.

```python
import matplotlib.pyplot as plt
from kneed import KneeLocator

def PCRSolver(X_train, y_train, X_test, y_test):
    U, S, Vt = np.linalg.svd(X_train, full_matrices=False)
    #1: mixed error criterion
    tol=1e-6
    k=np.argmax(S/(S[0]+1)<tol)-1  #argmax returns the first occurrence of the that satisfies the condition and this is k+1
```

```
    print("k=",k," with mixed error criterion and tol=",tol)
    print("Sigma_k=",S[k])
    U_k = U[:, :k]
    Vt_k = Vt[:k, :]
    S_k = np.diag(S)[:k, :k]
    S_k_inv = np.linalg.inv(S_k)
    theta = Vt_k.T @ S_k_inv @ U_k.T @ y_train
    y_train_pred = X_train @ theta
    residuals_train = y_train - y_train_pred
    print("Residuals for train set:
",np.linalg.norm(residuals_train,2))
    print("Maximum error for train set:
",np.max(np.abs(residuals_train)))
    print("Minimum error for train set:
",np.min(np.abs(residuals_train)))
    R2_train  = 1 - np.sum((y_train -
y_train_pred)**2)/np.sum((y_train - np.mean(y_train))**2)
    print("R2 for train set: ",R2_train)

    y_test_pred = X_test @ theta
    residuals_test = y_test - y_test_pred
    print("Residuals for test set: ",np.linalg.norm(residuals_test,2))
    print("Maximum error for test set:
",np.max(np.abs(residuals_test)))
    print("Minimum error for test set:
",np.min(np.abs(residuals_test)))
    R2_test  = 1 - np.sum((y_test - y_test_pred)**2)/np.sum((y_test -
np.mean(y_test))**2)
    print("R2 for test set: ",R2_test)






    #2: Scree Plot

    k1 = KneeLocator(range(len(S)), S, curve='convex',
direction='decreasing')
    plt.plot(range(len(S)), S)
    plt.scatter(k1.elbow, S[k1.elbow], c='red', s=100, alpha=0.5)
    plt.xlabel('Singular value index')
    plt.ylabel('Singular value')
    plt.title('Scree plot')
    plt.legend(['Singular values', 'Elbow'])
    plt.show()
    k=k1.elbow
    print("k=",k," with Screep plot")
    print("Sigma_k=",S[k])
```

```
    U_k = U[:, :k]
    Vt_k = Vt[:k, :]
    S_k = np.diag(S)[:k, :k]
    S_k_inv = np.linalg.inv(S_k)
    theta = Vt_k.T @ S_k_inv @ U_k.T @ y_train
    y_train_pred = X_train @ theta
    residuals_train = y_train - y_train_pred
    print("Residuals for train set:
",np.linalg.norm(residuals_train,2))
    print("Maximum error for train set:
",np.max(np.abs(residuals_train)))
    print("Minimum error for train set:
",np.min(np.abs(residuals_train)))
    R2_train  = 1 - np.sum((y_train -
y_train_pred)**2)/np.sum((y_train - np.mean(y_train))**2)
    print("R2 for train set: ",R2_train)

    y_test_pred = X_test @ theta
    residuals_test = y_test - y_test_pred
    print("Residuals for test set: ",np.linalg.norm(residuals_test,2))
    print("Maximum error for test set:
",np.max(np.abs(residuals_test)))
    print("Minimum error for test set:
",np.min(np.abs(residuals_test)))
    R2_test  = 1 - np.sum((y_test - y_test_pred)**2)/np.sum((y_test -
np.mean(y_test))**2)
    print("R2 for test set: ",R2_test)


    #3: Cumulative percentage of variance
    p=0.99
    total_sum=np.sum(S**2)
    cumulative_sum=np.cumsum(S**2)
    k=np.argmax((cumulative_sum/total_sum)>p)
    print("k=",k,"with cumulative percentage of variance and p=",p)
    print("Sigma_k=",S[k])
    U_k = U[:, :k]
    Vt_k = Vt[:k, :]
    S_k = np.diag(S)[:k, :k]
    S_k_inv = np.linalg.inv(S_k)
    theta = Vt_k.T @ S_k_inv @ U_k.T @ y_train
    y_train_pred = X_train @ theta
    residuals_train = y_train - y_train_pred
    print("Residuals for train set:
",np.linalg.norm(residuals_train,2))
    print("Maximum error for train set:
",np.max(np.abs(residuals_train)))
    print("Minimum error for train set:
",np.min(np.abs(residuals_train)))
    R2_train  = 1 - np.sum((y_train -
```

```
y_train_pred)**2)/np.sum((y_train - np.mean(y_train))**2)
    print("R2 for train set: ",R2_train)

    y_test_pred = X_test @ theta
    residuals_test = y_test - y_test_pred
    print("Residuals for test set: ",np.linalg.norm(residuals_test,2))
    print("Maximum error for test set: 
",np.max(np.abs(residuals_test)))
    print("Minimum error for test set: 
",np.min(np.abs(residuals_test)))
    R2_test  = 1 - np.sum((y_test - y_test_pred)**2)/np.sum((y_test - 
np.mean(y_test))**2)
    print("R2 for test set: ",R2_test)


import numpy as np
from sklearn.model_selection import train_test_split


X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    train_size = .9,
    test_size = .1,
    random_state = 5,
    shuffle = True
)
print("\nNORMAL EQUATIONS")
normalEquations(X_train, y_train, X_test, y_test)
print("\nQR SOLVER")
QRsolver(X_train, y_train, X_test, y_test)
print("\nSVD SOLVER")
SVDSolver(X_train, y_train, X_test, y_test)
print("\nLSTSQ SOLVER")
lstsqSolver(X_train, y_train, X_test, y_test)
print("\nPCR SOLVER")
PCRSolver(X_train, y_train, X_test, y_test)
```

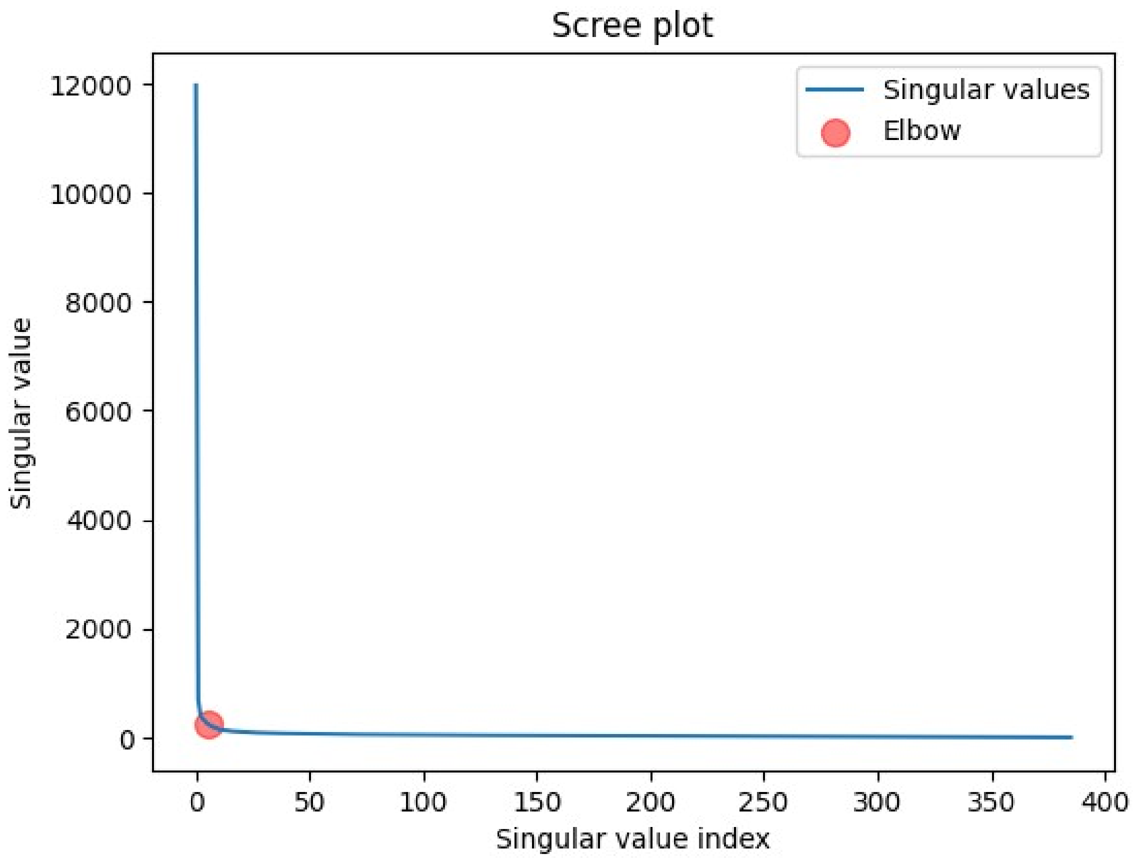
```
NORMAL EQUATIONS
Rank of train data:  375
Shape of train data:  (386, 386)
Matrix is singular

QR SOLVER
Condition number of QR factorization matrix:  2.4463550697566203e+32
Residuals for train set:  1798.3737270942406
Maximum error for train set:  49.47129163028276
Minimum error for train set:  1.4352963262354024e-12
R2 for train set:  0.8653831545119198
```

```
Residuals for test set:  613.1967475658972
Maximum error for test set:  46.98621458423358
Minimum error for test set:  0.0023313738117138882
R2 for test set:  0.8603214800189586

SVD SOLVER
Rank of X_train:  375
Number of non zero singular values:  386
Residuals for train set:  1798.3737270942406
Maximum error for train set:  49.47129163028183
Minimum error for train set:  5.613287612504791e-13
R2 for train set:  0.8653831545119198
Residuals for test set:  613.1967475658979
Maximum error for test set:  46.98621458423264
Minimum error for test set:  0.0023313738110175564
R2 for test set:  0.8603214800189583

LSTSQ SOLVER
{'driver': 'gelsd', 'condition': 0.1, 'AVGtime': 1.778600263595581,
'residuals_train': 7079.003325567669, 'Min error residuals_train':
0.00016317329873416497, 'Max error residuals_train':
86.12879856814983, 'R2_train': -1.085853217608597, 'residuals_test':
2361.142356336503, 'Min error residuals_test': 0.0015384355100849234,
'Max error residuals_test': 86.21332456608175, 'R2_test': -
1.0709722681710114}
{'driver': 'gelss', 'condition': 0.1, 'AVGtime': 1.7356050491333008,
'residuals_train': 7079.00332556767, 'Min error residuals_train':
0.00016317329874482311, 'Max error residuals_train':
86.12879856814982, 'R2_train': -1.085853217608597, 'residuals_test':
2361.142356336503, 'Min error residuals_test': 0.0015384355101133451,
'Max error residuals_test': 86.21332456608174, 'R2_test': -
1.0709722681710114}
{'driver': 'gelsy', 'condition': 0.1, 'AVGtime': 2.619104099273682,
'residuals_train': 7079.030051187763, 'Min error residuals_train':
0.0011036817574421 32, 'Max error residuals_train': 86.13955960940007,
'R2_train': -1.0858689672341675, 'residuals_test': 2361.119587258657,
'Min error residuals_test': 0.0034863125009110263, 'Max error
residuals_test': 86.22408570665593, 'R2_test': -1.070932326571298}
{'driver': 'gelsd', 'condition': 0.01, 'AVGtime': 1.564098834991455,
'residuals_train': 2852.209177373248, 'Min error residuals_train':
0.00018338035124543 24, 'Max error residuals_train': 59.94903873041823,
'R2_train': 0.6613880686470945, 'residuals_test': 947.1567219811014,
'Min error residuals_test': 0.002341357849196868, 'Max error
residuals_test': 57.95973924838983, 'R2_test': 0.6667473351505734}
{'driver': 'gelss', 'condition': 0.01, 'AVGtime': 1.6477362155914306,
'residuals_train': 2852.2091773732373, 'Min error residuals_train':
0.00018338034929143987, 'Max error residuals_train': 59.9490387304176,
'R2_train': 0.6613880686470971, 'residuals_test': 947.1567219810992,
'Min error residuals_test': 0.002341357847889469, 'Max error
```

```
residuals_test': 57.959739248389454, 'R2_test': 0.6667473351505749}
{'driver': 'gelsy', 'condition': 0.01, 'AVGtime': 2.6249019622802736,
'residuals_train': 7079.030051187763, 'Min error residuals_train':
0.0011036817574421 32, 'Max error residuals_train': 86.13955960940007,
'R2_train': -1.0858689672341675, 'residuals_test': 2361.119587258657,
'Min error residuals_test': 0.003486312500911026 3, 'Max error
residuals_test': 86.22408570665593, 'R2_test': -1.070932326571298}
{'driver': 'gelsd', 'condition': 0.0001, 'AVGtime':
1.5610857486724854, 'residuals_train': 1798.3994734141352, 'Min error
residuals_train': 1.8338172878884507e-05, 'Max error residuals_train':
49.47573432749514, 'R2_train': 0.8653793000147989, 'residuals_test':
613.1715704956185, 'Min error residuals_test': 0.0014430106109344365,
'Max error residuals_test': 46.9891455047549, 'R2_test':
0.8603329498244735}
{'driver': 'gelss', 'condition': 0.0001, 'AVGtime':
1.6475080013275147, 'residuals_train': 1798.3994734141352, 'Min error
residuals_train': 1.8338172111498352e-05, 'Max error residuals_train':
49.475734327497754, 'R2_train': 0.8653793000147989, 'residuals_test':
613.1715704956163, 'Min error residuals_test': 0.0014430106120997266,
'Max error residuals_test': 46.98914550475641, 'R2_test':
0.8603329498244745}
{'driver': 'gelsy', 'condition': 0.0001, 'AVGtime':
2.6496541023254396, 'residuals_train': 1798.3882302496897, 'Min error
residuals_train': 8.989160991035305e-05, 'Max error residuals_train':
49.47471047246335, 'R2_train': 0.8653809832425342, 'residuals_test':
613.1948421789593, 'Min error residuals_test': 0.0006585745238822938,
'Max error residuals_test': 46.98724460469762, 'R2_test':
0.860322348064052}
{'driver': 'gelsd', 'condition': 1e-08, 'AVGtime': 1.5690276622772217,
'residuals_train': 1798.3737270942404, 'Min error residuals_train':
1.8260948309034575e-12, 'Max error residuals_train':
49.47129163027922, 'R2_train': 0.8653831545119198, 'residuals_test':
613.1967475659, 'Min error residuals_test': 0.002331373807535897, 'Max
error residuals_test': 46.986214584231135, 'R2_test':
0.8603214800189574}
{'driver': 'gelss', 'condition': 1e-08, 'AVGtime': 1.633497953414917,
'residuals_train': 1798.3737270942404, 'Min error residuals_train':
4.050093593832571e-13, 'Max error residuals_train':
49.471291630281804, 'R2_train': 0.8653831545119198, 'residuals_test':
613.1967475658978, 'Min error residuals_test': 0.002331373811045978,
'Max error residuals_test': 46.98621458423261, 'R2_test':
0.8603214800189584}
{'driver': 'gelsy', 'condition': 1e-08, 'AVGtime': 2.636412000656128,
'residuals_train': 1798.3737270942406, 'Min error residuals_train':
1.3287149158713873e-12, 'Max error residuals_train':
49.471291630282785, 'R2_train': 0.8653831545119198, 'residuals_test':
613.1967475658972, 'Min error residuals_test': 0.002331373811642834,
'Max error residuals_test': 46.986214584233636, 'R2_test':
0.8603214800189586}
```

```
{'driver': 'gelsd', 'condition': 1e-12, 'AVGtime': 1.5784571647644043,
'residuals_train': 1798.3737270942404, 'Min error residuals_train':
1.8260948309034575e-12, 'Max error residuals_train':
49.47129163027922, 'R2_train': 0.8653831545119198, 'residuals_test':
613.1967475659, 'Min error residuals_test': 0.002331373807535897, 'Max
error residuals_test': 46.986214584231135, 'R2_test':
0.8603214800189574}
{'driver': 'gelss', 'condition': 1e-12, 'AVGtime': 1.6809359550476075,
'residuals_train': 1798.3737270942404, 'Min error residuals_train':
4.050093593832571e-13, 'Max error residuals_train':
49.471291630281804, 'R2_train': 0.8653831545119198, 'residuals_test':
613.1967475658978, 'Min error residuals_test': 0.002331373811045978,
'Max error residuals_test': 46.98621458423261, 'R2_test':
0.8603214800189584}
{'driver': 'gelsy', 'condition': 1e-12, 'AVGtime': 2.6335572719573976,
'residuals_train': 1798.3737270942406, 'Min error residuals_train':
1.3287149158713873e-12, 'Max error residuals_train':
49.471291630282785, 'R2_train': 0.8653831545119198, 'residuals_test':
613.1967475658972, 'Min error residuals_test': 0.002331373811642834,
'Max error residuals_test': 46.986214584233636, 'R2_test':
0.8603214800189586}
{'driver': 'gelsd', 'condition': 1e-16, 'AVGtime': 1.6632410049438477,
'residuals_train': 1798.3737270942404, 'Min error residuals_train':
1.8260948309034575e-12, 'Max error residuals_train':
49.47129163027922, 'R2_train': 0.8653831545119198, 'residuals_test':
613.1967475659, 'Min error residuals_test': 0.002331373807535897, 'Max
error residuals_test': 46.986214584231135, 'R2_test':
0.8603214800189574}
{'driver': 'gelss', 'condition': 1e-16, 'AVGtime': 1.7481321334838866,
'residuals_train': 1798.3737270942404, 'Min error residuals_train':
4.050093593832571e-13, 'Max error residuals_train':
49.471291630281804, 'R2_train': 0.8653831545119198, 'residuals_test':
613.1967475658978, 'Min error residuals_test': 0.002331373811045978,
'Max error residuals_test': 46.98621458423261, 'R2_test':
0.8603214800189584}
{'driver': 'gelsy', 'condition': 1e-16, 'AVGtime': 2.6745798110961916,
'residuals_train': 1798.3737270942406, 'Min error residuals_train':
1.3287149158713873e-12, 'Max error residuals_train':
49.471291630282785, 'R2_train': 0.8653831545119198, 'residuals_test':
613.1967475658972, 'Min error residuals_test': 0.002331373811642834,
'Max error residuals_test': 46.986214584233636, 'R2_test':
0.8603214800189586}

PCR SOLVER
k= 374  with mixed error criterion and tol= 1e-06
Sigma_k= 0.24235413263899452
Residuals for train set:  1798.3737432334817
Maximum error for train set:  49.47135129405567
Minimum error for train set:  8.885552780668604e-06
```

```
R2 for train set:  0.8653831520957216
Residuals for test set:  613.1967323323433
Maximum error for test set:  46.9862610598538
Minimum error for test set:  0.0023566596409949625
R2 for test set:  0.8603214869589831
```




```
k= 5  with Screep plot
Sigma_k= 250.9209568227013
Residuals for train set:  4314.872172761188
Maximum error for train set:  81.8056017452396
Minimum error for train set:  4.238081472607291e-05
R2 for train set:  0.22504697175187627
Residuals for test set:  1430.3535335393976
Maximum error for test set:  72.85619426228541
Minimum error for test set:  0.004824564473430826
R2 for test set:  0.23999451660000515
k= 1 with cumulative percentage of variance and p= 0.99
Sigma_k= 688.74133812504
Residuals for train set:  7079.00332556767
Maximum error for train set:  86.12879856814982
Minimum error for train set:  0.00016317329874482311
R2 for train set:  -1.085853217608597
```

```
Residuals for test set:  2361.142356336503
Maximum error for test set:  86.21332456608174
Minimum error for test set:  0.0015384355101133451
R2 for test set:  -1.0709722681710114
```

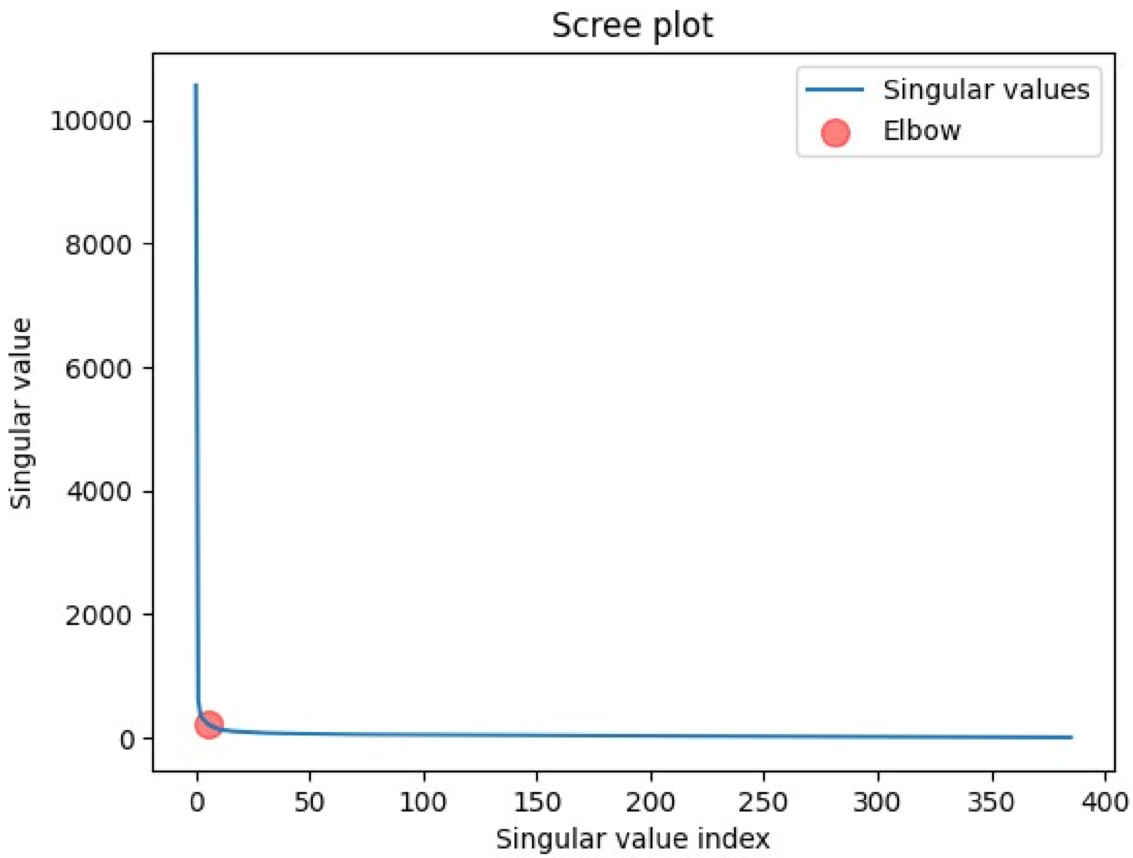
```python
X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    train_size = .7,
    test_size = .3,
    random_state = 5,
    shuffle = True
)

print("\nNORMAL EQUATIONS")
normalEquations(X_train, y_train, X_test, y_test)
print("\nQR SOLVER")
QRsolver(X_train, y_train, X_test, y_test)
print("\nSVD SOLVER")
SVDSolver(X_train, y_train, X_test, y_test)
print("\nLSTSQ SOLVER")
lstsqSolver(X_train, y_train, X_test, y_test)
print("\nPCR SOLVER")
PCRSolver(X_train, y_train, X_test, y_test)
```

```
NORMAL EQUATIONS
Rank of train data:  374
Shape of train data:  (386, 386)
Matrix is singular

QR SOLVER
Condition number of QR factorization matrix:  2.4045944568960685e+32
Residuals for train set:  1581.8224079089453
Maximum error for train set:  49.31879889962844
Minimum error for train set:  3.197442310920451e-14
R2 for train set:  0.8663961504183206
Residuals for test set:  1056.7495138518016
Maximum error for test set:  46.86165824234786
Minimum error for test set:  0.0005819565929527926
R2 for test set:  0.8602025218438263

SVD SOLVER
Rank of X_train:  374
Number of non zero singular values:  386
Residuals for train set:  1581.822407908945
Maximum error for train set:  49.31879889962739
Minimum error for train set:  4.973799150320701e-13
R2 for train set:  0.8663961504183206
Residuals for test set:  1056.7516844256186
```

```
Maximum error for test set:  46.86165824234698
Minimum error for test set:  0.0005819565922635661
R2 for test set:  0.8602019475524668

LSTSQ SOLVER
{'driver': 'gelsd', 'condition': 0.1, 'AVGtime': 1.46379656791687,
'residuals_train': 6242.025534840823, 'Min error residuals_train':
0.0017990607773157308, 'Max error residuals_train': 86.14804612741862,
'R2_train': -1.080437181562219, 'residuals_test': 4089.558851795378,
'Min error residuals_test': 0.0019065736120893462, 'Max error
residuals_test': 86.23257111283189, 'R2_test': -1.0936696064790175}
{'driver': 'gelss', 'condition': 0.1, 'AVGtime': 1.5929749488830567,
'residuals_train': 6242.025534840823, 'Min error residuals_train':
0.001799060777251782, 'Max error residuals_train': 86.14804612741861,
'R2_train': -1.080437181562219, 'residuals_test': 4089.558851795378,
'Min error residuals_test': 0.0019065736121319787, 'Max error
residuals_test': 86.23257111283188, 'R2_test': -1.0936696064790175}
{'driver': 'gelsy', 'condition': 0.1, 'AVGtime': 2.054758977890015,
'residuals_train': 6242.048764526194, 'Min error residuals_train':
0.00018963670160587753, 'Max error residuals_train':
86.15872329147919, 'R2_train': -1.0804526662765381, 'residuals_test':
4089.5916751438244, 'Min error residuals_test': 0.005121768266448612,
'Max error residuals_test': 86.24324837727318, 'R2_test': -
1.0937032147606436}
{'driver': 'gelsd', 'condition': 0.01, 'AVGtime': 1.317377233505249,
'residuals_train': 2524.7404922403675, 'Min error residuals_train':
0.0004900707692385708, 'Max error residuals_train': 59.87019625700878,
'R2_train': 0.6596411749341676, 'residuals_test': 1654.2352059891818,
'Min error residuals_test': 0.0004001028437556897, 'Max error
residuals_test': 59.98093166872872, 'R2_test': 0.6574296727962979}
{'driver': 'gelss', 'condition': 0.01, 'AVGtime': 1.6145201683044434,
'residuals_train': 2524.740492240369, 'Min error residuals_train':
0.0004900707691106732, 'Max error residuals_train':
59.870196257008764, 'R2_train': 0.6596411749341673, 'residuals_test':
1654.2352059891825, 'Min error residuals_test': 0.00040010284392622,
'Max error residuals_test': 59.98093166872871, 'R2_test':
0.6574296727962976}
{'driver': 'gelsy', 'condition': 0.01, 'AVGtime': 2.1120631217956545,
'residuals_train': 6242.048764526194, 'Min error residuals_train':
0.00018963670160587753, 'Max error residuals_train':
86.15872329147919, 'R2_train': -1.0804526662765381, 'residuals_test':
4089.5916751438244, 'Min error residuals_test': 0.005121768266448612,
'Max error residuals_test': 86.24324837727318, 'R2_test': -
1.0937032147606436}
{'driver': 'gelsd', 'condition': 0.0001, 'AVGtime':
1.3047335147857666, 'residuals_train': 1581.8437697309062, 'Min error
residuals_train': 0.0007629050405597582, 'Max error residuals_train':
49.31676311501686, 'R2_train': 0.8663925418704774, 'residuals_test':
1056.7598827874262, 'Min error residuals_test': 0.0005193352064569723,
```

```
'Max error residuals_test': 46.859145978843486, 'R2_test':
0.8601997784157118}
{'driver': 'gelss', 'condition': 0.0001, 'AVGtime':
1.6330953598022462, 'residuals_train': 1581.8437697309062, 'Min error
residuals_train': 0.0007629050438566765, 'Max error residuals_train':
49.31676311501602, 'R2_train': 0.8663925418704774, 'residuals_test':
1056.7598827874242, 'Min error residuals_test': 0.0005193352055457012,
'Max error residuals_test': 46.859145978843046, 'R2_test':
0.8601997784157124}
{'driver': 'gelsy', 'condition': 0.0001, 'AVGtime':
2.0623705863952635, 'residuals_train': 1581.8458133076267, 'Min error
residuals_train': 0.00027573671723502, 'Max error residuals_train':
49.314168214308275, 'R2_train': 0.866392196656528, 'residuals_test':
1056.762181979197, 'Min error residuals_test': 0.0002701702831870989,
'Max error residuals_test': 46.858824935300504, 'R2_test':
0.8601991700885527}
{'driver': 'gelsd', 'condition': 1e-08, 'AVGtime': 1.2989150524139403,
'residuals_train': 1581.8224079089448, 'Min error residuals_train':
6.231459792616079e-12, 'Max error residuals_train':
49.318798899628355, 'R2_train': 0.8663961504183206, 'residuals_test':
1056.7516844256204, 'Min error residuals_test': 0.0005819565920646141,
'Max error residuals_test': 46.8616582423474, 'R2_test':
0.8602019475524664}
{'driver': 'gelss', 'condition': 1e-08, 'AVGtime': 1.6138881206512452,
'residuals_train': 1581.822407908945, 'Min error residuals_train':
7.105427357601002e-14, 'Max error residuals_train': 49.31879889962756,
'R2_train': 0.8663961504183206, 'residuals_test': 1056.7516844256186,
'Min error residuals_test': 0.000581956592178301, 'Max error
residuals_test': 46.86165824234698, 'R2_test': 0.8602019475524668}
{'driver': 'gelsy', 'condition': 1e-08, 'AVGtime': 2.0616582870483398,
'residuals_train': 1581.822407908945, 'Min error residuals_train':
1.1013412404281553e-13, 'Max error residuals_train':
49.31879889962847, 'R2_train': 0.8663961504183206, 'residuals_test':
1056.7516844256193, 'Min error residuals_test': 0.0005819565928035786,
'Max error residuals_test': 46.8616582423479, 'R2_test':
0.8602019475524667}
{'driver': 'gelsd', 'condition': 1e-12, 'AVGtime': 1.3123390674591064,
'residuals_train': 1581.8224079089448, 'Min error residuals_train':
6.231459792616079e-12, 'Max error residuals_train':
49.318798899628355, 'R2_train': 0.8663961504183206, 'residuals_test':
1056.7516844256204, 'Min error residuals_test': 0.0005819565920646141,
'Max error residuals_test': 46.8616582423474, 'R2_test':
0.8602019475524664}
{'driver': 'gelss', 'condition': 1e-12, 'AVGtime': 1.6143386840820313,
'residuals_train': 1581.822407908945, 'Min error residuals_train':
7.105427357601002e-14, 'Max error residuals_train': 49.31879889962756,
'R2_train': 0.8663961504183206, 'residuals_test': 1056.7516844256186,
'Min error residuals_test': 0.000581956592178301, 'Max error
residuals_test': 46.86165824234698, 'R2_test': 0.8602019475524668}
```

```
{'driver': 'gelsy', 'condition': 1e-12, 'AVGtime': 2.0766768932342528,
'residuals_train': 1581.822407908945, 'Min error residuals_train':
1.1013412404281553e-13, 'Max error residuals_train':
49.31879889962847, 'R2_train': 0.8663961504183206, 'residuals_test':
1056.7516844256193, 'Min error residuals_test': 0.0005819565928035786,
'Max error residuals_test': 46.8616582423479, 'R2_test':
0.8602019475524667}
{'driver': 'gelsd', 'condition': 1e-16, 'AVGtime': 1.30711932182312,
'residuals_train': 1581.8224079089448, 'Min error residuals_train':
6.231459792616079e-12, 'Max error residuals_train':
49.318798899628355, 'R2_train': 0.8663961504183206, 'residuals_test':
1056.7516844256204, 'Min error residuals_test': 0.0005819565920646141,
'Max error residuals_test': 46.8616582423474, 'R2_test':
0.8602019475524664}
{'driver': 'gelss', 'condition': 1e-16, 'AVGtime': 1.6059051990509032,
'residuals_train': 1581.822407908945, 'Min error residuals_train':
7.105427357601002e-14, 'Max error residuals_train': 49.31879889962756,
'R2_train': 0.8663961504183206, 'residuals_test': 1056.7516844256186,
'Min error residuals_test': 0.000581956592178301, 'Max error
residuals_test': 46.86165824234698, 'R2_test': 0.8602019475524668}
{'driver': 'gelsy', 'condition': 1e-16, 'AVGtime': 2.0868249893188477,
'residuals_train': 1581.822407908945, 'Min error residuals_train':
1.1013412404281553e-13, 'Max error residuals_train':
49.31879889962847, 'R2_train': 0.8663961504183206, 'residuals_test':
1056.7516844256193, 'Min error residuals_test': 0.0005819565928035786,
'Max error residuals_test': 46.8616582423479, 'R2_test':
0.8602019475524667}

PCR SOLVER
k= 373  with mixed error criterion and tol= 1e-06
Sigma_k= 0.2419127715529588
Residuals for train set:  1581.8224258404935
Maximum error for train set:  49.31887428974955
Minimum error for train set:  1.0329512576845445e-05
R2 for train set:  0.8663961473892525
Residuals for test set:  1056.751680001771
Maximum error for test set:  46.861719477894866
Minimum error for test set:  0.0005178873795301797
R2 for test set:  0.8602019487229315
```




```
k= 5  with Screep plot
Sigma_k= 221.56058987580698
Residuals for train set:  3827.891659917939
Maximum error for train set:  77.98276634172981
Minimum error for train set:  0.0007443352258604818
R2 for train set:  0.21761088436209697
Residuals for test set:  2501.375548893731
Maximum error for test set:  82.1296511792311
Minimum error for test set:  0.0006695430814858128
R2 for test set:  0.21672723822862605
k= 1 with cumulative percentage of variance and p= 0.99
Sigma_k= 606.6409563436413
Residuals for train set:  6242.025534840823
Maximum error for train set:  86.14804612741864
Minimum error for train set:  0.0017990607773654688
R2 for train set:  -1.080437181562219
Residuals for test set:  4089.558851795378
Maximum error for test set:  86.23257111283189
Minimum error for test set:  0.001906573612053819
R2 for test set:  -1.093669606479018

X = X - np.mean(X, axis=0)
y=y-np.mean(y)
```

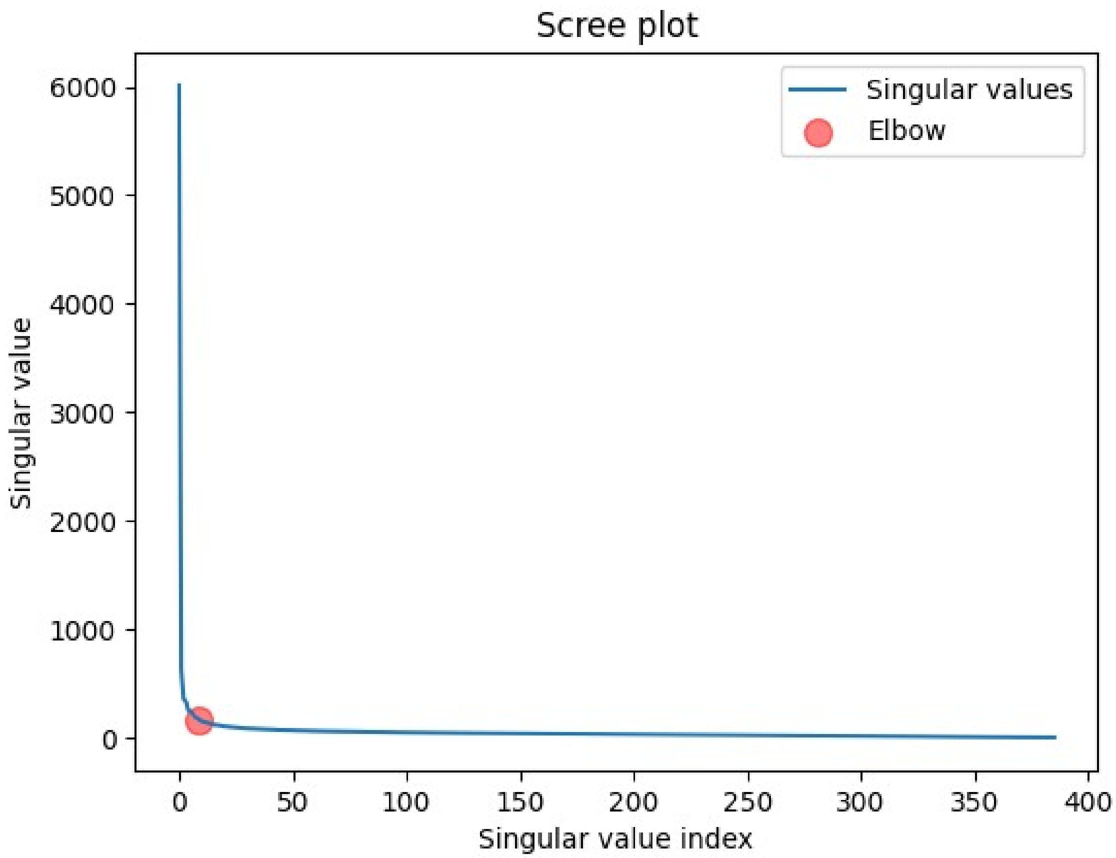
```
X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    train_size = .9,
    test_size = .1,
    random_state = 5,
    shuffle = True
)

print("\nNORMAL EQUATIONS")
normalEquations(X_train, y_train, X_test, y_test)
print("\nQR SOLVER")
QRsolver(X_train, y_train, X_test, y_test)
print("\nSVD SOLVER")
SVDSolver(X_train, y_train, X_test, y_test)
print("\nLSTSQ SOLVER")
lstsqSolver(X_train, y_train, X_test, y_test)
print("\nPCR SOLVER")
PCRSolver(X_train, y_train, X_test, y_test)
```

```
NORMAL EQUATIONS
Rank of train data:  374
Shape of train data:  (386, 386)
Matrix is singular

QR SOLVER
Condition number of QR factorization matrix:  inf
Residuals for train set:  1798.375168923055
Maximum error for train set:  49.48141087921306
Minimum error for train set:  0.00010451105330844257
R2 for train set:  0.8653829386563148
Residuals for test set:  613.1883250084952
Maximum error for test set:  46.99654288414338
Minimum error for test set:  0.0006169212601747631
R2 for test set:  0.8603253170982564

SVD SOLVER
Rank of X_train:  374
Number of non zero singular values:  386
Residuals for train set:  1798.375168923055
Maximum error for train set:  49.48141087921296
Minimum error for train set:  0.00010451105312370146
R2 for train set:  0.8653829386563148
Residuals for test set:  613.1883250084953
Maximum error for test set:  46.99654288414334
Minimum error for test set:  0.0006169212603666097
R2 for test set:  0.8603253170982563
```

```
LSTSQ SOLVER
{'driver': 'gelsd', 'condition': 0.1, 'AVGtime': 1.7849128723144532,
'residuals_train': 4755.570567097338, 'Min error residuals_train':
0.000262427226318529, 'Max error residuals_train': 50.36096798485837,
'R2_train': 0.05866377462668071, 'residuals_test': 1587.9902364224715,
'Min error residuals_test': 0.0012715306054857933, 'Max error
residuals_test': 50.13814410503954, 'R2_test': 0.0632459161733121}
{'driver': 'gelss', 'condition': 0.1, 'AVGtime': 1.945432186126709,
'residuals_train': 4755.570567097334, 'Min error residuals_train':
0.0002624272262981009, 'Max error residuals_train': 50.36096798485853,
'R2_train': 0.05866377462668182, 'residuals_test': 1587.9902364224702,
'Min error residuals_test': 0.0012715306053419084, 'Max error
residuals_test': 50.1381441050397, 'R2_test': 0.06324591617331332}
{'driver': 'gelsy', 'condition': 0.1, 'AVGtime': 2.79490442276001,
'residuals_train': 4900.828826291249, 'Min error residuals_train':
0.0003575149926536292, 'Max error residuals_train': 50.81533254000823,
'R2_train': 0.00027954336952462633, 'residuals_test':
1640.5141987499783, 'Min error residuals_test': 0.0003739759662430009,
'Max error residuals_test': 50.89981613229195, 'R2_test':
0.0002534215123161099}
{'driver': 'gelsd', 'condition': 0.01, 'AVGtime': 1.8146845340728759,
'residuals_train': 2108.137325866397, 'Min error residuals_train':
0.00013572860698474187, 'Max error residuals_train':
50.161593481761926, 'R2_train': 0.815014662390456, 'residuals_test':
703.1095775793399, 'Min error residuals_test': 0.00036970048081386153,
'Max error residuals_test': 45.97968388437563, 'R2_test':
0.8163563284379058}
{'driver': 'gelss', 'condition': 0.01, 'AVGtime': 2.1100863933563234,
'residuals_train': 2108.1373258663994, 'Min error residuals_train':
0.00013572860682842247, 'Max error residuals_train':
50.161593481763525, 'R2_train': 0.8150146623904556, 'residuals_test':
703.1095775793397, 'Min error residuals_test': 0.00036970048049234094,
'Max error residuals_test': 45.97968388437485, 'R2_test':
0.816356328437906}
{'driver': 'gelsy', 'condition': 0.01, 'AVGtime': 3.0165273666381838,
'residuals_train': 2520.2405401243936, 'Min error residuals_train':
7.746705687594613e-06, 'Max error residuals_train': 47.27200080131459,
'R2_train': 0.7356231050963493, 'residuals_test': 838.5783311330284,
'Min error residuals_test': 0.0005136025978078607, 'Max error
residuals_test': 45.74035126344225, 'R2_test': 0.7387735121708896}
{'driver': 'gelsd', 'condition': 0.0001, 'AVGtime':
1.7419917583465576, 'residuals_train': 1798.3820295191035, 'Min error
residuals_train': 6.735755479780892e-05, 'Max error residuals_train':
49.48069319797044, 'R2_train': 0.865381911556901, 'residuals_test':
613.1889342656056, 'Min error residuals_test': 0.0013570743457371748,
'Max error residuals_test': 46.995565748867016, 'R2_test':
0.8603250395396909}
{'driver': 'gelss', 'condition': 0.0001, 'AVGtime':
1.7329828262329101, 'residuals_train': 1798.3820295191033, 'Min error
```

```
residuals_train': 6.735755465214766e-05, 'Max error residuals_train':
49.480693197970695, 'R2_train': 0.865381911556901, 'residuals_test':
613.1889342656057, 'Min error residuals_test': 0.0013570743452895329,
'Max error residuals_test': 46.99556574886715, 'R2_test':
0.8603250395396908}
{'driver': 'gelsy', 'condition': 0.0001, 'AVGtime':
2.7779926300048827, 'residuals_train': 1798.382051640633, 'Min error
residuals_train': 0.00024318692035407707, 'Max error residuals_train':
49.48074237739562, 'R2_train': 0.8653819082450819, 'residuals_test':
613.1877983373976, 'Min error residuals_test': 0.0014086055356834493,
'Max error residuals_test': 46.99563332089457, 'R2_test':
0.8603255570329844}
{'driver': 'gelsd', 'condition': 1e-08, 'AVGtime': 1.7098489284515381,
'residuals_train': 1798.3751689230546, 'Min error residuals_train':
0.00010451105293896035, 'Max error residuals_train': 49.4814108792126,
'R2_train': 0.8653829386563148, 'residuals_test': 613.1883250084949,
'Min error residuals_test': 0.000616921259741332, 'Max error
residuals_test': 46.996542884143025, 'R2_test': 0.8603253170982565}
{'driver': 'gelss', 'condition': 1e-08, 'AVGtime': 1.774636697769165,
'residuals_train': 1798.375168923055, 'Min error residuals_train':
0.00010451105289632778, 'Max error residuals_train':
49.48141087921289, 'R2_train': 0.8653829386563148, 'residuals_test':
613.1883250084951, 'Min error residuals_test': 0.0006169212602671337,
'Max error residuals_test': 46.99654288414318, 'R2_test':
0.8603253170982565}
{'driver': 'gelsy', 'condition': 1e-08, 'AVGtime': 2.7203440189361574,
'residuals_train': 1798.375168923055, 'Min error residuals_train':
0.00010451105330133714, 'Max error residuals_train':
49.48141087921306, 'R2_train': 0.8653829386563147, 'residuals_test':
613.1883250084953, 'Min error residuals_test': 0.0006169212601605523,
'Max error residuals_test': 46.99654288414338, 'R2_test':
0.8603253170982564}
{'driver': 'gelsd', 'condition': 1e-12, 'AVGtime': 1.6896587371826173,
'residuals_train': 1798.3751689230546, 'Min error residuals_train':
0.00010451105293896035, 'Max error residuals_train': 49.4814108792126,
'R2_train': 0.8653829386563148, 'residuals_test': 613.1883250084949,
'Min error residuals_test': 0.000616921259741332, 'Max error
residuals_test': 46.996542884143025, 'R2_test': 0.8603253170982565}
{'driver': 'gelss', 'condition': 1e-12, 'AVGtime': 1.7570572376251221,
'residuals_train': 1798.375168923055, 'Min error residuals_train':
0.00010451105289632778, 'Max error residuals_train':
49.48141087921289, 'R2_train': 0.8653829386563148, 'residuals_test':
613.1883250084951, 'Min error residuals_test': 0.0006169212602671337,
'Max error residuals_test': 46.99654288414318, 'R2_test':
0.8603253170982565}
{'driver': 'gelsy', 'condition': 1e-12, 'AVGtime': 2.7103029251098634,
'residuals_train': 1798.375168923055, 'Min error residuals_train':
0.00010451105330133714, 'Max error residuals_train':
49.48141087921306, 'R2_train': 0.8653829386563147, 'residuals_test':
```

```
613.1883250084953, 'Min error residuals_test': 0.0006169212601605523,
'Max error residuals_test': 46.99654288414338, 'R2_test':
0.8603253170982564}
{'driver': 'gelsd', 'condition': 1e-16, 'AVGtime': 1.669171190261841,
'residuals_train': 1798.3751689230546, 'Min error residuals_train':
0.00010451105293896035, 'Max error residuals_train': 49.4814108792126,
'R2_train': 0.8653829386563148, 'residuals_test': 613.1883250084949,
'Min error residuals_test': 0.000616921259741332, 'Max error
residuals_test': 46.996542884143025, 'R2_test': 0.8603253170982565}
{'driver': 'gelss', 'condition': 1e-16, 'AVGtime': 1.7121876716613769,
'residuals_train': 1798.375168923055, 'Min error residuals_train':
0.00010451105289632778, 'Max error residuals_train':
49.48141087921289, 'R2_train': 0.8653829386563148, 'residuals_test':
613.1883250084951, 'Min error residuals_test': 0.0006169212602671337,
'Max error residuals_test': 46.99654288414318, 'R2_test':
0.8603253170982565}
{'driver': 'gelsy', 'condition': 1e-16, 'AVGtime': 2.6775108337402345,
'residuals_train': 1798.375168923055, 'Min error residuals_train':
0.00010451105330133714, 'Max error residuals_train':
49.48141087921306, 'R2_train': 0.8653829386563147, 'residuals_test':
613.1883250084953, 'Min error residuals_test': 0.0006169212601605523,
'Max error residuals_test': 46.99654288414338, 'R2_test':
0.8603253170982564}

PCR SOLVER
k= 373  with mixed error criterion and tol= 1e-06
Sigma_k= 0.24804292714425785
Residuals for train set:  1798.3765462771964
Maximum error for train set:  49.48194745002998
Minimum error for train set:  6.089655833818597e-05
R2 for train set:  0.865382732453025
Residuals for test set:  613.1881807499456
Maximum error for test set:  46.9969652643756
Minimum error for test set:  0.0002830075600641635
R2 for test set:  0.8603253828179187
```




```
k= 9  with Screep plot
Sigma_k= 160.0371010848383
Residuals for train set:  2834.123989129834
Maximum error for train set:  53.56869903518497
Minimum error for train set:  6.0258639919652524e-05
R2 for train set:  0.6656685720463946
Residuals for test set:  942.4515117574208
Maximum error for test set:  53.68821132157274
Minimum error for test set:  0.005007439769400435
R2 for test set:  0.6700501235002101
k= 46 with cumulative percentage of variance and p= 0.99
Sigma_k= 69.20495430475549
Residuals for train set:  2135.767310283201
Maximum error for train set:  50.95697458632548
Minimum error for train set:  0.00013314212461068564
R2 for train set:  0.810133921331676
Residuals for test set:  709.0843727595702
Maximum error for test set:  48.38086047422813
Minimum error for test set:  0.001812555850200681
R2 for test set:  0.8132219797565957
```

CHANGE DATASET DIMENSION

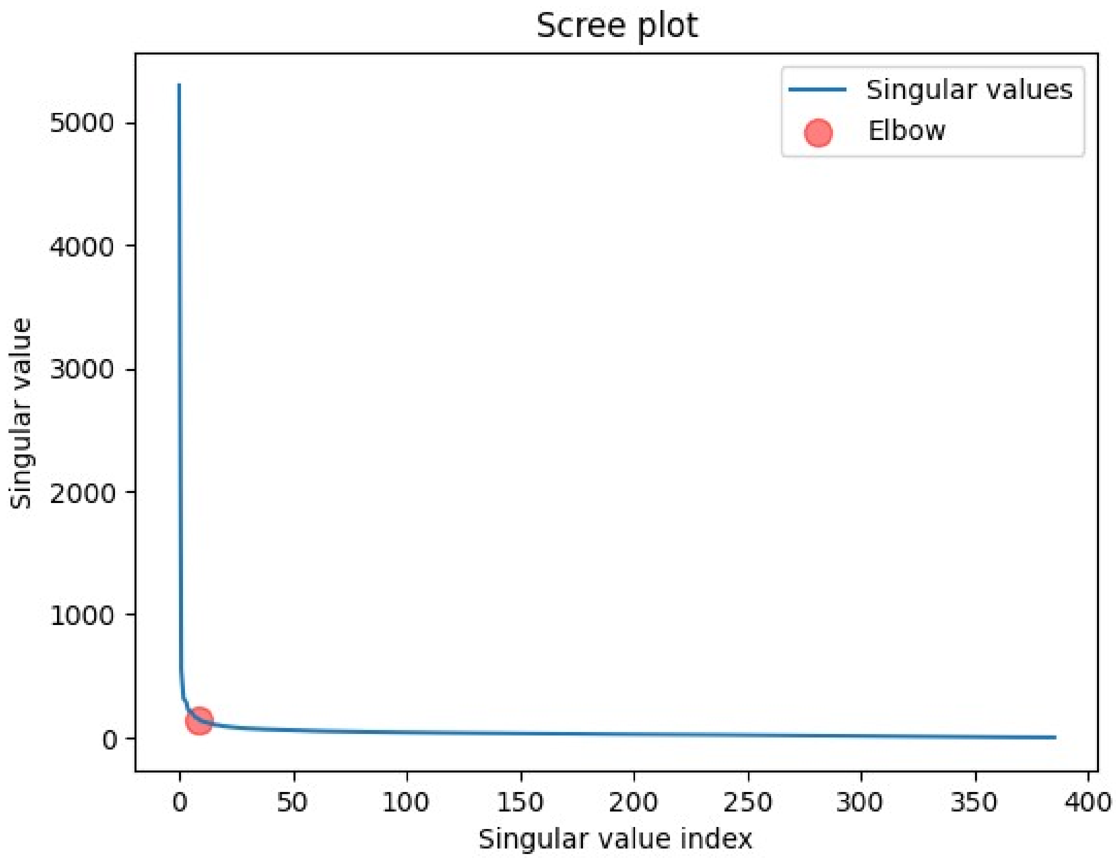
```
X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    train_size = 0.7,
    test_size = .3,
    random_state = 5,
    shuffle = True
)

print("\nNORMAL EQUATIONS")
normalEquations(X_train, y_train, X_test, y_test)
print("\nQR SOLVER")
QRsolver(X_train, y_train, X_test, y_test)
print("\nSVD SOLVER")
SVDSolver(X_train, y_train, X_test, y_test)
print("\nLSTSQ SOLVER")
lstsqSolver(X_train, y_train, X_test, y_test)
print("\nPCR SOLVER")
PCRSolver(X_train, y_train, X_test, y_test)
```

```
NORMAL EQUATIONS
Rank of train data:  373
Shape of train data:  (386, 386)
Matrix is singular

QR SOLVER
Condition number of QR factorization matrix:  inf
Residuals for train set:  1581.8224079089448
Maximum error for train set:  49.31879889963359
Minimum error for train set:  3.1938895972416503e-12
R2 for train set:  0.8663961504183206
Residuals for test set:  24323.803995926748
Maximum error for test set:  17502.659908336667
Minimum error for test set:  0.0005819565969105156
R2 for test set:  -73.06588457680839

SVD SOLVER
Rank of X_train:  374
Number of non zero singular values:  386
Residuals for train set:  1581.8224079089446
Maximum error for train set:  49.318798899683344
Minimum error for train set:  3.9126035744629917e-11
R2 for train set:  0.8663961504183207
Residuals for test set:  24323.80398699717
Maximum error for test set:  17502.659901901887
Minimum error for test set:  0.0005819565577045438
R2 for test set:  -73.06588452242733

LSTSQ SOLVER
```

```
{'driver': 'gelsd', 'condition': 0.1, 'AVGtime': 1.304492425918579,
'residuals_train': 4199.813466862267, 'Min error residuals_train':
0.0007778557055178048, 'Max error residuals_train': 50.34528515106919,
'R2_train': 0.05818949864139089, 'residuals_test': 2738.064978785508,
'Min error residuals_test': 0.0017434290058133683, 'Max error
residuals_test': 50.23110506691755, 'R2_test': 0.06148174467595835}
{'driver': 'gelss', 'condition': 0.1, 'AVGtime': 1.8272982120513916,
'residuals_train': 4199.813466862264, 'Min error residuals_train':
0.0007778557053468305, 'Max error residuals_train': 50.34528515106909,
'R2_train': 0.05818949864139222, 'residuals_test': 2738.0649787855064,
'Min error residuals_test': 0.001743429005886199, 'Max error
residuals_test': 50.23110506691745, 'R2_test': 0.061481744675960015}
{'driver': 'gelsy', 'condition': 0.1, 'AVGtime': 2.098579263687134,
'residuals_train': 4327.143540775084, 'Min error residuals_train':
0.003853264487539809, 'Max error residuals_train': 50.77097759824818,
'R2_train': 0.0002161229312992452, 'residuals_test':
2825.830505497923, 'Min error residuals_test': 0.0005380666435376158,
'Max error residuals_test': 50.85546110958141, 'R2_test':
0.0003512161608367758 7}
{'driver': 'gelsd', 'condition': 0.01, 'AVGtime': 1.3862902164459228,
'residuals_train': 1856.2978631758865, 'Min error residuals_train':
5.694623258634124e-05, 'Max error residuals_train': 50.37107954236377,
'R2_train': 0.816008036324967, 'residuals_test': 1218.4421714493662,
'Min error residuals_test': 0.00024393765594865613, 'Max error
residuals_test': 49.1741317140975, 'R2_test': 0.8141489292378865}
{'driver': 'gelss', 'condition': 0.01, 'AVGtime': 1.7717936038970947,
'residuals_train': 1856.297863175887, 'Min error residuals_train':
5.6946232447785405e-05, 'Max error residuals_train':
50.37107954236397, 'R2_train': 0.8160080363249669, 'residuals_test':
1218.4421714493665, 'Min error residuals_test':
0.00024393765635100095, 'Max error residuals_test':
49.174131714097875, 'R2_test': 0.8141489292378865}
{'driver': 'gelsy', 'condition': 0.01, 'AVGtime': 2.491049957275391,
'residuals_train': 2262.0315035962994, 'Min error residuals_train':
0.000878605419591949, 'Max error residuals_train': 46.14311581758492,
'R2_train': 0.7267873377369896, 'residuals_test': 1483.2331242384055,
'Min error residuals_test': 0.000640977382154758, 'Max error
residuals_test': 45.62933976984492, 'R2_test': 0.7245935829332961}
{'driver': 'gelsd', 'condition': 0.0001, 'AVGtime':
1.4514110565185547, 'residuals_train': 1581.830968104192, 'Min error
residuals_train': 0.0010936828899694717, 'Max error residuals_train':
49.30710240018327, 'R2_train': 0.866394704392361, 'residuals_test':
1056.7479613315134, 'Min error residuals_test':
0.00040345495549054533, 'Max error residuals_test': 46.84894729577002,
'R2_test': 0.8602029326095919}
{'driver': 'gelss', 'condition': 0.0001, 'AVGtime':
2.0281887531280516, 'residuals_train': 1581.8309681041917, 'Min error
residuals_train': 0.0010936828900449669, 'Max error residuals_train':
49.30710240018255, 'R2_train': 0.866394704392361, 'residuals_test':
```

```
1056.7479613315134, 'Min error residuals_test': 0.0004034549560731904,
'Max error residuals_test': 46.84894729576919, 'R2_test':
0.8602029326095919}
{'driver': 'gelsy', 'condition': 0.0001, 'AVGtime': 2.345961332321167,
'residuals_train': 1581.831004882127, 'Min error residuals_train':
0.0003969291845997702, 'Max error residuals_train': 49.307162353691,
'R2_train': 0.8663946981796531, 'residuals_test': 1056.7484149362238,
'Min error residuals_test': 0.00031059155839230357, 'Max error
residuals_test': 46.84905600491508, 'R2_test': 0.8602028125949354}
{'driver': 'gelsd', 'condition': 1e-08, 'AVGtime': 1.5774375915527343,
'residuals_train': 1581.822407908945, 'Min error residuals_train':
1.5376144801848568e-11, 'Max error residuals_train':
49.31879889962687, 'R2_train': 0.8663961504183207, 'residuals_test':
24323.803994354585, 'Min error residuals_test': 0.0005819566007279064,
'Max error residuals_test': 17502.65990720368, 'R2_test': -
73.06588456723391}
{'driver': 'gelss', 'condition': 1e-08, 'AVGtime': 2.3048064708709717,
'residuals_train': 1581.8224079089453, 'Min error residuals_train':
1.0899725566559937e-11, 'Max error residuals_train':
49.318798899612474, 'R2_train': 0.8663961504183206, 'residuals_test':
24323.80399435412, 'Min error residuals_test': 0.0005819566077427396,
'Max error residuals_test': 17502.659907203324, 'R2_test': -
73.06588456723107}
{'driver': 'gelsy', 'condition': 1e-08, 'AVGtime': 2.5028640747070314,
'residuals_train': 1581.822407908945, 'Min error residuals_train':
3.5420555377640994e-12, 'Max error residuals_train':
49.31879889963379, 'R2_train': 0.8663961504183206, 'residuals_test':
24323.803995927075, 'Min error residuals_test': 0.0005819565974967134,
'Max error residuals_test': 17502.659908336907, 'R2_test': -
73.06588457681035}
{'driver': 'gelsd', 'condition': 1e-12, 'AVGtime': 1.7076046466827393,
'residuals_train': 1581.822407908945, 'Min error residuals_train':
1.5376144801848568e-11, 'Max error residuals_train':
49.31879889962687, 'R2_train': 0.8663961504183207, 'residuals_test':
24323.803994354585, 'Min error residuals_test': 0.0005819566007279064,
'Max error residuals_test': 17502.65990720368, 'R2_test': -
73.06588456723391}
{'driver': 'gelss', 'condition': 1e-12, 'AVGtime': 2.104235792160034,
'residuals_train': 1581.8224079089453, 'Min error residuals_train':
1.0899725566559937e-11, 'Max error residuals_train':
49.318798899612474, 'R2_train': 0.8663961504183206, 'residuals_test':
24323.80399435412, 'Min error residuals_test': 0.0005819566077427396,
'Max error residuals_test': 17502.659907203324, 'R2_test': -
73.06588456723107}
{'driver': 'gelsy', 'condition': 1e-12, 'AVGtime': 2.6398069858551025,
'residuals_train': 1581.822407908945, 'Min error residuals_train':
3.5420555377640994e-12, 'Max error residuals_train':
49.31879889963379, 'R2_train': 0.8663961504183206, 'residuals_test':
24323.803995927075, 'Min error residuals_test': 0.0005819565974967134,
```

```
'Max error residuals_test': 17502.659908336907, 'R2_test': -
73.06588457681035}
{'driver': 'gelsd', 'condition': 1e-16, 'AVGtime': 1.639665460586548,
'residuals_train': 1581.8230194231592, 'Min error residuals_train':
0.00010492014328988883, 'Max error residuals_train':
49.310829051343205, 'R2_train': 0.8663960471189004, 'residuals_test':
20758.818481852188, 'Min error residuals_test': 0.0010493647844036502,
'Max error residuals_test': 14933.18677097949, 'R2_test': -
52.946153831635115}
{'driver': 'gelss', 'condition': 1e-16, 'AVGtime': 2.2896727085113526,
'residuals_train': 1581.8224079089453, 'Min error residuals_train':
1.0899725566559937e-11, 'Max error residuals_train':
49.318798899612474, 'R2_train': 0.8663961504183206, 'residuals_test':
24323.80399435412, 'Min error residuals_test': 0.0005819566077427396,
'Max error residuals_test': 17502.659907203324, 'R2_test': -
73.06588456723107}
{'driver': 'gelsy', 'condition': 1e-16, 'AVGtime': 2.146107482910156,
'residuals_train': 1581.822407908945, 'Min error residuals_train':
3.5420555377640994e-12, 'Max error residuals_train':
49.31879889963379, 'R2_train': 0.8663961504183206, 'residuals_test':
24323.803995927075, 'Min error residuals_test': 0.0005819565974967134,
'Max error residuals_test': 17502.659908336907, 'R2_test': -
73.06588457681035}

PCR SOLVER
k= 372  with mixed error criterion and tol= 1e-06
Sigma_k= 0.24758868405284473
Residuals for train set:  1581.8256182319878
Maximum error for train set:  49.308567591462875
Minimum error for train set:  0.0008552917770172641
R2 for train set:  0.8663956081173012
Residuals for test set:  1056.745414663184
Maximum error for test set:  46.8505079503417
Minimum error for test set:  6.500971892187124e-05
R2 for test set:  0.8602036064057064
```




```
k= 9  with Screep plot
Sigma_k= 141.26352434443368
Residuals for train set:  2497.332145821892
Maximum error for train set:  53.68137331724654
Minimum error for train set:  0.0001362779641826961
R2 for train set:  0.6669908704885368
Residuals for test set:  1637.0185070544176
Maximum error for test set:  53.81545195900935
Minimum error for test set:  0.0005248476698405113
R2 for test set:  0.6645232691281926
k= 46 with cumulative percentage of variance and p= 0.99
Sigma_k= 61.22165898458327
Residuals for train set:  1883.4871974261753
Maximum error for train set:  50.84759004578369
Minimum error for train set:  0.00034606460312680554
R2 for train set:  0.8105786750485895
Residuals for test set:  1233.3201425743985
Maximum error for test set:  51.03784522031048
Minimum error for test set:  0.0009238708125600681
R2 for test set:  0.8095824940245349
```

For a setted configuration of the training set and the testing set, we have the following results:

1. **Normal Equation**: The normal equation cannot be used because the matrix $X^T X$ is not full rank.
2. **QR Factorization**: The QR factorization cannot be used because the matrix X is not full rank, so we use the Pivot QR factorization.
3. **SVD**: The SVD can be used to solve the linear regression problem. SVD and QR factorization have almost the same results
4. **Scipy.linalg.lstsq**: We can notice that, decreasing the value of cond, the residual error decreases as the errors and R2. Regarding the execution time, we can notiche that in general the gelsy driver is slower than the other two. The execution time tends to increase with the decrease of the value of cond. At the beginning, gelsy performs worse than the other two, but as the value of cond decreases, the execution time of gelsy becomes closer to the other two drivers.
5. **PCR**: When the value of the last singular values increas, the error gets worse. This happens also for the cond number in ltsq

Train Set	Data Centering	Solver	R2 Train	R2 Test	Res Train	Res Test	Max Err Train	Min Err Train	Max Err Test	Min Err Test
90%	No	QR	0.865	0.860	1798.37	613.19	49.47	1.43e-12	46.98	0.002
70%	No	QR	0.866	0.860	1581.82	1056.74	49.31	3.19e-14	46.86	0.0006
90%	Yes	QR	0.865	0.860	1798.37	613.18	49.48	0.0001	46.99	0.0006
70%	Yes	QR	0.866	-73.065	1581.82	24323.80	49.31	3.19e-12	17502.65	0.0006

For the QR method, we can see that for the train set results are very similar, we have better result with less data in the training set with an improvement when the data is centered. For the test set we have better results with more data in the training set. The last configuration is the worst one, with a very high error in the test set.

Train Set	Data Centering	Solver	R2 Train	R2 Test	Res Train	Res Test	Max Err Train	Min Err Train	Max Err Test	Min Err Test
90%	No	SVD	0.865	0.860	1798.37	613.19	49.47	5.61e-13	46.98	0.002
70%	No	SVD	0.866	0.860	1581.82	1056.74	49.31	3.19e-14	46.86	0.0006
90%	Yes	SVD	0.865	0.860	1798.37	613.18	49.48	0.0001	46.99	0.0006
70%	Yes	SVD	0.866	-73.065	1581.82	24323.80	49.31	3.19e-11	17502.65	0.0006

For the SVD method, we can see that for the train set results are very similar, we have better result with less data in the training set with an improvement when the data is centered. For the test set we have better results with more data in the training set. The last configuration is the worst one, with a very high error in the test set.

Train Set	Data Centering	Solver	Condition	R2 Train	R2 Test	Res Train	Res Test	Max Err Train	Min Err Train	Max Err Test	Min Err Test	Sigma_k
90%	No	PCR	mixed error criterion, tol=1e-06	0.865	0.860	1798.37	613.19	49.47	8.88e-06	46.98	0.002	0.242
90%	No	PCR	Scree plot	0.225	0.239	4314.87	1430.35	81.80	4.23e-05	72.85	0.004	250.92
90%	No	PCR	cumulative variance, p=0.99	-1.085	-1.070	7079.00	2361.14	86.12	0.0001	86.21	0.001	688.74
70%	No	PCR	mixed error criterion, tol=1e-06	0.866	0.860	1581.82	1056.75	49.31	1.03e-05	46.86	0.0005	0.241
70%	No	PCR	Scree plot	0.217	0.216	3827.89	2501.37	77.98	0.0007	82.12	0.0006	221.56
70%	No	PCR	cumulative variance, p=0.99	-1.080	-1.093	6242.02	4089.55	86.14	0.001	86.23	0.001	606.64
90%	Yes	PCR	mixed error criterion, tol=1e-06	0.865	0.860	1798.37	613.18	49.48	6.08e-05	46.99	0.00028	0.248
90%	Yes	PCR	Scree plot	0.665	0.670	2834.12	942.45	53.56	6.02e-05	53.68	0.005	160.00

Train Set	Data Centering	Solver	Condition	R2 Train	R2 Test	Res Train	Res Test	Max Err Train	Min Err Train	Max Err Test	Min Err Test	Sigma_k
90%	Yes	PCR	cumulative variance, p=0.99	0.810	0.813	2135.76	709.08	50.95	0.0001	48.38	0.001	369.20
70%	Yes	PCR	mixed error criterion, tol=1e-06	0.866	0.860	1581.82	1056.74	49.30	0.0008	46.85	6.5e-5	0.247
70%	Yes	PCR	Scree plot	0.666	0.664	2497.33	1637.081	53.68	0.0001	53.81	0.0005	141.26
70%	Yes	PCR	cumulative variance, p=0.99	0.810	0.809	1883.48	1233.32	50.84	0.0003	51.03	0.0009	61.22

We can notice that in general the best criterion is the first one, of course because it takes more singular_values, so the approximation is better. When we center our data we get better results

Train Set	Data Centering	Solver	Condition	R2 Train	R2 Test	Res Train	Res Test	Max Err Train	Min Err Train	Max Err Test	Min Err Test	AVGtime
90%	No	LSTSQ (gelsd)	0.1	-1.08585	-1.07097	7079.00333	2361.14236	86.12880	0.00016	86.21332	0.00154	1.77860
90%	No	LSTSQ (gelsd)	0.01	0.66139	0.66675	2852.20918	947.15672	59.94904	0.00018	57.95974	0.00234	1.56410
90%	No	LSTSQ (gelsd)	0.0001	0.86538	0.86033	1798.39947	613.17157	49.47573	0.00002	46.98915	0.00144	1.56109

Train Set	Data Centering	Solver	Condition	R2 Train	R2 Test	Res Train	Res Test	Max Err Train	Min Err Train	Max Err Test	Min Err Test	AVGtime
90%	No	LSTSQ (gelsd)	1e-08	0.865638	0.860032	1798.37373	613.19675	49.47129	0.00000	46.98621	0.00233	1.569903
90%	No	LSTSQ (gelsd)	1e-12	0.865638	0.860032	1798.37373	613.19675	49.47129	0.00000	46.98621	0.00233	1.578846
90%	No	LSTSQ (gelsd)	1e-16	0.865638	0.860032	1798.37373	613.19675	49.47129	0.00000	46.98621	0.00233	1.663324
70%	No	LSTSQ (gelsd)	0.1	-1.08044	-1.09367	6242.02553	4089.55885	86.14805	0.00179	86.23257	0.00191	1.463380
70%	No	LSTSQ (gelsd)	0.01	0.65964	0.65743	2524.74049	1654.23521	59.87020	0.00049	59.98093	0.00040	1.317378
70%	No	LSTSQ (gelsd)	0.0001	0.86639	0.86020	1581.84377	1056.75988	49.31676	0.00076	46.85915	0.00052	1.304473
70%	No	LSTSQ (gelsd)	1e-08	0.86640	0.86020	1581.82241	1056.75168	49.31880	0.00000	46.86166	0.00058	1.29892
70%	No	LSTSQ (gelsd)	1e-12	0.86640	0.86020	1581.82241	1056.75168	49.31880	0.00000	46.86166	0.00058	1.312334
70%	No	LSTSQ (gelsd)	1e-16	0.86640	0.86020	1581.82241	1056.75168	49.31880	0.00000	46.86166	0.00058	1.307712
90%	Yes	LSTSQ (gelsd)	0.1	0.05866	0.0632	4755.5705	1587.99	50.36097	0.00026	50.13814	0.00127	1.78491

Train Set	Data Centering	Solver	Condition	R2 Train	R2 Test	Res Train	Res Test	Max Err Train	Min Err Train	Max Err Test	Min Err Test	AVG time
					5	7	024					
90%	Yes	LSTSQ (gelsd)	0.01	0.81501	0.816366	2108.13733	703.10958	50.16159	0.00014	45.97968	0.00037	1.81468
90%	Yes	LSTSQ (gelsd)	0.0001	0.86538	0.860333	1798.38203	613.18893	49.48069	0.00007	46.99557	0.00136	1.74199
90%	Yes	LSTSQ (gelsd)	1e-08	0.86538	0.860333	1798.37517	613.18833	49.48141	0.00010	46.99654	0.00062	1.70985
90%	Yes	LSTSQ (gelsd)	1e-12	0.86538	0.860333	1798.37517	613.18833	49.48141	0.00010	46.99654	0.00062	1.68966
90%	Yes	LSTSQ (gelsd)	1e-16	0.86538	0.860333	1798.37517	613.18833	49.48141	0.00010	46.99654	0.00062	1.66917
70%	Yes	LSTSQ (gelsd)	0.1	0.05819	0.061448	4199.81347	2738.06498	50.34529	0.00078	50.23111	0.00174	1.30449
70%	Yes	LSTSQ (gelsd)	0.01	0.81601	0.814155	1856.29786	1218.44217	50.37108	0.00006	49.17413	0.00024	1.38629
70%	Yes	LSTSQ (gelsd)	0.0001	0.86639	0.860200	1581.83097	1056.74796	49.30710	0.00109	46.84895	0.00040	1.45141
70%	Yes	LSTSQ (gelsd)	1e-08	0.86640	-73.065888	1581.82241	24323.80399	49.31880	0.00000	17502.65991	0.00058	1.57744

Train Set	Data Centering	Solver	Condition	R2 Train	R2 Test	Res Train	Res Test	Max Err Train	Min Err Train	Max Err Test	Min Err Test	AVGtime
70%	Yes	LSTSQ (gelsd)	1e-12	0.86640	-73.06588	1581.82241	24323.80399	49.31880	0.00000	17502.65991	0.00058	1.70760
70%	Yes	LSTSQ (gelsd)	1e-16	0.86640	-52.94615	1581.82302	20758.81848	49.31083	0.00010	14933.18677	0.00105	1.63967
90%	No	LSTSQ (gelss)	0.1	-1.08585	-1.07097	7079.00333	2361.14236	86.12880	0.00016	86.21332	0.00154	1.73561
90%	No	LSTSQ (gelss)	0.01	0.66139	0.66675	2852.20918	947.15672	59.94904	0.00018	57.95974	0.00234	1.64774
90%	No	LSTSQ (gelss)	0.0001	0.86538	0.86033	1798.39947	613.17157	49.47573	0.00002	46.98915	0.00144	1.64751
90%	No	LSTSQ (gelss)	1e-08	0.86538	0.86032	1798.37373	613.19675	49.47129	0.00000	46.98621	0.00233	1.63350
90%	No	LSTSQ (gelss)	1e-12	0.86538	0.86032	1798.37373	613.19675	49.47129	0.00000	46.98621	0.00233	1.68094
90%	No	LSTSQ (gelss)	1e-16	0.86538	0.86032	1798.37373	613.19675	49.47129	0.00000	46.98621	0.00233	1.74813
70%	No	LSTSQ (gelss)	0.1	-1.08044	-1.09367	6242.02553	4089.55885	86.14805	0.00179	86.23257	0.00191	1.59297
70	No	LSTSQ	0.0	0.6	0.	252	16	59.87	0.000	59.98	0.000	1.

Train Set	Data Centering	Solver	Condition	R2 Train	R2 Test	Res Train	Res Test	Max Err Train	Min Err Train	Max Err Test	Min Err Test	AVGtime
%		(gelss)	1	5964	65743	4.74049	54.23521	020	49	093	40	61452
70%	No	LSTSQ (gelss)	0.0001	0.866639	0.86020	1581.84377	1056.75988	49.31676	0.00076	46.85915	0.00052	1.63310
70%	No	LSTSQ (gelss)	1e-08	0.866640	0.86020	1581.82241	1056.75168	49.31880	0.00000	46.86166	0.00058	1.61389
70%	No	LSTSQ (gelss)	1e-12	0.866640	0.86020	1581.82241	1056.75168	49.31880	0.00000	46.86166	0.00058	1.61434
70%	No	LSTSQ (gelss)	1e-16	0.866640	0.86020	1581.82241	1056.75168	49.31880	0.00000	46.86166	0.00058	1.60591
90%	Yes	LSTSQ (gelss)	0.1	0.05866	0.06325	4755.57057	1587.99024	50.36097	0.00026	50.13814	0.00127	1.94543
90%	Yes	LSTSQ (gelss)	0.01	0.81501	0.81636	2108.13733	703.10958	50.16159	0.00014	45.97968	0.00037	2.11009
90%	Yes	LSTSQ (gelss)	0.0001	0.86538	0.86033	1798.38203	613.18893	49.48069	0.00007	46.99557	0.00136	1.73298
90%	Yes	LSTSQ (gelss)	1e-08	0.86538	0.86033	1798.37517	613.18833	49.48141	0.00010	46.99654	0.00062	1.77464
90%	Yes	LSTSQ (gelss)	1e-12	0.86538	0.86033	1798.37517	613.18833	49.48141	0.00010	46.99654	0.00062	1.75706

Train Set	Data Centering	Solver	Condition	R2 Train	R2 Test	Res Train	Res Test	Max Err Train	Min Err Train	Max Err Test	Min Err Test	AVGtime
90%	Yes	LSTSQ (gelss)	1e-16	0.86538	0.860333	1798.37517	613.18833	49.48141	0.00010	46.99654	0.00062	1.71219
70%	Yes	LSTSQ (gelss)	0.1	0.05819	0.06148	4199.81347	2738.06498	50.34529	0.00078	50.23111	0.00174	1.82730
70%	Yes	LSTSQ (gelss)	0.01	0.81601	0.81415	1856.29786	1218.44217	50.37108	0.00006	49.17413	0.00024	1.77179
70%	Yes	LSTSQ (gelss)	0.0001	0.86639	0.86020	1581.83097	1056.74796	49.30710	0.00109	46.84895	0.00040	2.02819
70%	Yes	LSTSQ (gelss)	1e-08	0.86640	-73.06588	1581.82241	24323.80399	49.31880	0.00000	17502.65991	0.00058	2.30481
70%	Yes	LSTSQ (gelss)	1e-12	0.86640	-73.06588	1581.82241	24323.80399	49.31880	0.00000	17502.65991	0.00058	2.10424
70%	Yes	LSTSQ (gelss)	1e-16	0.86640	-73.06588	1581.82241	24323.80399	49.31880	0.00000	17502.65991	0.00058	2.28967
90%	No	LSTSQ (gelsy)	0.1	-1.08587	-1.07093	7079.03005	2361.11959	86.13956	0.00110	86.22409	0.00349	2.61910
90%	No	LSTSQ (gelsy)	0.01	-1.085	-1.07	7079.0300	2361.119	86.13956	0.00110	86.22409	0.00349	2.6249

Train Set	Data Centering	Solver	Condition	R2 Train	R2 Test	Res Train	Res Test	Max Err Train	Min Err Train	Max Err Test	Min Err Test	AVGtime
				87	093	5	59					0
90%	No	LSTSQ (gelsy)	0.0001	0.86538	0.86032	1798.38823	613.19484	49.47471	0.00009	46.98724	0.00066	2.64965
90%	No	LSTSQ (gelsy)	1e-08	0.86538	0.86032	1798.37373	613.19675	49.47129	0.00000	46.98621	0.00233	2.63641
90%	No	LSTSQ (gelsy)	1e-12	0.86538	0.86032	1798.37373	613.19675	49.47129	0.00000	46.98621	0.00233	2.63356
90%	No	LSTSQ (gelsy)	1e-16	0.86538	0.86032	1798.37373	613.19675	49.47129	0.00000	46.98621	0.00233	2.67458
70%	No	LSTSQ (gelsy)	0.1	-1.08045	-1.09370	6242.04876	4089.59168	86.15872	0.00019	86.24325	0.00512	2.05476
70%	No	LSTSQ (gelsy)	0.01	-1.08045	-1.09370	6242.04876	4089.59168	86.15872	0.00019	86.24325	0.00512	2.11206
70%	No	LSTSQ (gelsy)	0.0001	0.86639	0.86020	1581.84581	1056.76218	49.31417	0.00028	46.85882	0.00027	2.06237
70%	No	LSTSQ (gelsy)	1e-08	0.86640	0.86020	1581.82241	1056.75168	49.31880	0.00000	46.86166	0.00058	2.06166
70%	No	LSTSQ (gelsy)	1e-12	0.86640	0.86020	1581.82241	1056.75168	49.31880	0.00000	46.86166	0.00058	2.07668

Train Set	Data Centering	Solver	Condition	R2 Train	R2 Test	Res Train	Res Test	Max Err Train	Min Err Train	Max Err Test	Min Err Test	AVG time
70%	No	LSTSQ (gelsy)	1e-16	0.86640	0.86020	1581.82241	1056.75168	49.31880	0.00000	46.86166	0.00058	2.08682
90%	Yes	LSTSQ (gelsy)	0.1	0.00028	0.00025	4900.82883	1640.51420	50.81533	0.00036	50.89982	0.00037	2.79490
90%	Yes	LSTSQ (gelsy)	0.01	0.73562	0.73877	2520.24054	838.57833	47.27200	0.00001	45.74035	0.00051	3.01653
90%	Yes	LSTSQ (gelsy)	0.0001	0.86538	0.86033	1798.38205	613.18780	49.48074	0.00024	46.99563	0.00141	2.77799
90%	Yes	LSTSQ (gelsy)	1e-08	0.86538	0.86033	1798.37517	613.18833	49.48141	0.00010	46.99654	0.00062	2.72034
90%	Yes	LSTSQ (gelsy)	1e-12	0.86538	0.86033	1798.37517	613.18833	49.48141	0.00010	46.99654	0.00062	2.71030
90%	Yes	LSTSQ (gelsy)	1e-16	0.86538	0.86033	1798.37517	613.18833	49.48141	0.00010	46.99654	0.00062	2.67751
70%	Yes	LSTSQ (gelsy)	0.1	0.00022	0.00035	4327.14354	2825.83051	50.77098	0.00385	50.85546	0.00054	2.09858
70%	Yes	LSTSQ (gelsy)	0.01	0.72679	0.72459	2262.03150	1483.23312	46.14312	0.00088	45.62934	0.00064	2.49105
70%	Yes	LSTSQ (gelsy)	0.0001	0.86639	0.8602	1581.83100	1056.74	49.30716	0.00040	46.84906	0.00031	2.3459

Train Set	Data Centering	Solver	Condition	R2 Train	R2 Test	Res Train	Res Test	Max Err Train	Min Err Train	Max Err Test	Min Err Test	AVGtime
					0		841					6
70%	Yes	LSTSQ (gelsy)	1e-08	0.86640	-73.06588	1581.82241	24323.80399	49.31880	0.00000	17502.65991	0.00058	2.50286
70%	Yes	LSTSQ (gelsy)	1e-12	0.86640	-73.06588	1581.82241	24323.80399	49.31880	0.00000	17502.65991	0.00058	2.63981
70%	Yes	LSTSQ (gelsy)	1e-16	0.86640	-73.06588	1581.82241	24323.80399	49.31880	0.00000	17502.65991	0.00058	2.14611

- The gelsd driver performs poorly with a condition of 0.1, but improves with lower condition values.
- Data centering with a 70% training set leads to a catastrophic failure in generalization, especially with lower condition values. The 90% set is less affected.
- Lower condition values (e.g., 0.0001, 1e-08, 1e-12, 1e-16) generally provide the best results, with R^2 values close to those of QR and SVD with non-centered data.
- The gelsd driver is generally faster than gelss and gelsy.
- The gelss driver shows similar trends to gelsd regarding the impact of the condition parameter and data centering.
- gelsy is the slowest and does not offer any significant advantage in term of performance over gelsd or gelss.

When we center our data, we obtain similar results of PCA

Example

```python
from sklearn.decomposition import PCA
n_components= 46
pca = PCA(n_components=n_components)
pca.fit(X_train)


#now use PCA to compute theta for the least squares problem
X_train_pca = pca.transform(X_train)
X_test_pca = pca.transform(X_test)
```

```
U, S, Vt = np.linalg.svd(X_train_pca, full_matrices=False)
S = np.diag(S)

theta = Vt.T @ np.linalg.inv(S) @ U.T @ y_train
y_train_pred = X_train_pca @ theta
residuals_train = y_train - y_train_pred
print("Residuals for train set: ",np.linalg.norm(residuals_train,2))
print("Maximum error for train set: ",np.max(np.abs(residuals_train)))
print("Minimum error for train set: ",np.min(np.abs(residuals_train)))
R2_train  = 1 - np.sum((y_train - y_train_pred)**2)/np.sum((y_train -
np.mean(y_train))**2)
print("R2 for train set: ",R2_train)
y_test_pred = X_test_pca @ theta
residuals_test = y_test - y_test_pred
print("Residuals for test set: ",np.linalg.norm(residuals_test,2))
print("Maximum error for test set: ",np.max(np.abs(residuals_test)))
print("Minimum error for test set: ",np.min(np.abs(residuals_test)))
R2_test  = 1 - np.sum((y_test - y_test_pred)**2)/np.sum((y_test -
np.mean(y_test))**2)
print("R2 for test set: ",R2_test)
```

```
Residuals for train set:  1881.996070664596
Maximum error for train set:  50.27252631978616
Minimum error for train set:  0.00018055820469253803
R2 for train set:  0.8108784800073678
Residuals for test set:  1233.9720054889888
Maximum error for test set:  50.60630051087459
Minimum error for test set:  0.0003054529720927235
R2 for test set:  0.8093811530926048
```