Name : Niket Ralebhat Scholar Number : 211112268

Section : 2

Lab 1

Importing Libraries :

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline

from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Ridge
```

Ans 1:

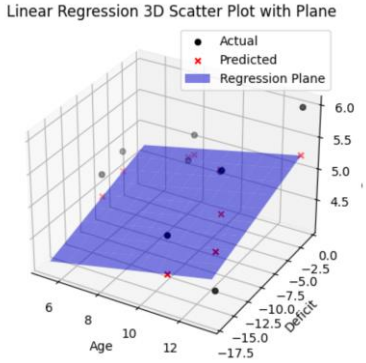
1. Linear Regression :-

```
def solve(i):
    i_str = str(i)
    data1 = pd.read_csv(f'diabetes-5-{i_str}tra.csv')
    data2 = pd.read_csv(f'diabetes-5-{i_str}tst.csv')

    X_train = data1.iloc[:, :-1]
    y_train = data1.iloc[:, -1]

    X_test = data2.iloc[:, :-1]
    y_test = data2.iloc[:, -1]

    model = LinearRegression()

    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)

    Mse = mean_squared_error(y_test, y_pred)
    Mae = mean_absolute_error(y_test, y_pred)
    R2 = r2_score(y_test, y_pred)

    print(f'Mean Squared Error: {Mse}')
    print(f'Mean Absolute Error: {Mae}')
    print(f'R-squared: {R2}')

    mse.append(Mse)
    mae.append(Mae)
    r2.append(R2)
```

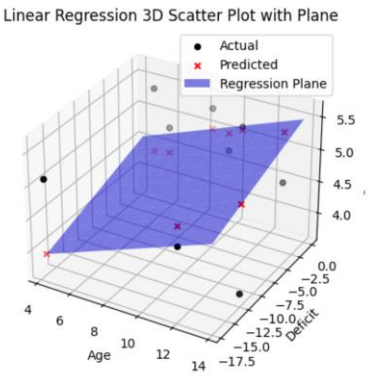
```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

ax.scatter(X_test['Age'], X_test['Deficit'], y_test, c='black', marker='o', label='Actual')
ax.scatter(X_test['Age'], X_test['Deficit'], y_pred, c='red', marker='x', label='Predicted')

x_min, x_max = X_test['Age'].min(), X_test['Age'].max()
y_min, y_max = X_test['Deficit'].min(), X_test['Deficit'].max()
xx, yy = np.meshgrid(np.linspace(x_min, x_max, 10), np.linspace(y_min, y_max, 10))
zz = model.predict(np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape)
ax.plot_surface(xx, yy, zz, alpha=0.5, color='blue', label='Regression Plane')

ax.set_xlabel('Age')
ax.set_ylabel('Deficit')
ax.set_zlabel('C_peptide')
ax.set_title('Linear Regression 3D Scatter Plot with Plane')

ax.legend()
plt.show()
```

```
mse= []
mae =[]
r2 = []

for i in range(5):
    solve(i+1)

avg_mse = 0
avg_mae = 0
avg_r2 = 0
for i in range(5):
    avg_mse += mse[i]
    avg_mae += mae[i]
    avg_r2 += r2[i]

avg_mse /= 5
avg_mae /= 5
avg_r2 /= 5

print(avg_mse)
print(avg_mae)
print(avg_r2)
```

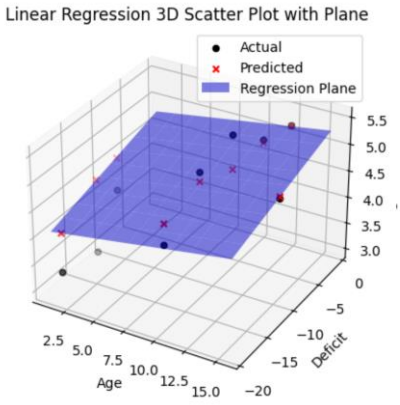
```
Mean Squared Error: 0.38492501173257776
Mean Absolute Error: 0.4505413422282265
R-squared: 0.43413927494847926
```



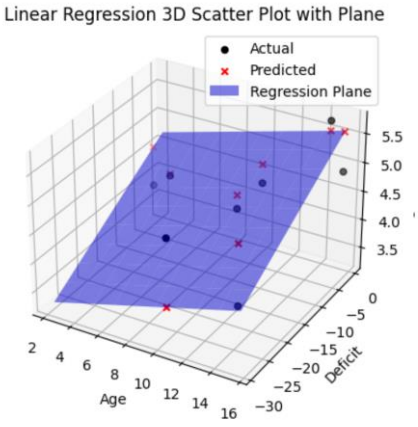

```
Mean Squared Error: 0.23295552546486803
Mean Absolute Error: 0.41407954920495854
R-squared: 0.0470001230982674
```

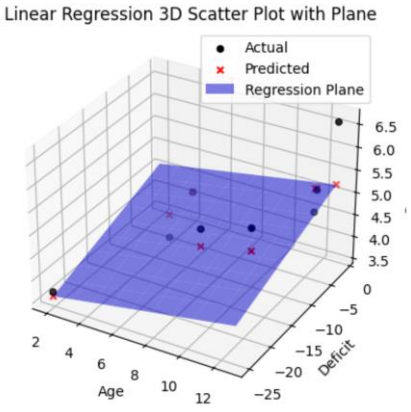


```
Mean Squared Error: 0.4612340443906487
Mean Absolute Error: 0.549037077786952
R-squared: -0.0804897086750187
```




```
Mean Squared Error: 0.3539756074435353
Mean Absolute Error: 0.43210433537735554
R-squared: 0.4869918732702385
```






```
Mean Squared Error: 0.5781152720365408
Mean Absolute Error: 0.6246237320672821
R-squared: -1.0011682493572578
```

```
0.40224109221363413
0.49407720733295485
-0.02270533734305826
```

2. Regression Order 2 & 3 :-

```python
def solve2(i):
    i_str = str(i)
    data1 = pd.read_csv(f'diabetes-5-{i_str}tra.csv')
    data2 = pd.read_csv(f'diabetes-5-{i_str}tst.csv')

    X_train = data1[['Age', 'Deficit']]
    y_train = data1['C_peptide']

    X_test = data2[['Age', 'Deficit']]
    y_test = data2['C_peptide']
```

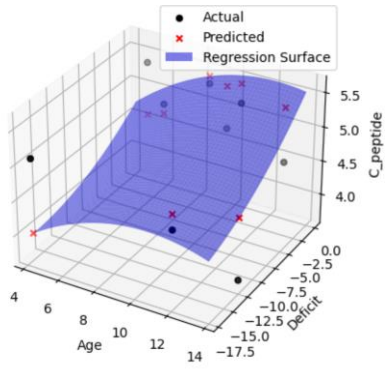
```
model = make_pipeline(PolynomialFeatures(degree = 2), LinearRegression())
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

Mse = mean_squared_error(y_test, y_pred)
Mae = mean_absolute_error(y_test, y_pred)
R2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {Mse}')
print(f'Mean Absolute Error: {Mae}')
print(f'R-squared: {R2}')

mse_2.append(Mse)
mae_2.append(Mae)
r2_2.append(R2)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

ax.scatter(X_test['Age'], X_test['Deficit'], y_test, c='black', marker='o', label='Actual')
ax.scatter(X_test['Age'], X_test['Deficit'], y_pred, c='red', marker='x', label='Predicted')

x_min, x_max = X_test['Age'].min(), X_test['Age'].max()
y_min, y_max = X_test['Deficit'].min(), X_test['Deficit'].max()
xx, yy = np.meshgrid(np.linspace(x_min, x_max, 50), np.linspace(y_min, y_max, 50))
zz = model.predict(np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape)
ax.plot_surface(xx, yy, zz, alpha=0.5, color='blue', label='Regression Surface')

ax.set_xlabel('Age')
ax.set_ylabel('Deficit')
ax.set_zlabel('C_peptide')
ax.set_title('Polynomial Regression 3D Scatter Plot with Surface (Order 2)')

ax.legend()
plt.show()
```

```
mse_2= []
mae_2 =[]
r2_2 = []

for i in range(5):
    solve2(i+1)

avg_mse_2 = 0
avg_mae_2 = 0
avg_r2_2 = 0
for i in range(5):
    avg_mse_2 += mse_2[i]
    avg_mae_2 += mae_2[i]
    avg_r2_2 += r2_2[i]

avg_mse_2 /= 5
avg_mae_2 /= 5
avg_r2_2 /= 5

print(avg_mse_2)
print(avg_mae_2)
print(avg_r2_2)
```

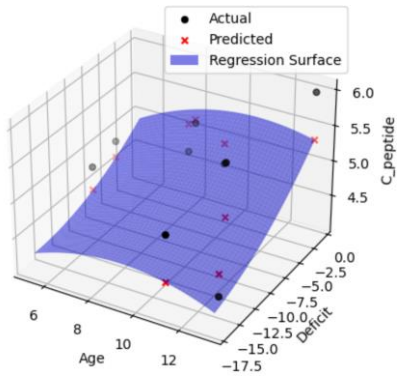
```
def solve2(i):
    i_str = str(i)
    data1 = pd.read_csv(f'diabetes-5-{i_str}tra.csv')
    data2 = pd.read_csv(f'diabetes-5-{i_str}tst.csv')

    X_train = data1[['Age', 'Deficit']]
    y_train = data1['C_peptide']

    X_test = data2[['Age', 'Deficit']]
    y_test = data2['C_peptide']

    model = make_pipeline(PolynomialFeatures(degree = 3), LinearRegression())
    model.fit(X_train, y_train)
```

```
y_pred = model.predict(X_test)

Mse = mean_squared_error(y_test, y_pred)
Mae = mean_absolute_error(y_test, y_pred)
R2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {Mse}')
print(f'Mean Absolute Error: {Mae}')
print(f'R-squared: {R2}')

mse_2.append(Mse)
mae_2.append(Mae)
r2_2.append(R2)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

ax.scatter(X_test['Age'], X_test['Deficit'], y_test, c='black', marker='o', label='Actual')
ax.scatter(X_test['Age'], X_test['Deficit'], y_pred, c='red', marker='x', label='Predicted')

x_min, x_max = X_test['Age'].min(), X_test['Age'].max()
y_min, y_max = X_test['Deficit'].min(), X_test['Deficit'].max()
xx, yy = np.meshgrid(np.linspace(x_min, x_max, 50), np.linspace(y_min, y_max, 50))
zz = model.predict(np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape)
ax.plot_surface(xx, yy, zz, alpha=0.5, color='blue', label='Regression Surface')

ax.set_xlabel('Age')
ax.set_ylabel('Deficit')
ax.set_zlabel('C_peptide')
ax.set_title('Polynomial Regression 3D Scatter Plot with Surface (Order 2)')

ax.legend()
plt.show()
```

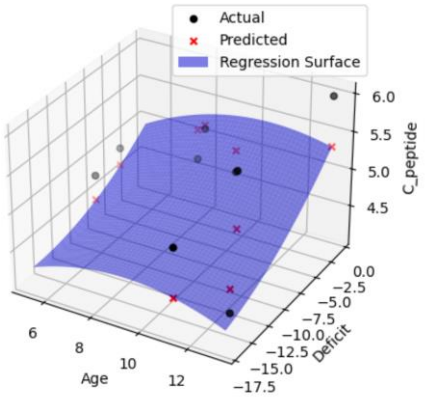
Mean Squared Error: 0.418269577034059
Mean Absolute Error: 0.5478504512588268
R-squared: -0.4478562281948204



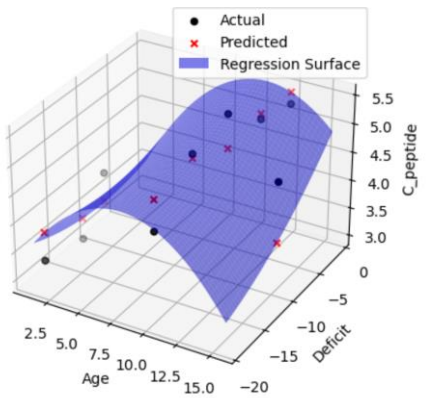
Mean Squared Error: 0.2195858811506907
Mean Absolute Error: 0.4126001819737721
R-squared: 0.10169412256535648



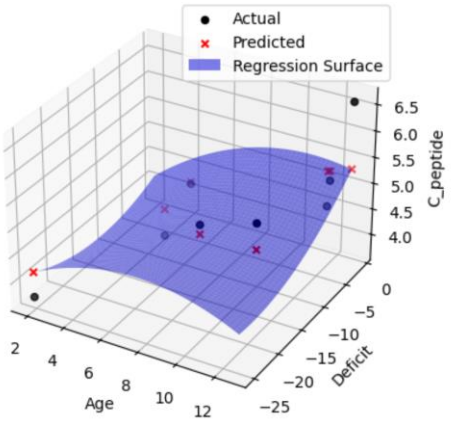
Mean Squared Error: 0.27671280845498347
Mean Absolute Error: 0.4419479779582142
R-squared: 0.5932171055380462



Mean Squared Error: 0.24375312437204583
Mean Absolute Error: 0.402301375976717
R-squared: 0.42898243192492935



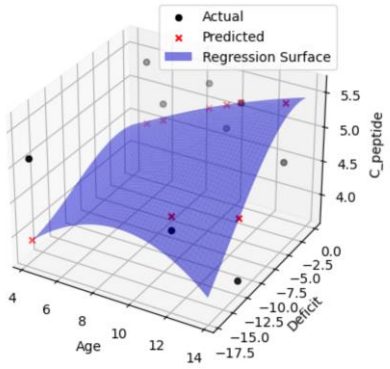
Mean Squared Error: 0.36167311669288477
Mean Absolute Error: 0.4816623700533394
R-squared: 0.47583606276393486



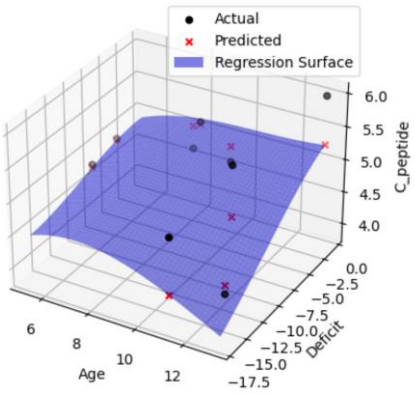
0.30399890154093273
0.45727247144417393
0.2303746989194893

Order 3 :-

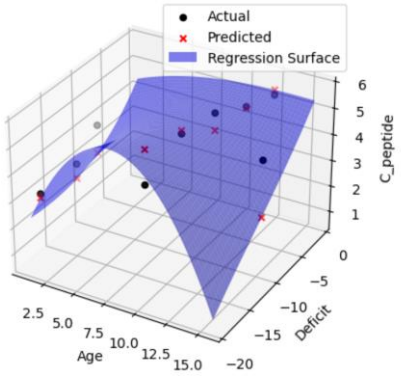
Mean Squared Error: 0.45619220866059884
Mean Absolute Error: 0.5551429193974313
R-squared: -0.5791268761328427



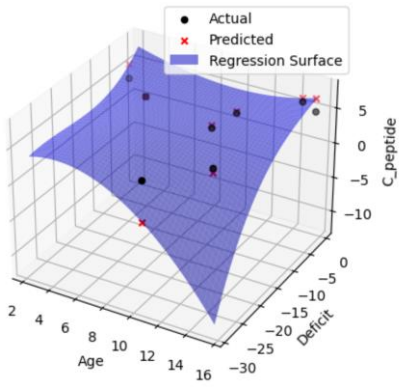
Mean Squared Error: 0.23162739202245572
Mean Absolute Error: 0.3550265503478813
R-squared: 0.05243339627177235

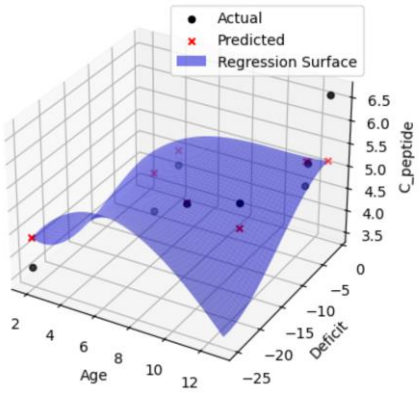


Mean Squared Error: 0.9106226956545892
Mean Absolute Error: 0.6942176152711146
R-squared: -0.33866494279531256



Mean Squared Error: 5.192339743355129
Mean Absolute Error: 1.4261986701731457
R-squared: -11.163607012252129






```
Mean Squared Error: 0.4918342300781183
Mean Absolute Error: 0.5566889322604411
R-squared: 0.28719676800272687
```

```
1.4565232539541781
0.7174549374900028
-2.348353733381157
```

3. Regression with ridge regularization :-

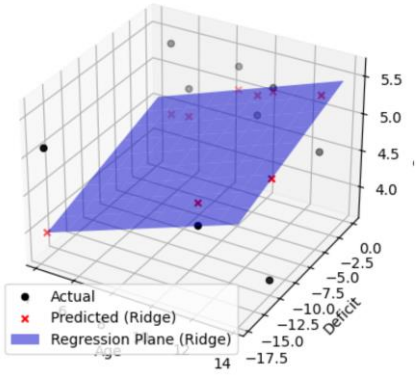
```python
def solve3(i):
    i_str = str(i)
    data1 = pd.read_csv(f'diabetes-5-{i_str}tra.csv')
    data2 = pd.read_csv(f'diabetes-5-{i_str}tst.csv')

    X_train = data1.iloc[:, :-1]
    y_train = data1.iloc[:, -1]

    X_test = data2.iloc[:, :-1]
    y_test = data2.iloc[:, -1]

    ridge_model = Ridge()

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

    grid_search = GridSearchCV(ridge_model, param_grid, cv=5, scoring='neg_mean_squared_error')
    grid_search.fit(X_train, y_train)

    best_ridge_model = grid_search.best_estimator_

    y_pred_ridge = best_ridge_model.predict(X_test)

    Mse_3 = mean_squared_error(y_test, y_pred_ridge)
    Mae_3 = mean_absolute_error(y_test, y_pred_ridge)
    R2_3 = r2_score(y_test, y_pred_ridge)
```

```python
print(f'Mean Squared Error (Ridge): {Mse_3}')
print(f'Mean Absolute Error (Ridge): {Mae_3}')
print(f'R-squared (Ridge): {R2_3}')
print(f'Best alpha from grid search: {grid_search.best_params_["alpha"]}')

mse_2.append(Mse_3)
mae_2.append(Mae_3)
r2_2.append(R2_3)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

ax.scatter(X_test['Age'], X_test['Deficit'], y_test, c='black', marker='o', label='Actual')
ax.scatter(X_test['Age'], X_test['Deficit'], y_pred_ridge, c='red', marker='x', label='Predicted (Ridge)')
```

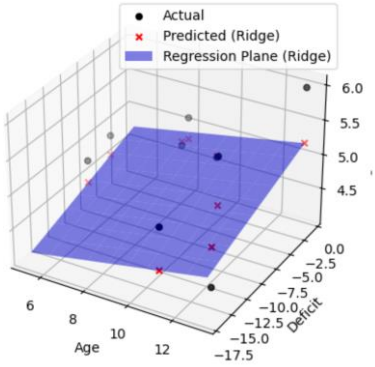
```python
x_min, x_max = X_test['Age'].min(), X_test['Age'].max()
y_min, y_max = X_test['Deficit'].min(), X_test['Deficit'].max()
xx, yy = np.meshgrid(np.linspace(x_min, x_max, 10), np.linspace(y_min, y_max, 10))
zz_ridge = best_ridge_model.predict(np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape)
ax.plot_surface(xx, yy, zz_ridge, alpha=0.5, color='blue', label='Regression Plane (Ridge)')

ax.set_xlabel('Age')
ax.set_ylabel('Deficit')
ax.set_zlabel('C_peptide')
ax.set_title('Ridge Regression 3D Scatter Plot with Plane')

ax.legend()
plt.show()
```

```python
mse_3= []
mae_3 =[]
r2_3 = []

for i in range(5):
    solve3(i+1)

avg_mse_3 = 0
avg_mae_3 = 0
avg_r2_3 = 0
for i in range(5):
    avg_mse_3 += mse_3[i]
    avg_mae_3 += mae_3[i]
    avg_r2_3 += r2_3[i]

avg_mse_3 /= 5
avg_mae_3 /= 5
avg_r2_3 /= 5

print(avg_mse_3)
print(avg_mae_3)
print(avg_r2_3)
```

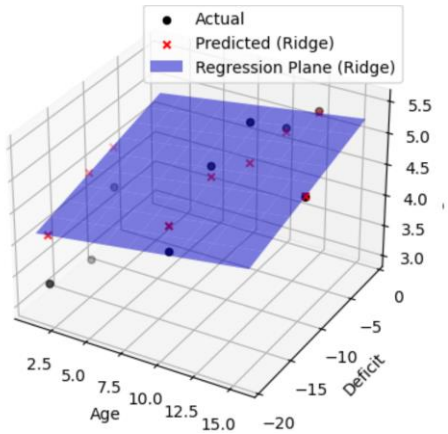
Mean Squared Error (Ridge): 0.5562409760631762
Mean Absolute Error (Ridge): 0.6131470503719512
R-squared (Ridge): -0.9254495325263801
Best alpha from grid search: 31.187165404930422



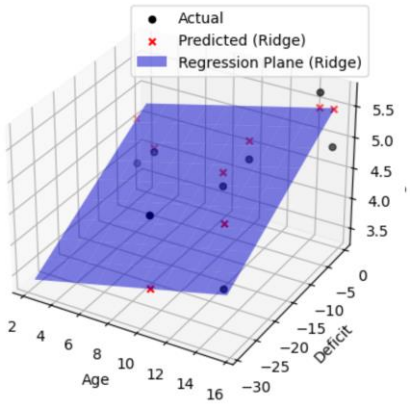
Mean Squared Error (Ridge): 0.23717048288022896
Mean Absolute Error (Ridge): 0.4113385792266906
R-squared (Ridge): 0.02975711548997273
Best alpha from grid search: 105.70520810009836

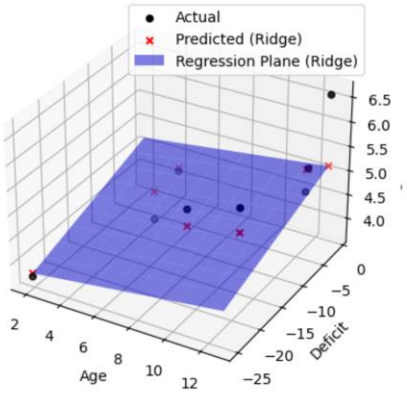


Mean Squared Error (Ridge): 0.40836586900536354
Mean Absolute Error (Ridge): 0.46415419445695794
R-squared (Ridge): 0.3996799384857632
Best alpha from grid search: 57.41642455935722



Mean Squared Error (Ridge): 0.44998502615149927
Mean Absolute Error (Ridge): 0.5506447506481741
R-squared (Ridge): -0.05413768937393648
Best alpha from grid search: 105.70520810009836



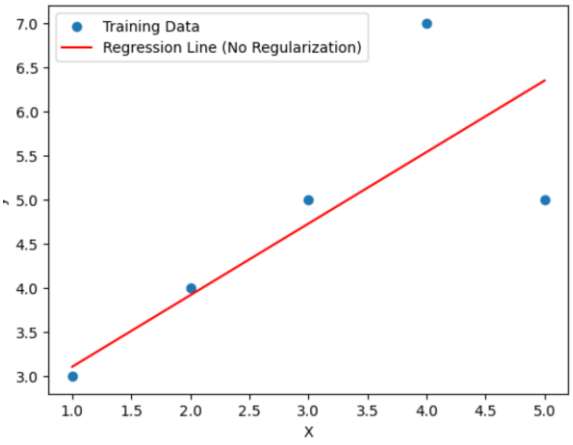


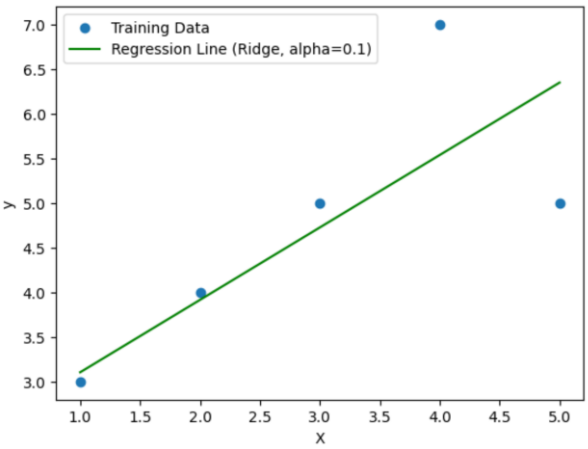

```
Mean Squared Error (Ridge): 0.3797174833271306
Mean Absolute Error (Ridge): 0.43899191887167477
R-squared (Ridge): 0.4496848067722743
Best alpha from grid search: 194.60617942055623

0.4062959674854797
0.49565529871508973
-0.02009307223046126
```

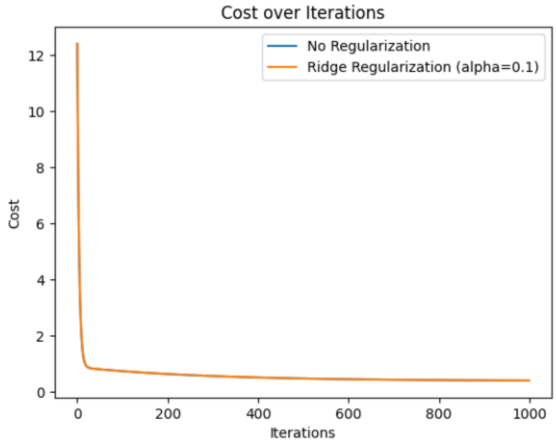
Ans 2:

```
X = np.array([1, 2, 3, 4, 5]).reshape(-1, 1)
y = np.array([3, 4, 5, 7, 5]).reshape(-1, 1)

X_b = np.c_[np.ones((X.shape[0], 1)), X]

learning_rate = 0.01
num_iterations = 1000
ridge_alpha = 0.1

theta = np.zeros((X_b.shape[1], 1))
history_cost = []

for _ in range(num_iterations):
    errors = np.dot(X_b, theta) - y
    gradients = (np.dot(X_b.T, errors) + ridge_alpha * np.vstack((0, theta[1:]))) / len(y)
    theta -= learning_rate * gradients

    cost = np.sum(errors**2) / (2 * len(y))
    history_cost.append(cost)

plt.scatter(X, y, label='Training Data')
plt.plot(X, np.dot(X_b, theta), color='red', label='Regression Line (No Regularization)')
plt.xlabel('X')
plt.ylabel('y')
plt.legend()
plt.show()
```

```
theta_ridge = np.zeros((X_b.shape[1], 1))
history_cost_ridge = []

for _ in range(num_iterations):
    errors_ridge = np.dot(X_b, theta_ridge) - y
    gradients_ridge = (np.dot(X_b.T, errors_ridge) + ridge_alpha * np.vstack((0, theta_ridge[1:]))) / len(y)
    theta_ridge -= learning_rate * gradients_ridge

    cost_ridge = np.sum(errors_ridge**2) / (2 * len(y))
    history_cost_ridge.append(cost_ridge)

plt.scatter(X, y, label='Training Data')
plt.plot(X, np.dot(X_b, theta_ridge), color='green', label=f'Regression Line (Ridge, alpha={ridge_alpha})')
plt.xlabel('X')
plt.ylabel('y')
plt.legend()
plt.show()

plt.plot(range(num_iterations), history_cost, label='No Regularization')
plt.plot(range(num_iterations), history_cost_ridge, label=f'Ridge Regularization (alpha={ridge_alpha})')
plt.xlabel('Iterations')
plt.ylabel('Cost')
plt.title('Cost over Iterations')
plt.legend()
plt.show()
```


Training Data
Regression Line (No Regularization)
X


Training Data
Regression Line (Ridge, alpha=0.1)
y
X


Cost over Iterations
No Regularization
Ridge Regularization (alpha=0.1)
Cost
Iterations