Question 3:

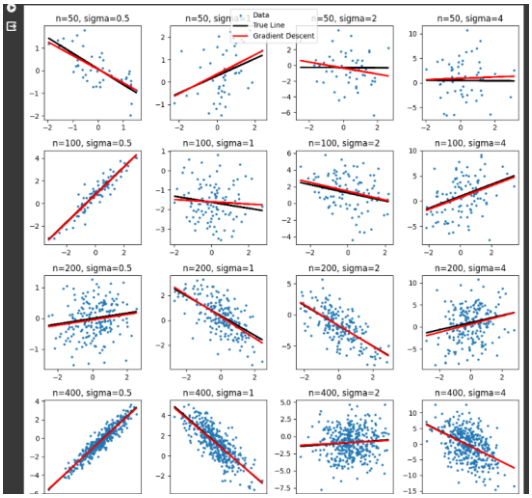
Create a report investigating how different values of n and σ impact the ability for your linear regression function to learn the coefficients, β, used to generate the output vector Y.

Answer:

(check uploaded colab for code)

- **Dataset Creation**:
  - We generate datasets with varying levels of noise (sigma) and different sizes (n).
  - These datasets are used for training a model.
- **Linear Regression Model**:
  - We apply linear regression using gradient descent to the generated datasets.
  - The goal is to find a linear relationship between input features (X) and output labels (Y).
- **Visualization**:
  - The code creates a grid of subplots, where each subplot corresponds to a combination of n and sigma.
  - For each combination, it:
    - Scatters the data points.
    - Plots the true linear relationship (black line) and the regression line obtained through gradient descent (red line).
    - Sets the subplot title with the specific n and sigma values.
- **Legend**:
  - A common legend is added to all subplots, explaining the data points, true line, and gradient descent line.

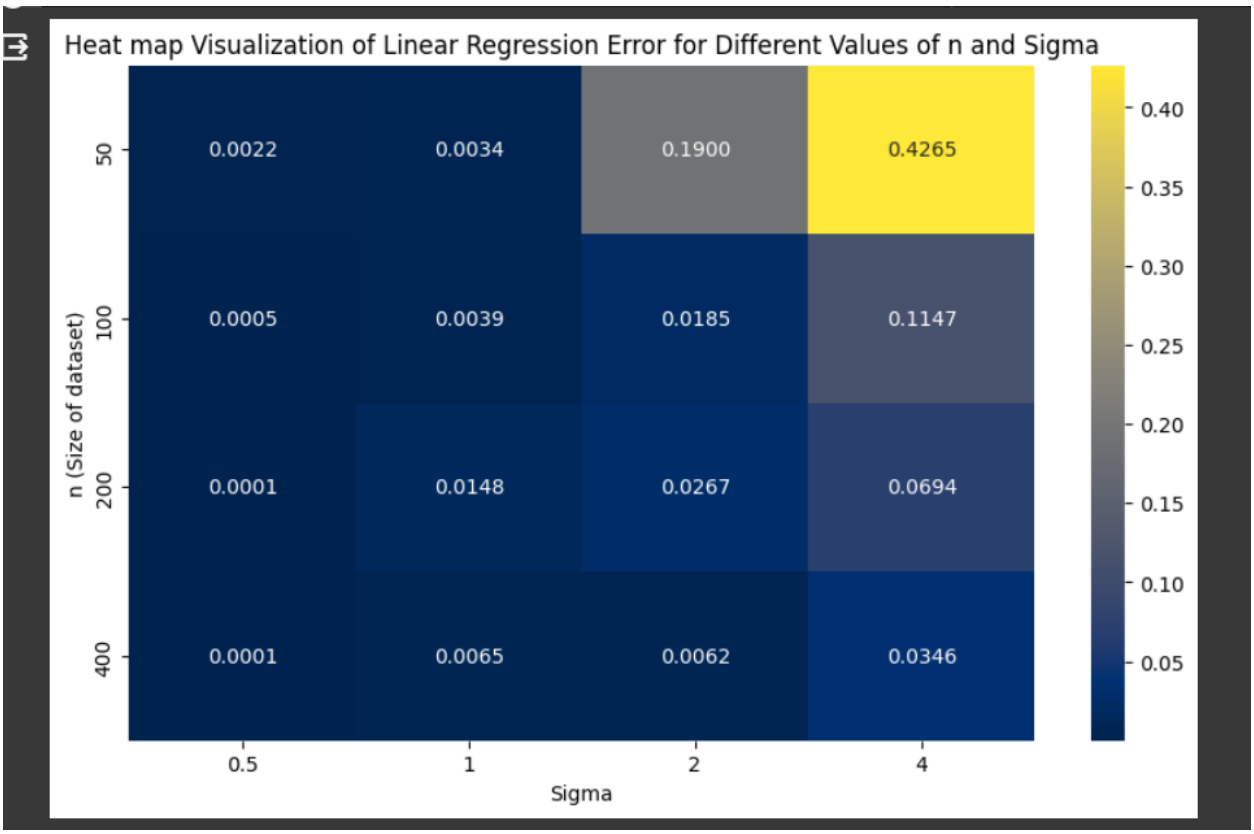
```
# Define values for dataset size (n) and noise level (sigma)
n_values = [50, 100, 200, 400]
sigma_values = [0.5, 1, 2, 4]

# Create subplots grid with specified dimensions
fig, axs = plt.subplots(len(n_values), len(sigma_values), figsize=(10, 10))

# Iterate over each combination of n and sigma
for i, n in enumerate(n_values):
    for j, sigma in enumerate(sigma_values):
        #Generate dataset and apply linear regression
        X, Y, true_beta = generate_dataset(sigma, n, m)
        beta_gd, final_cost = linear_regression_gradient_descent(X, Y, kappa,tau,lamda)

        # Scatter plot with true line and regression line
        axs[i, j].scatter(X, Y, label='Data', s=5)
        axs[i, j].plot(X, np.dot(np.hstack((np.ones((X.shape[0], 1)), X)), true_beta), color='black', label='True Line', linewidth=2)
        axs[i, j].plot(X, np.dot(np.hstack((np.ones((X.shape[0], 1)), X)), beta_gd), color='red', label='Gradient Descent', linewidth=2)

        # Set subplot title with n and sigma values
        axs[i, j].set_title(f'n={n}, sigma={sigma}')

# Add a common legend for all subplots
handles, labels = axs[0, 0].get_legend_handles_labels()
fig.legend(handles, labels, loc='upper center')

# Adjust layout for better visualization and plot
plt.tight_layout()
plt.show()
```



Heatmap Visualization

- **Dataset Generation**:
  - For each combination of n and sigma, it generates a dataset using the generate_dataset function.
  - The dataset includes input features (X), output labels (Y), and the true regression coefficients (true_beta).
- **Linear Regression Model**:
  - It applies linear regression using gradient descent (linear_regression_gradient_descent) to estimate the regression coefficients (beta_gd).
  - The goal is to find the best-fitting linear relationship between X and Y.
- **Error Calculation**:
  - The mean squared error (MSE) between the true coefficients and the estimated coefficients is computed.
  - This error quantifies how well the model fits the data.
- **Heatmap Visualization**:
  - The errors matrix stores the computed MSE values for different combinations of n and sigma.
  - The heatmap is created using Seaborn (sns.heatmap), where:
    - Rows represent different dataset sizes (n).
    - Columns represent different noise levels (sigma).
    - Each cell color corresponds to the MSE value.
    - Annotations show the exact MSE values.
- **Interpretation**:
  - Darker colors indicate higher errors, suggesting poorer model performance.
  - The heatmap helps visualize how changing n and sigma affects the accuracy of the linear regression model.

```python
# HEAT MAP VISUALIZATION

n_values = [50, 100, 200, 400]
sigma_values = [0.5, 1, 2, 4]
errors = np.zeros((len(n_values), len(sigma_values)))

for i, n_val in enumerate(n_values):
    for j, sigma_val in enumerate(sigma_values):
        X_val, Y_val, true_beta_val = generate_dataset(sigma_val, n_val, m)
        beta_gd_val, final_cost_val = linear_regression_gradient_descent(X_val, Y_val, kappa, tau, lamda)
        error = np.mean((true_beta_val - beta_gd_val) ** 2)
        errors[i, j] = error

# Create a formatted heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(errors, annot=True, fmt=".4f", xticklabels=sigma_values, yticklabels=n_values, cmap='cividis')
plt.xlabel('Sigma')
plt.ylabel('n (Size of dataset)')
plt.title('Heat map Visualization of Linear Regression Error for Different Values of n and Sigma')
plt.show()
```



Result:

- As the dataset size (n) increases, there is a noticeable decrease in the linear regression error. This reduction in error is consistent across all Sigma values, indicating that larger sample sizes contribute to the development of more accurate linear regression models.
- With a fixed n, an increase in the value of σ corresponds to a larger error. This implies that a lower standard deviation (σ) leads to reduced error, thereby creating more precise models.

3D Visualization (Extra):

- Figure and Subplot Setup:
    - Created a new 3D figure (fig) with a subplot (ax) using mpl_toolkits.mplot3d.
- Meshgrid Creation:
    - Utilized np.meshgrid to create a meshgrid for n and sigma values (n_mesh and sigma_mesh).
- Surface Plotting:
    - Plotted the 3D surface using ax.plot_surface.
    - Used sigma_mesh, n_mesh, and errors.T as x, y, and z values, respectively.
    - Applied 'viridis' colormap (cmap) and set transparency (alpha) for better visualization.
- Axis Labels and Title:
    - Added labels to the x-axis (Sigma), y-axis (n - Size of dataset), and z-axis (Error).
    - Set the title as '3D Visualization of Linear Regression Error for Different Values of n and Sigma'.
- Color Bar Inclusion:
    - Integrated a color bar using fig.colorbar to provide a reference for the color mapping.
- Plot Display:
    - Displayed the 3D visualization using plt.show().

```python
# 3D Visualization (extra)

from mpl_toolkits.mplot3d import Axes3D

# Create 3D Visualization
fig = plt.figure(figsize=(12, 8))
ax = fig.add_subplot(111, projection='3d')

# Create a meshgrid for n and sigma values
n_mesh, sigma_mesh = np.meshgrid(n_values, sigma_values)

# Plot the surface
surf = ax.plot_surface(sigma_mesh, n_mesh, errors.T, cmap='viridis', alpha=0.8)

# Add labels and title
ax.set_xlabel('Sigma')
ax.set_ylabel('n (Size of dataset)')
ax.set_zlabel('Error')
ax.set_title('3D Visualization of Linear Regression Error for Different Values of n and Sigma')

# Add a color bar
fig.colorbar(surf, ax=ax, shrink=0.5, aspect=10)

plt.show()
```