IDC410

REPORT

Implement Linear Regression

Assignment 1

Name: **Fayiz M**

Registration Number: **MS21078**

Aim;

- To investigate 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.
- To perform some visualization like heatmap to show the relation between n, σ and error of linear regression

Theory;

Linear regression analysis is utilized to forecast the value of one variable based on another variable's value. The variable being forecasted is referred to as the dependent variable, while the variable used for prediction is termed the independent variable. Linear regression models offer simplicity and provide a straightforward mathematical formula for generating predictions. This analytical tool finds applications across diverse domains in both business and academic research. (Source: IBM)

Gradient Descent stands out as one of the predominant optimization algorithms extensively employed in training machine learning models. Its primary objective is to minimize the disparity between actual and anticipated outcomes through iterative refinement. This iterative process is fundamental for training both traditional machine learning and deep learning models. Notably, Gradient Descent facilitates the localization of the function's local minimum, contributing significantly to the optimization process. (Source: https://www.javatpoint.com)

Methods and Code Snippets;

- Function to generate an m+1 dimensional data set, of size n, consisting of m continuous independent variables (X) and one dependent variable (Y)

```
import numpy as np

def generate_dataset(sigma, n, m):
    X = np.random.randn(n, m)
    true_beta = np.random.randn(m + 1, 1)
    X_with_intercept = np.hstack((np.ones((X.shape[0], 1)), X))
    Y = np.dot(X_with_intercept, true_beta) + np.random.normal(loc=0, scale=sigma, size=(n, 1))
    return X, Y, true_beta
generate_dataset(4, 4, 5)
```

- function that learns the parameters of a linear regression line

```
def linear_regression_gradient_descent(X, Y, kappa, tau, lamda):
    m = X.shape[1]
    if X.shape[1] == m:
        X = np.hstack((np.ones((X.shape[0], 1)), X))
    m = X.shape[1]
    beta = np.random.randn(m, 1)
    prev_cost = np.inf

    for i in range(kappa):
        Y_pred = np.dot(X, beta)
        error = Y_pred - Y
        gradient = np.dot(X.T, error)
        beta -= lamda * gradient
        cost = np.mean((error) ** 2)
        if abs(cost - prev_cost) < tau:
            break

        prev_cost = cost

    return beta, cost
```

- Example with a training set.

```
sigma = 0.4
n = 100
m = 1
X, Y, true_beta = generate_dataset(sigma, n, m)
kappa = 1000
tau = 1e-6
lamda = 0.0001


beta_gd, final_cost = linear_regression_gradient_descent(X, Y, kappa, tau,lamda)
import matplotlib.pyplot as plt

# Scatter plot
plt.scatter(X, Y, label='Data')
plt.plot(X, np.dot(np.hstack((np.ones((X.shape[0], 1)), X)), true_beta), color='green', label='True Line')
plt.plot(X, np.dot(np.hstack((np.ones((X.shape[0], 1)), X)), beta_gd), color='red', label='Gradient Descent')
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Comparison of Linear Regression Methods')
plt.legend()
plt.show()
```

- To investigate 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.
- Scatter plots with different value of n and σ

```
import numpy as np
import matplotlib.pyplot as plt

n_values = [50, 100, 200, 500]
sigma_values = [0.1, 0.5, 1, 2]
fig, axs = plt.subplots(len(n_values), len(sigma_values), figsize=(15, 15))


for i, n in enumerate(n_values):
    for j, sigma in enumerate(sigma_values):
        X, Y, true_beta = generate_dataset(sigma, n, m)
        beta_gd, final_cost = linear_regression_gradient_descent(X, Y, kappa,tau,lamda)

# Scatter plot
        axs[i, j].scatter(X, Y, label='Data', s=10)
        axs[i, j].plot(X, np.dot(np.hstack((np.ones((X.shape[0], 1)), X)), true_beta), color='green', 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)
        axs[i, j].set_title(f'n={n}, sigma={sigma}')

handles, labels = axs[0, 0].get_legend_handles_labels()
fig.legend(handles, labels, loc='upper center')

plt.tight_layout()
plt.show()
```

- Heatmap's code snippet

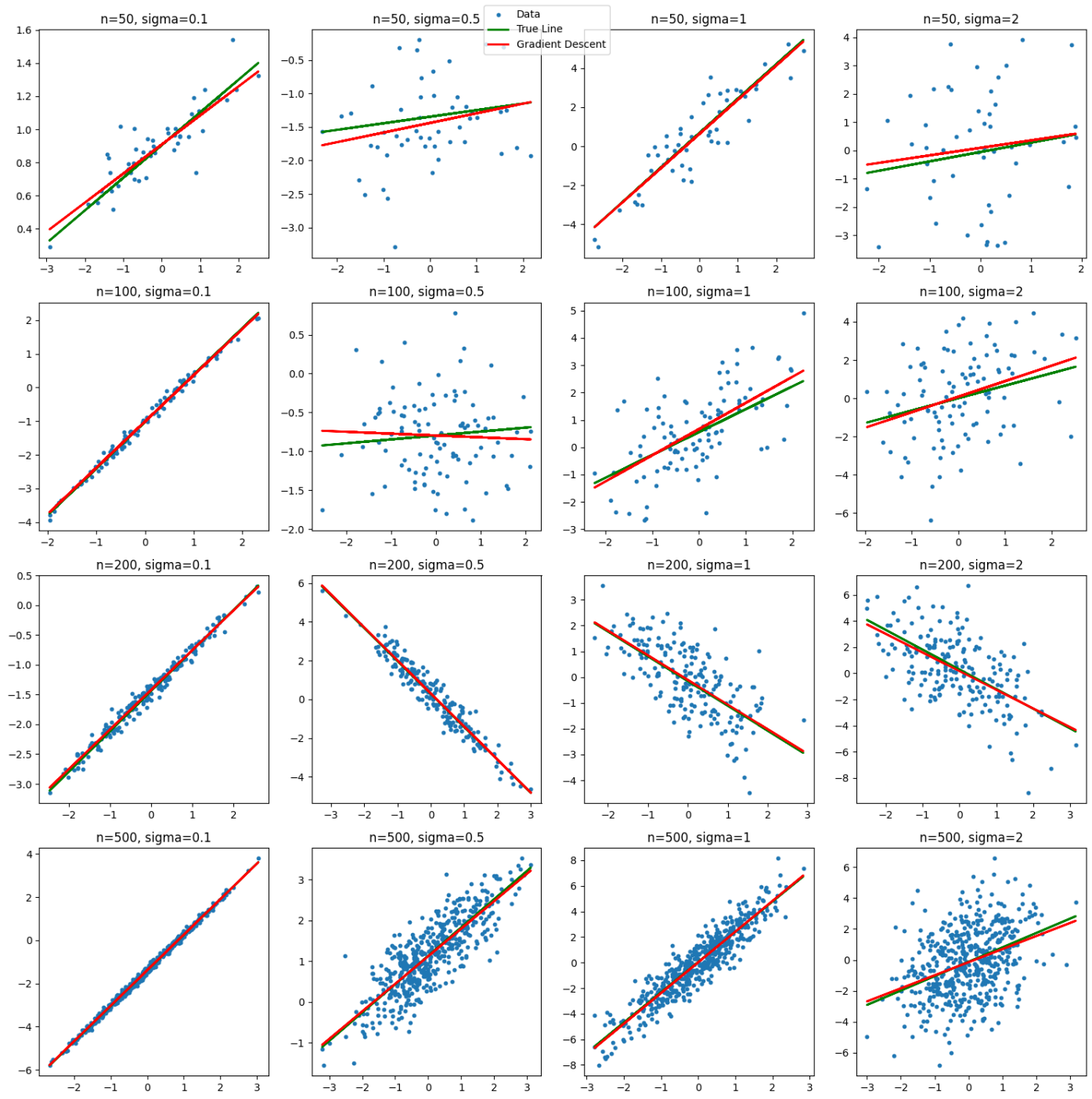
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


n_values = [50, 100, 200, 500]
sigma_values = [0.1, 0.5, 1, 2]
errors = np.zeros((len(n_values), len(sigma_values)))
for i, n in enumerate(n_values):
    for j, sigma in enumerate(sigma_values):
        X, Y, true_beta = generate_dataset(sigma, n, m)
        beta_gd, final_cost = linear_regression_gradient_descent(X, Y,kappa,tau,lamda)
        error = np.mean((true_beta - beta_gd)**2)
        errors[i, j] = error

# Creating a heatmap for better visualization
sns.heatmap(errors, annot=True, xticklabels=sigma_values, yticklabels=n_values, cmap='viridis')
plt.xlabel('Sigma')
plt.ylabel('n (Size of dataset)')
plt.title('Error of Linear Regression for Different Values of n and Sigma')
plt.show()
```

For further analysis of code, visit the below provided link for google Colab,

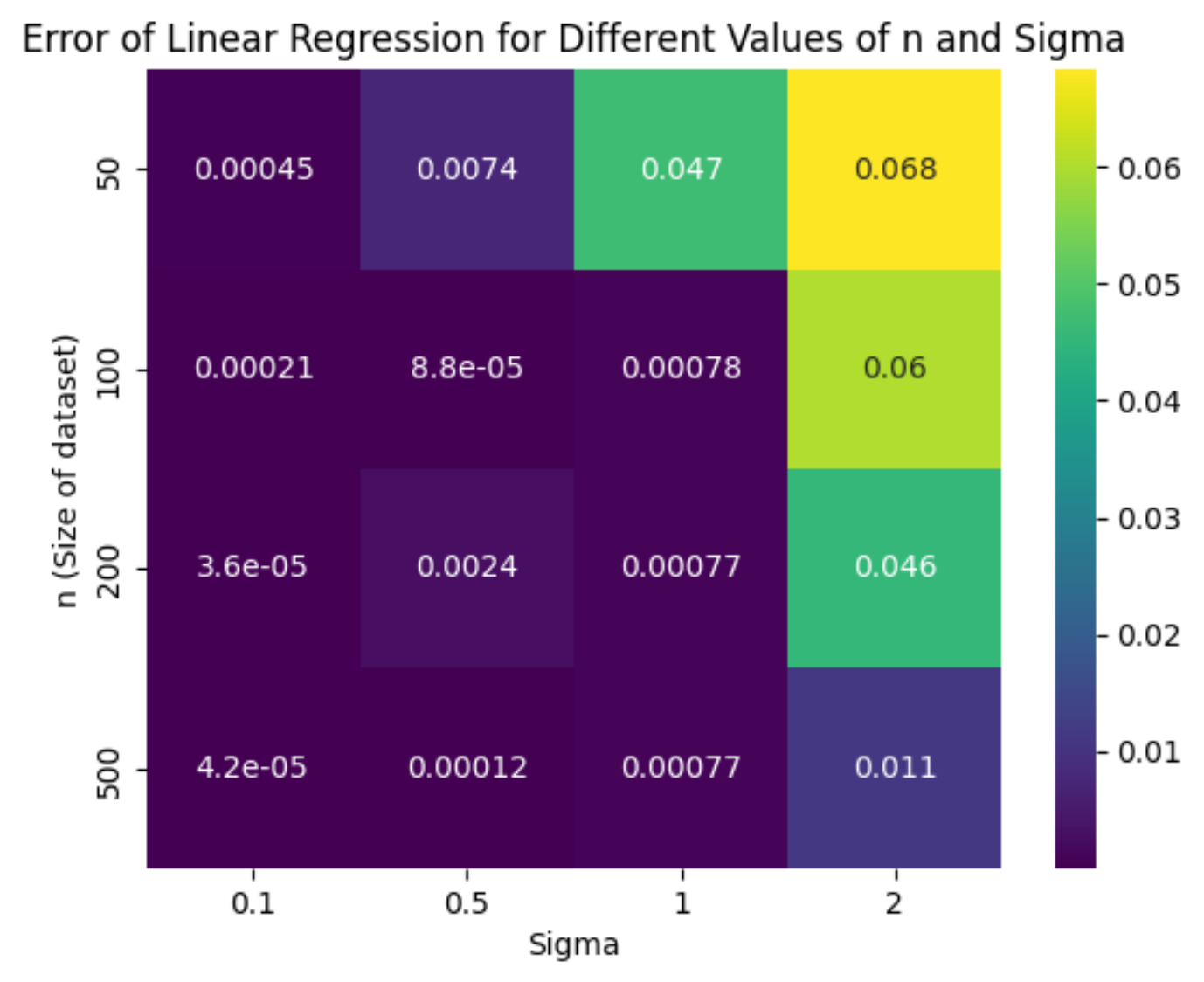
Google colab

[https://colab.research.google.com/drive/1fruPbN4PRFrMvP_8Bw9OJLiQPY10SDxf]

Visualization: Scatter plots;



Visualization: Heat Map;



Observation and Inferences;

- We can observe that as dataset size(n) increases the error of linear regression decreases
- The error decreases across all Sigma values. This suggests that larger sample sizes lead to more accurate linear regression models.
- The lowest error is observed at n=500 and Sigma=0.1. This combination of parameters might yield the most accurate linear regression model according to the given dataset.
- For a given n, the larger value of σ, larger the error is. This indicates that a lower standard deviation (σ) results in a lower error, hence more precise models.

- From the scatterplots for different value of n and sigma, we can infer that in most of the case the true line and gradient descent line appears to be similar and accurate. This suggests more accuracy of the linear regression using gradient descent algorithm.
- The heatmap showing n, sigma and error of linear regression uncoils that, as n value increases error decreases and lower the sigma lesser the error is.