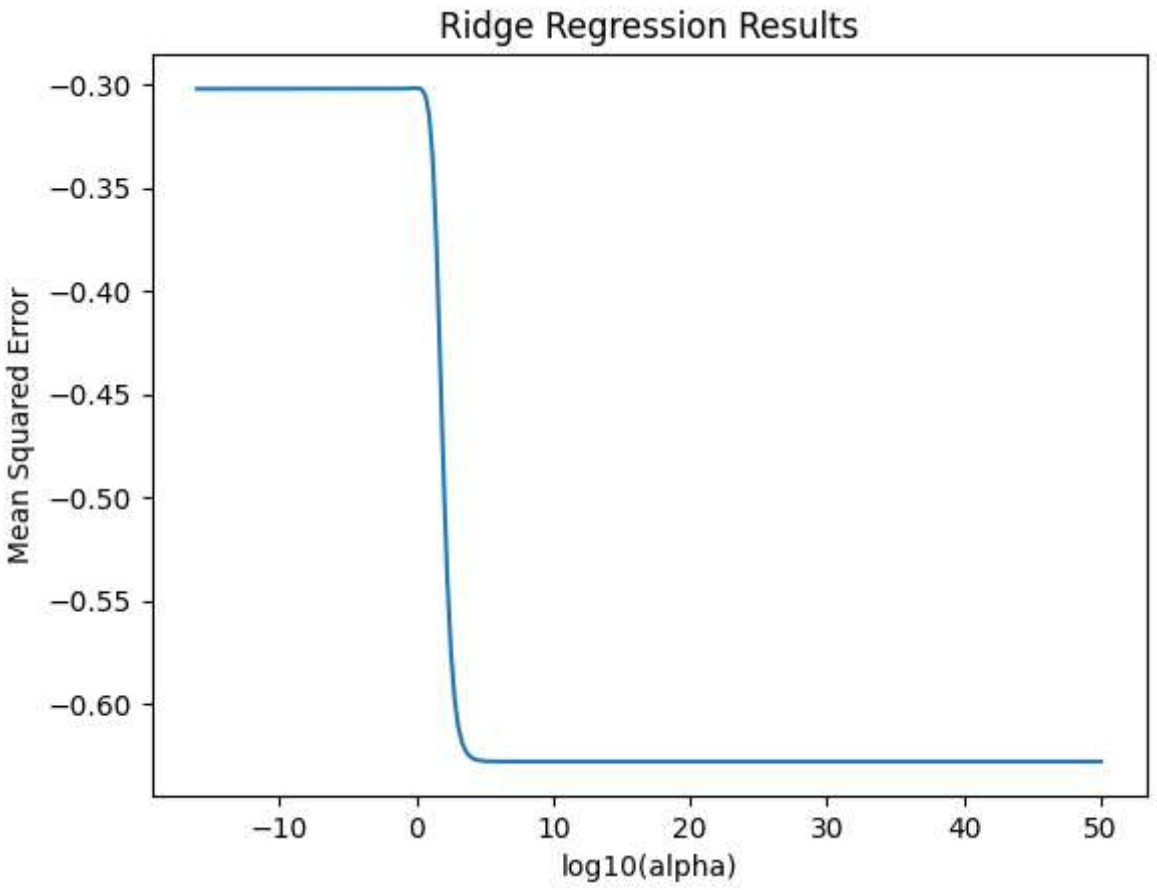
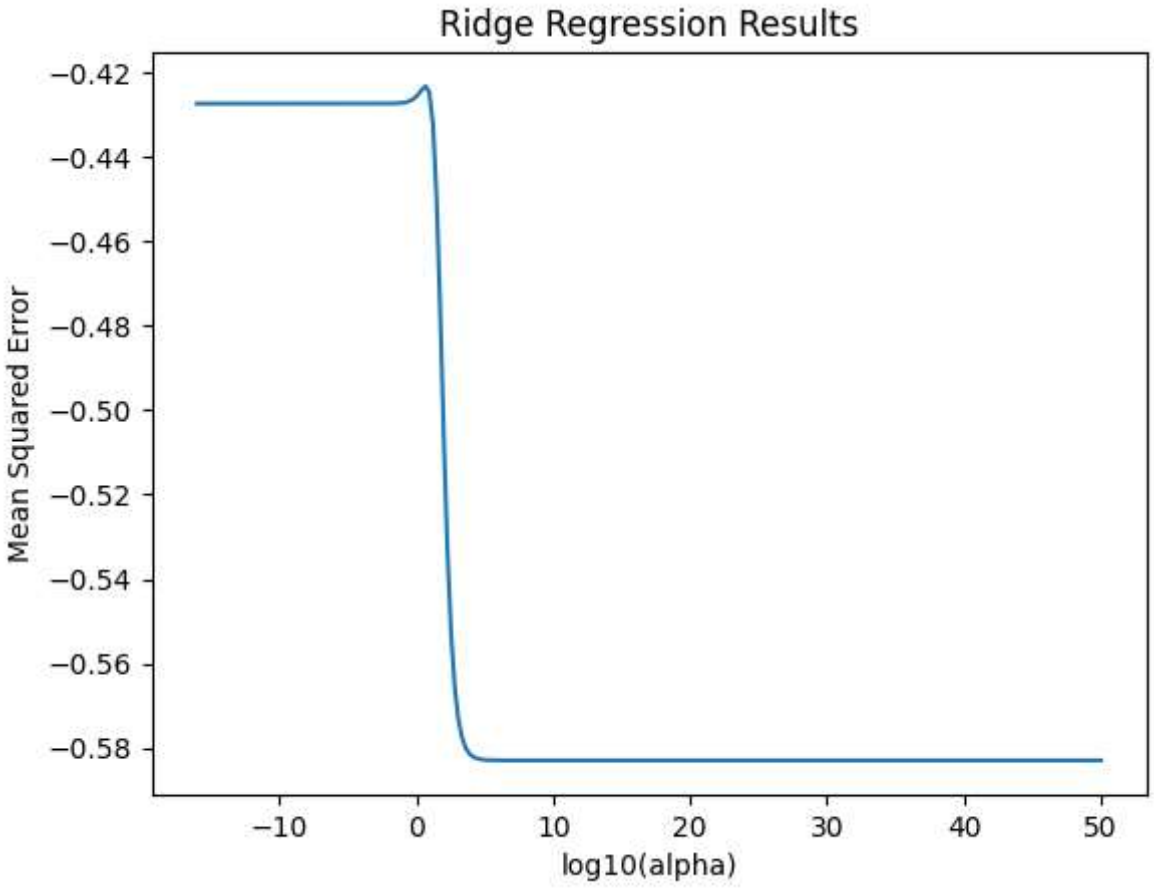
In [ ]:
```
# Name : Niket Ralebhat   Scholar Number : 211112268   Section : 2


import numpy as np
import pandas as pd
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
from sklearn.linear_model import Ridge
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt


for i in range (1 , 6):
    i_str = str(i)
    column = ['Age' , 'Deficit' , 'C_peptide']
    df = pd.read_csv(f'diabetes-5-fold/diabetes-5-{i_str}tra.dat', delimiter=',' ,
    X_train = df.drop("C_peptide" , axis = 1)
    y_train = df.C_peptide

    test = pd.read_csv(f'diabetes-5-fold/diabetes-5-{i_str}tst.dat', delimiter=','
    X_test= test.drop("C_peptide" , axis = 1)
    y_test = test.C_peptide

    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)

    ridge = Ridge()

    param_grid = {'alpha': np.logspace(-16, 50, 250)}

    grid_search = GridSearchCV(ridge, param_grid, cv=5, scoring='neg_mean_squared_e
    grid_search.fit(X_train_scaled, y_train)

    best_alpha = grid_search.best_params_['alpha']

    final_ridge_model = Ridge(alpha=best_alpha)
    final_ridge_model.fit(X_train_scaled, y_train)

    test_predictions = final_ridge_model.predict(X_test_scaled)

    plt.plot(np.log10(param_grid['alpha']), grid_search.cv_results_['mean_test_scor
    plt.xlabel('log10(alpha)')
    plt.ylabel('Mean Squared Error')
    plt.title('Ridge Regression Results')
    plt.show()

    mse = mean_squared_error(y_test, test_predictions)
    mae = mean_absolute_error(y_test  , test_predictions)
    r2 = r2_score(y_test , test_predictions)

    print(f"MSE : {mse}")
    print(f"MAE : {mae}")
```

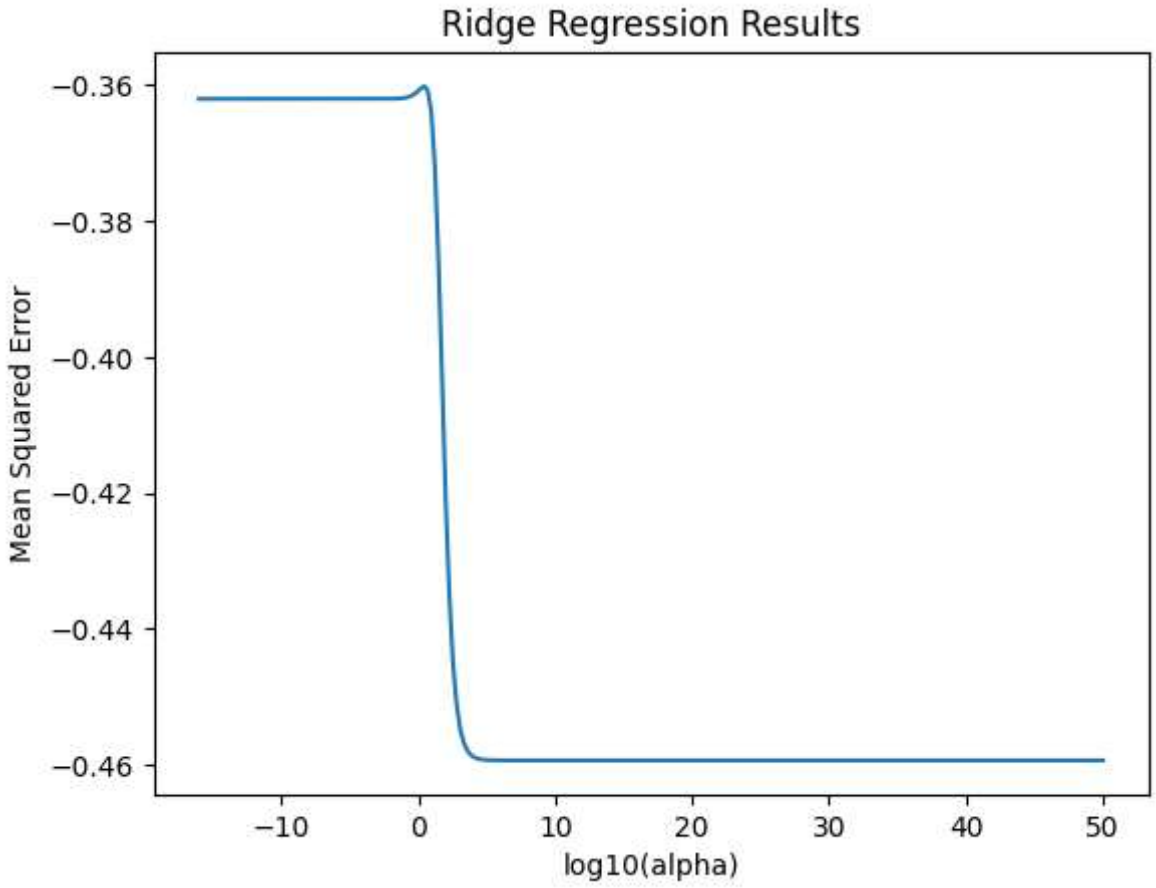
```
    print(f"R2 : {r2}")
    print(f"Best alpha for Grid Search : {grid_search.best_params_["alpha"]}")
```



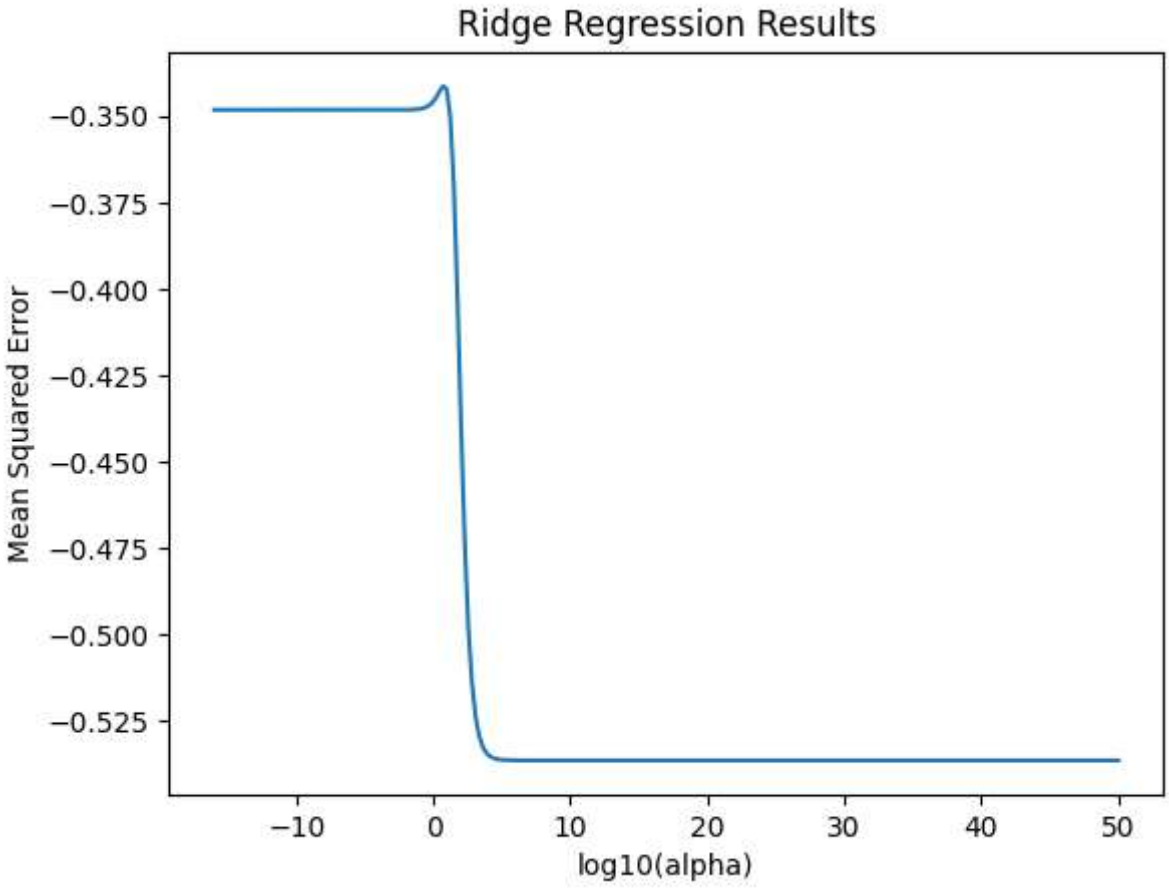

```
MSE : 0.5690949478554852
MAE : 0.6207832693536963
R2 : -0.9699440502689882
Best alpha for Grid Search : 0.8009666945806034
```



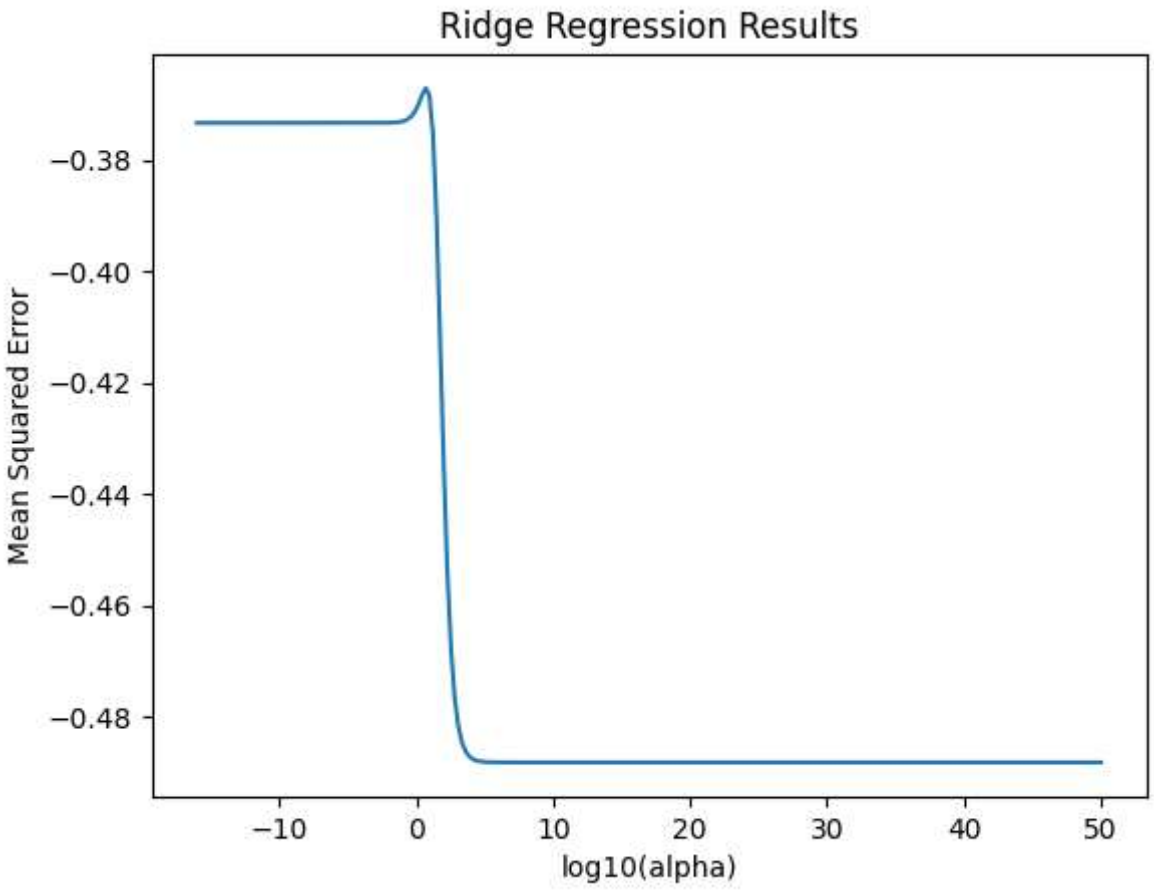

```
MSE : 0.24150081536690882
MAE : 0.4136484829111626
R2 : 0.012042118953555114
Best alpha for Grid Search : 4.9979876738269144
```




```
MSE : 0.40451286663969105
MAE : 0.4656565409642581
R2 : 0.40534406174564475
Best alpha for Grid Search : 2.7147818672390773
```



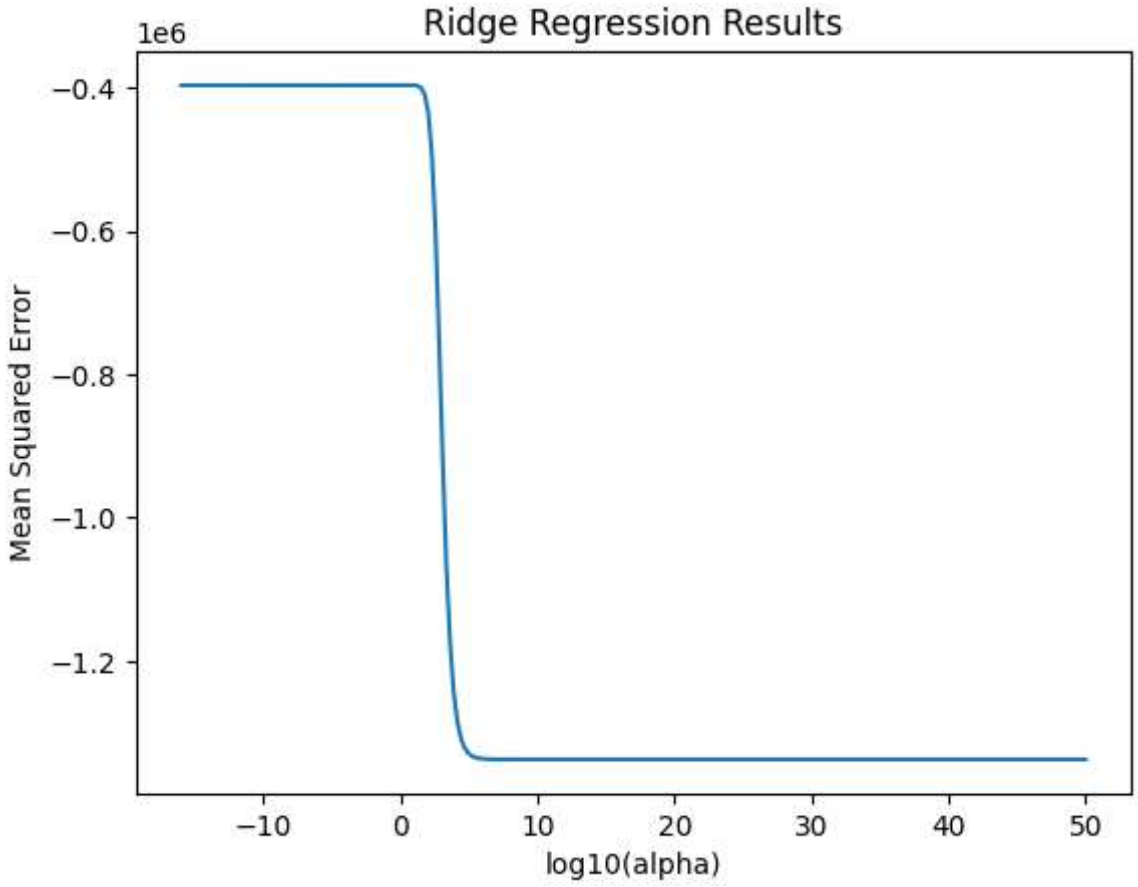

```
MSE : 0.4351850808898524
MAE : 0.5426469623847832
R2 : -0.01946724659409038
Best alpha for Grid Search : 4.9979876738269144
```



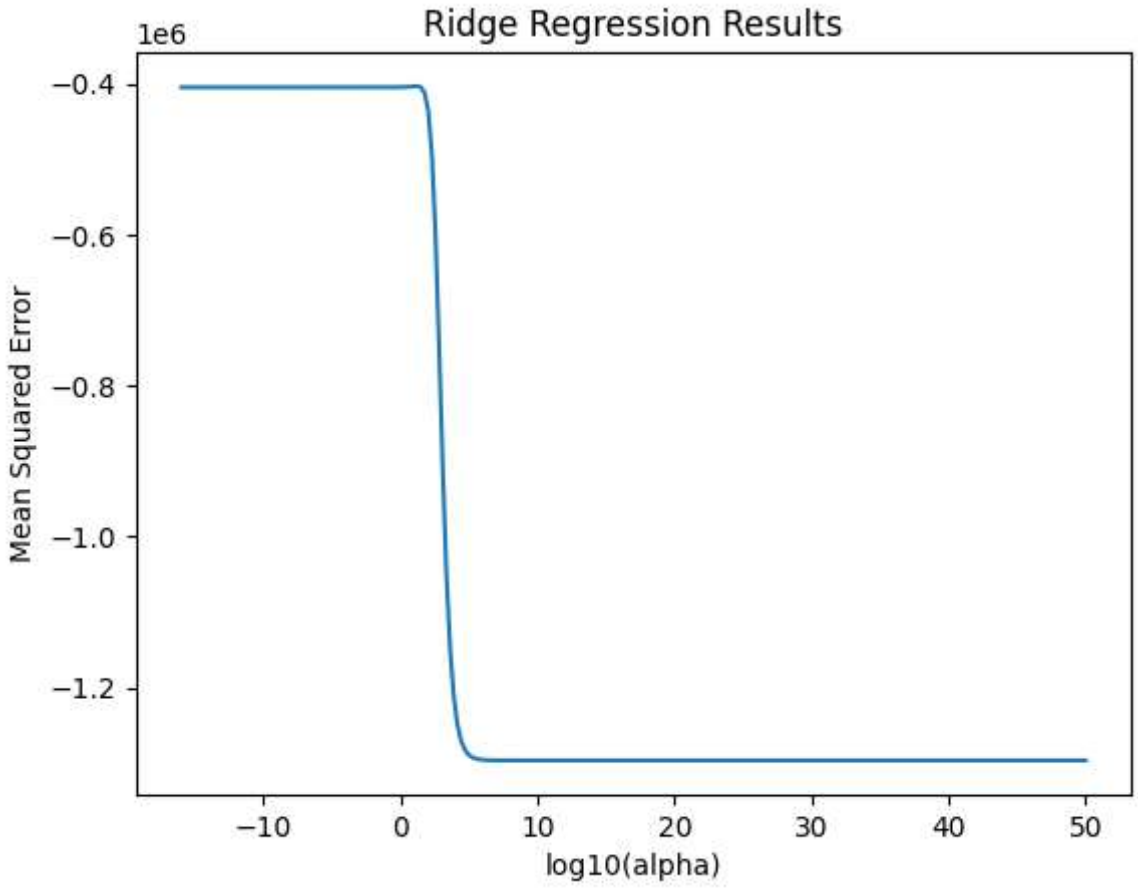

```
MSE : 0.36626641841972285
MAE : 0.4210637178402076
R2 : 0.46917910373953187
Best alpha for Grid Search : 4.9979876738269144
```

In [ ]:
```
for i in range (1 , 6):
    i_str = str(i)
    column = ['Inhabitants' , 'Distance' , 'Length']
    df = pd.read_csv(f'ele-1-5-fold/ele-1-5-{i_str}tra.dat', delimiter=',' , skipro
    X_train = df.drop("Length" , axis = 1)
    y_train = df.Length

    test = pd.read_csv(f'ele-1-5-fold/ele-1-5-{i_str}tst.dat', delimiter=',' , skip
    X_test= test.drop("Length" , axis = 1)
    y_test = test.Length

    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)

    ridge = Ridge()

    param_grid = {'alpha': np.logspace(-16, 50, 250)}

    grid_search = GridSearchCV(ridge, param_grid, cv=5, scoring='neg_mean_squared_e
    grid_search.fit(X_train_scaled, y_train)

    best_alpha = grid_search.best_params_['alpha']
```

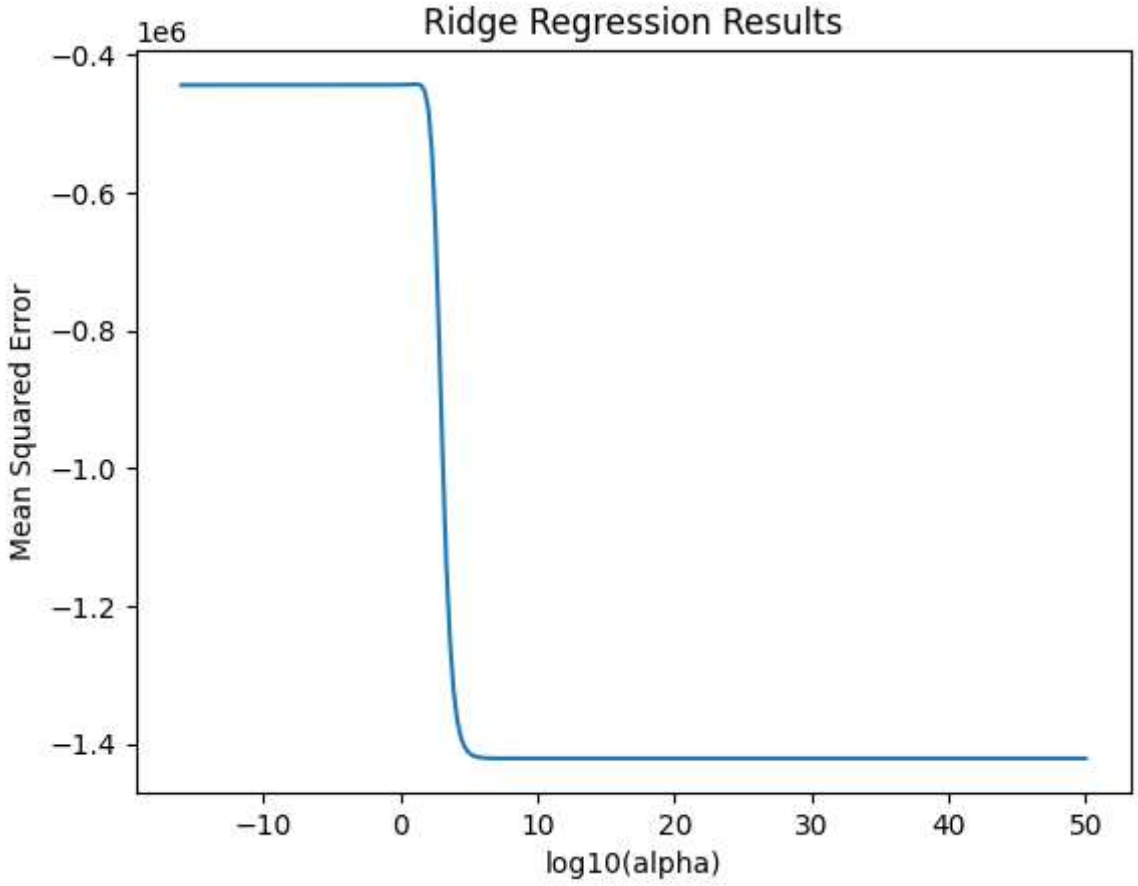
```
final_ridge_model = Ridge(alpha=best_alpha)
final_ridge_model.fit(X_train_scaled, y_train)

test_predictions = final_ridge_model.predict(X_test_scaled)

plt.plot(np.log10(param_grid['alpha']), grid_search.cv_results_['mean_test_scor
plt.xlabel('log10(alpha)')
plt.ylabel('Mean Squared Error')
plt.title('Ridge Regression Results')
plt.show()

mse = mean_squared_error(y_test, test_predictions)
mae = mean_absolute_error(y_test  , test_predictions)
r2 = r2_score(y_test , test_predictions)

print(f"MSE : {mse}")
print(f"MAE : {mae}")
print(f"R2 : {r2}")
print(f"Best alpha for Grid Search : {grid_search.best_params_["alpha"]}")
```



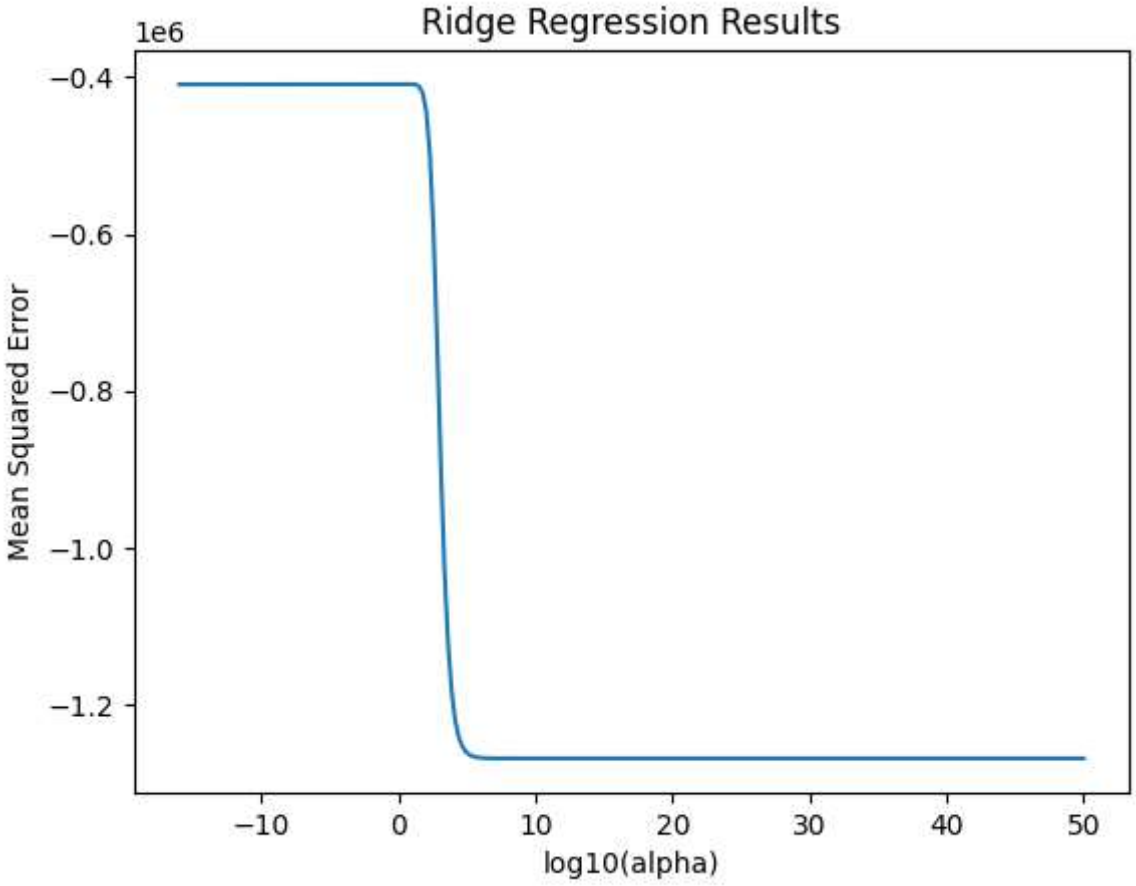

```
MSE : 478000.4406987172
MAE : 429.69189527379683
R2 : 0.6353687179714904
Best alpha for Grid Search : 4.9979876738269144
```



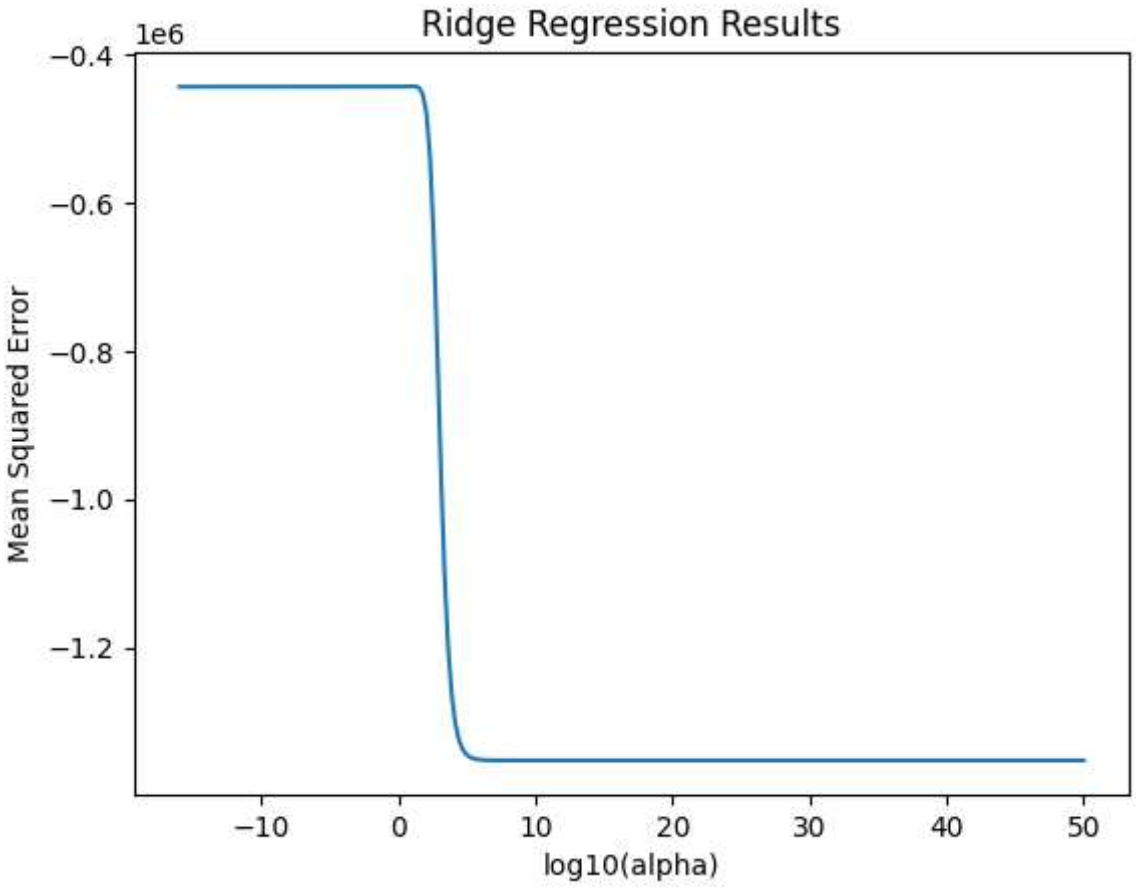

```
MSE : 486604.33508571127
MAE : 430.6547535460312
R2 : 0.6725045969260713
Best alpha for Grid Search : 16.940088022878758
```




```
MSE : 331533.45542865136
MAE : 408.2926062133873
R2 : 0.6641132283605522
Best alpha for Grid Search : 16.940088022878758
```



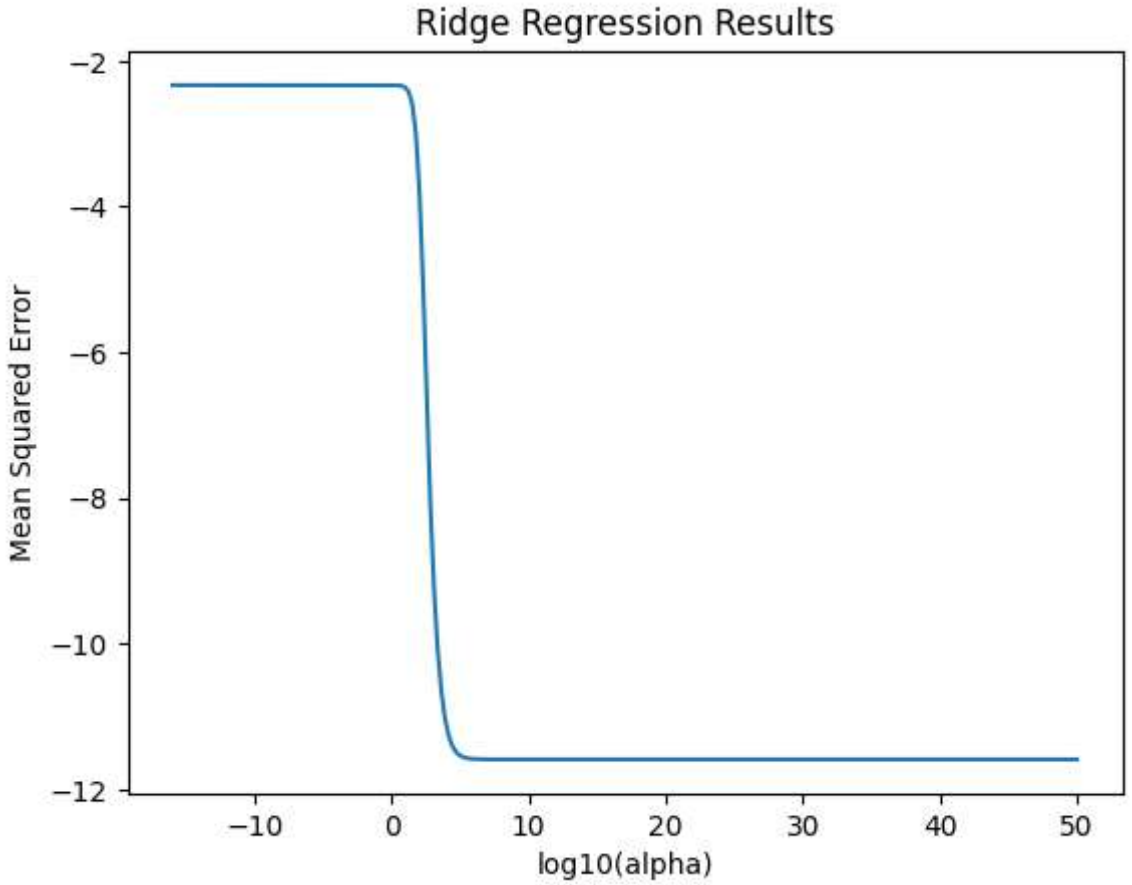

```
MSE : 457088.10840260435
MAE : 427.32311164079374
R2 : 0.7142868589335575
Best alpha for Grid Search : 9.201432015283869
```



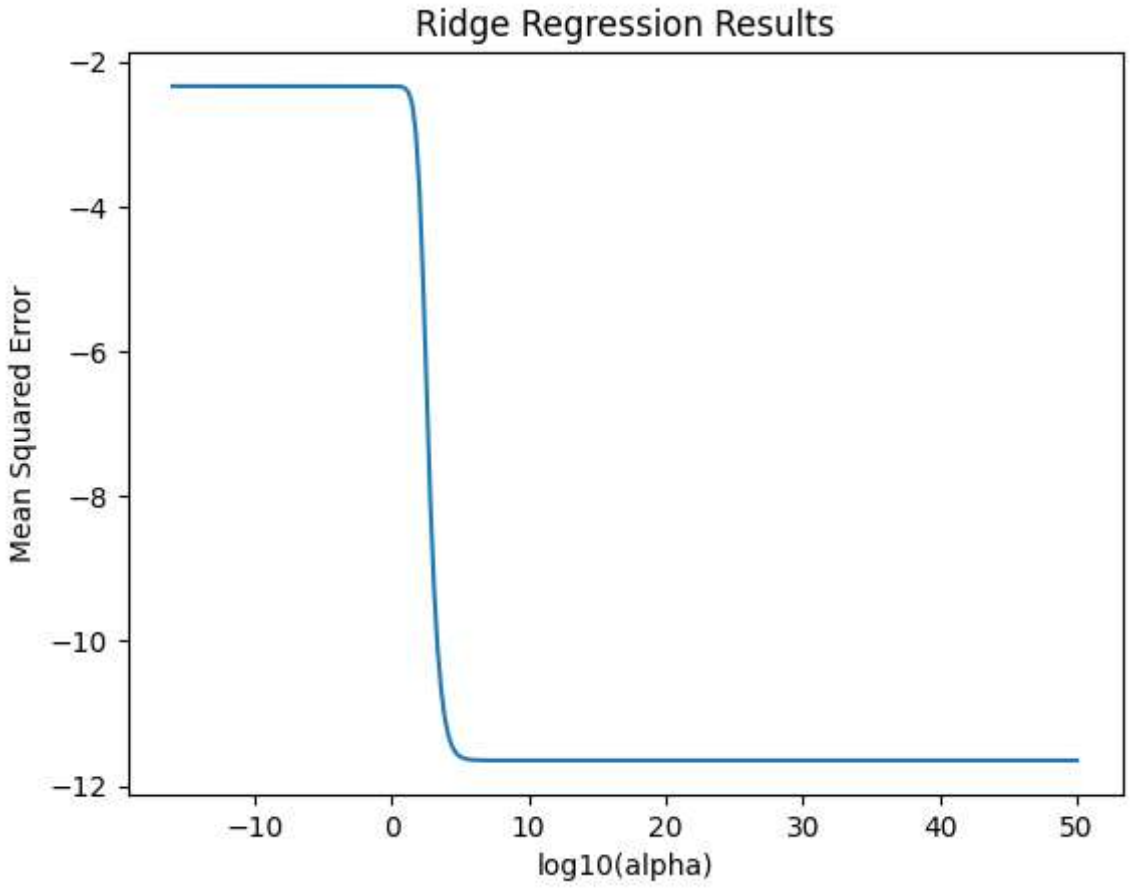

```
MSE : 334015.17399860435
MAE : 402.0231046838057
R2 : 0.7340516130637601
Best alpha for Grid Search : 9.201432015283869
```

In [ ]:
```
for i in range (1 , 6):
    i_str = str(i)
    column = ['Strength' , 'Temperature' , 'Pressure']
    df = pd.read_csv(f'plastic-5-fold/plastic-5-{i_str}tra.dat', delimiter=',' , sk
    X_train = df.drop("Pressure" , axis = 1)
    y_train = df.Pressure

    test = pd.read_csv(f'plastic-5-fold/plastic-5-{i_str}tst.dat', delimiter=',' ,
    X_test= test.drop("Pressure" , axis = 1)
    y_test = test.Pressure

    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)

    ridge = Ridge()

    param_grid = {'alpha': np.logspace(-16, 50, 250)}

    grid_search = GridSearchCV(ridge, param_grid, cv=5, scoring='neg_mean_squared_e
    grid_search.fit(X_train_scaled, y_train)

    best_alpha = grid_search.best_params_['alpha']
```

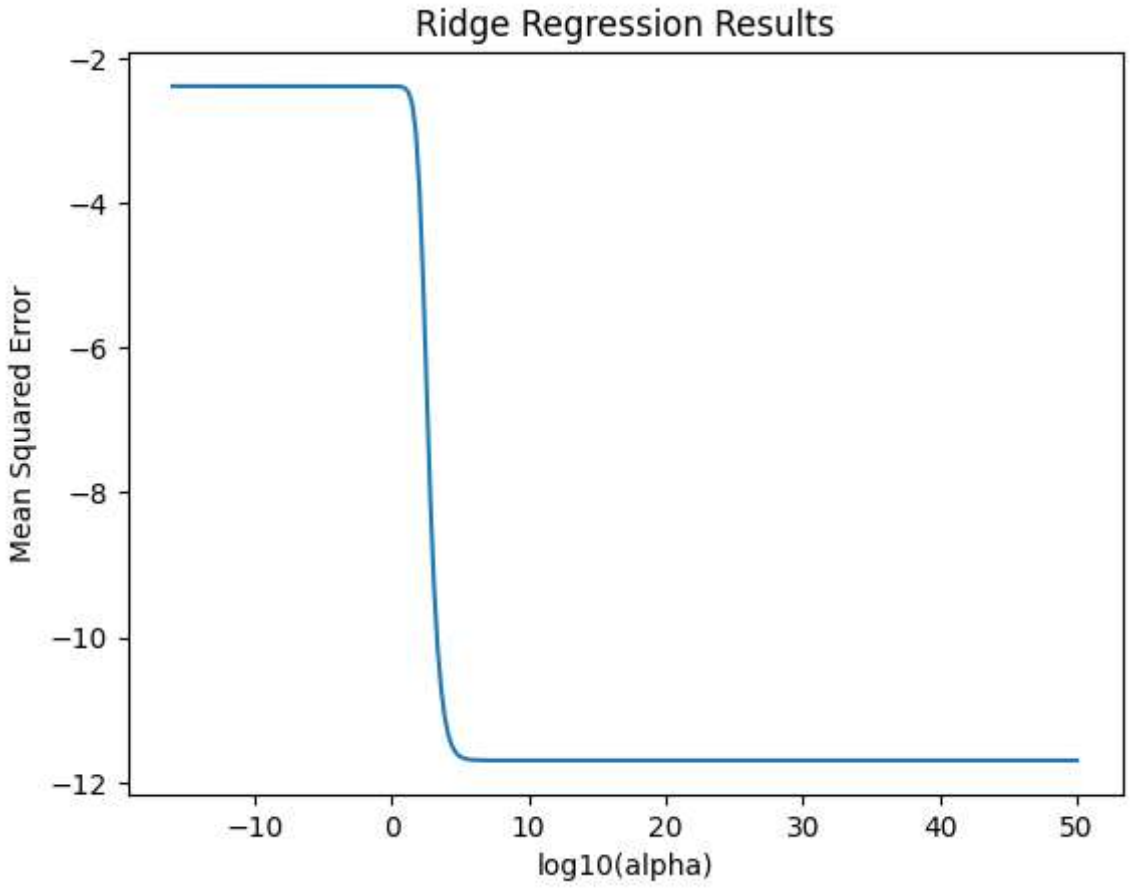
```
final_ridge_model = Ridge(alpha=best_alpha)
final_ridge_model.fit(X_train_scaled, y_train)

test_predictions = final_ridge_model.predict(X_test_scaled)

plt.plot(np.log10(param_grid['alpha']), grid_search.cv_results_['mean_test_scor
plt.xlabel('log10(alpha)')
plt.ylabel('Mean Squared Error')
plt.title('Ridge Regression Results')
plt.show()

mse = mean_squared_error(y_test, test_predictions)
mae = mean_absolute_error(y_test  , test_predictions)
r2 = r2_score(y_test , test_predictions)

print(f"MSE : {mse}")
print(f"MAE : {mae}")
print(f"R2 : {r2}")
print(f"Best alpha for Grid Search : {grid_search.best_params_["alpha"]}")
```



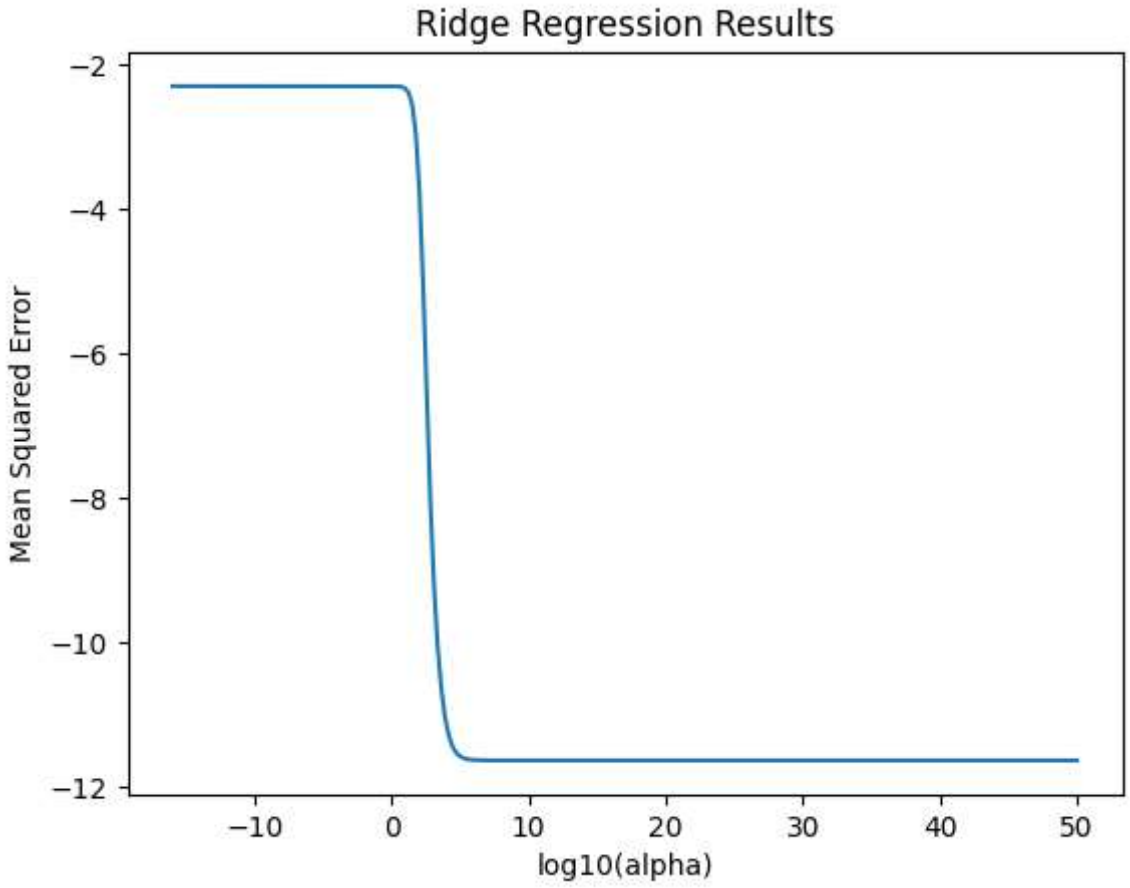

```
MSE : 2.379543115074265
MAE : 1.2430503569615632
R2 : 0.8033205580480629
Best alpha for Grid Search : 0.011173591019485101
```



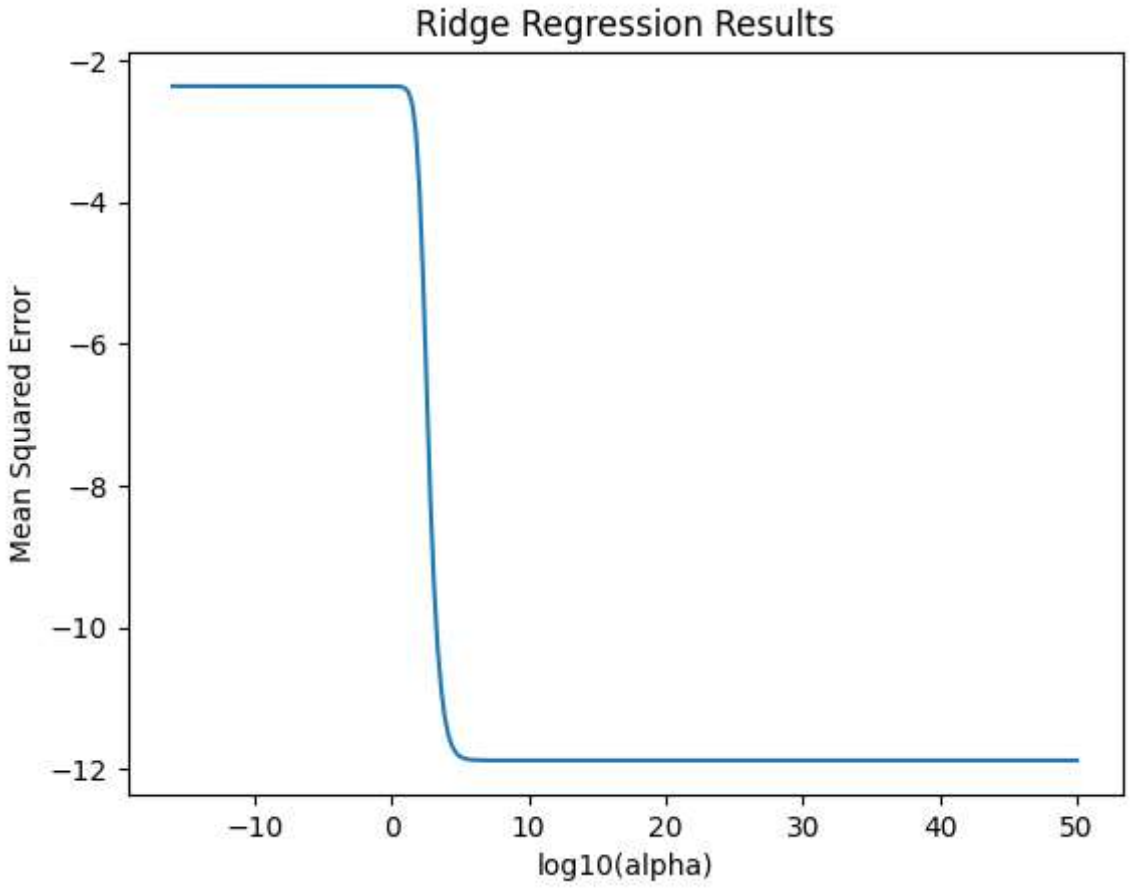

```
MSE : 2.355713556195716
MAE : 1.2190614270481475
R2 : 0.8005581956985022
Best alpha for Grid Search : 0.06972258717575149
```




```
MSE : 2.191511156840035
MAE : 1.195749176305646
R2 : 0.8104999817532533
Best alpha for Grid Search : 1e-16
```




```
MSE : 2.511348243447355
MAE : 1.2698180525725358
R2 : 0.7867689939719807
Best alpha for Grid Search : 0.020570886693214975
```




```
MSE : 2.279607608587684
MAE : 1.2346377421134789
R2 : 0.7904744124191445
Best alpha for Grid Search : 9.460271806598613e-13
```