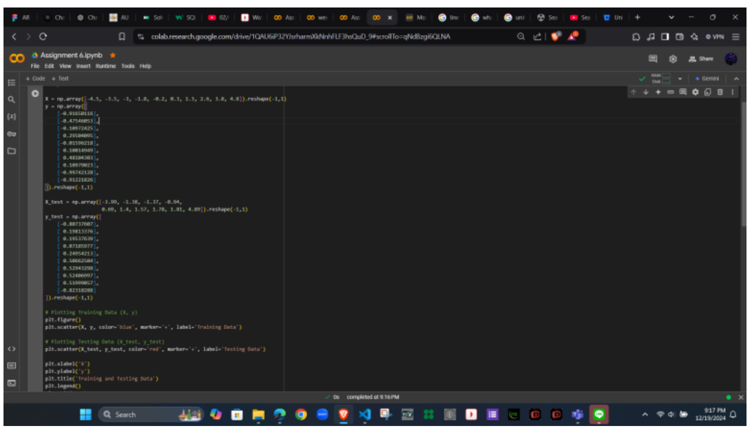
Yel Lin ID 6520242

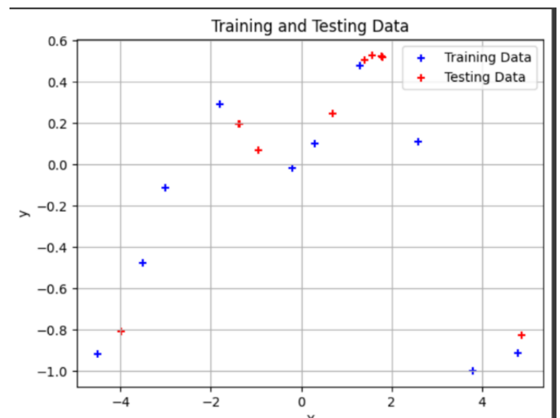
Assignment 6: Gradient Descent for Linear Regression

Question 1: Linear Regression Model ug Maximum Likelihood Estimation

Step 1.1 : Plotting Training and Testing Data

- Ploting the relationship between the training data and testing data on the same graph





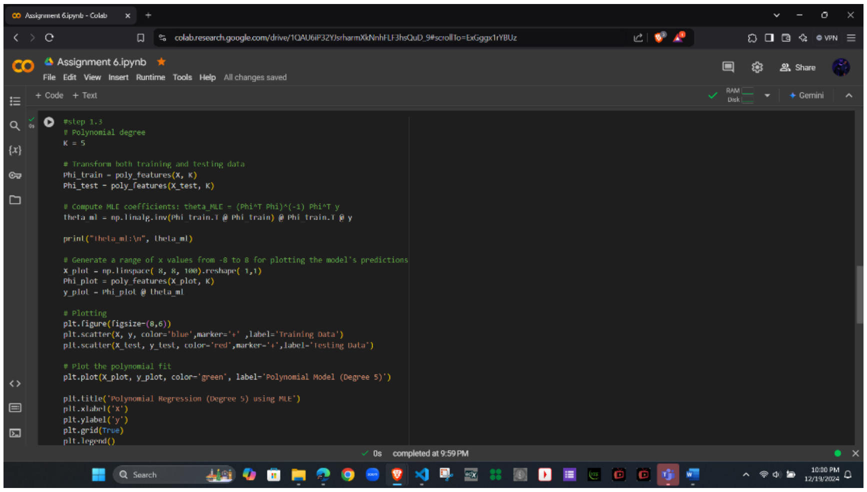
1.2 Polynomial Feature Transformation

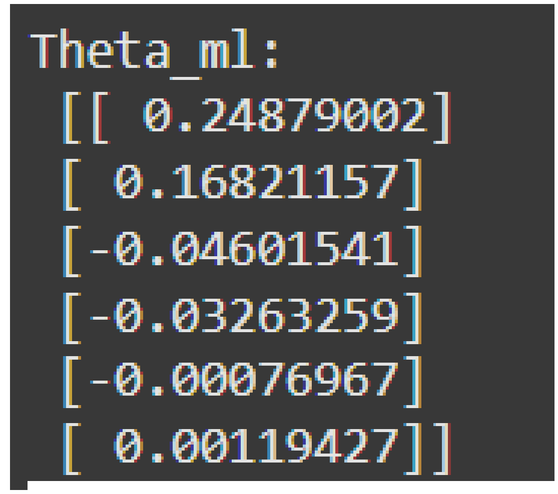
```
#step 1.2
def poly_features(X, K):
    # return np.hstack([X**i for i in range(K + 1)])
    X= X.flatten()
    N= X.shape[0]
    Phi= np.zeros((N,K+1))
    for k in range (K+1):
        Phi[:,k]= X**k
    return Phi
```

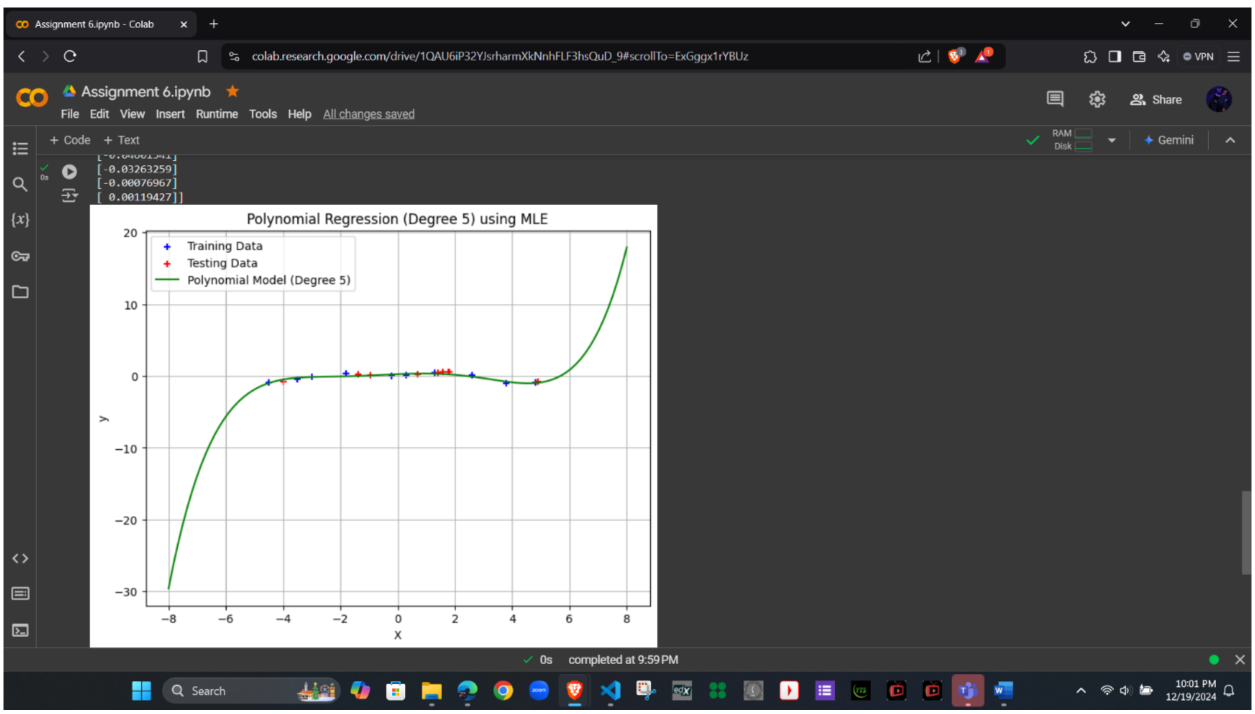
The purpose and the effect of this transformation:

Polynomial transformation enables linear models to fit nonlinear data by adding higher-degree features. It improves flexibility but risks underfitting with low degrees or overfitting with high degrees. The right degree balances accuracy and complexity for better generalization.

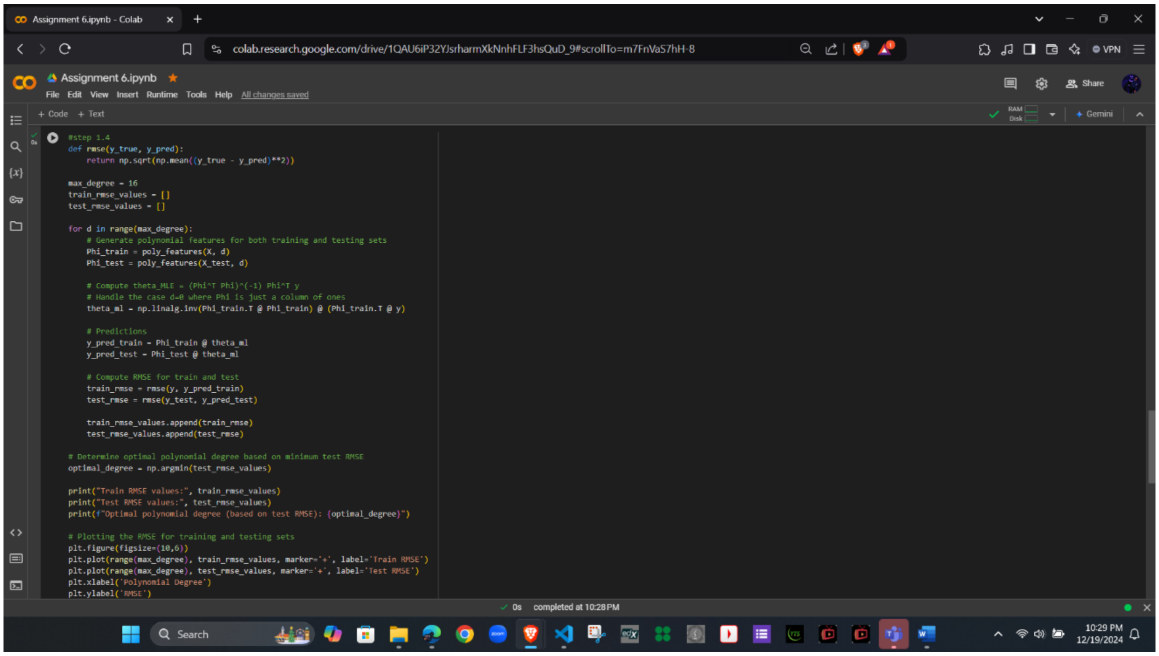
1.3 Fitting the Model Ug Maximum Likelihood

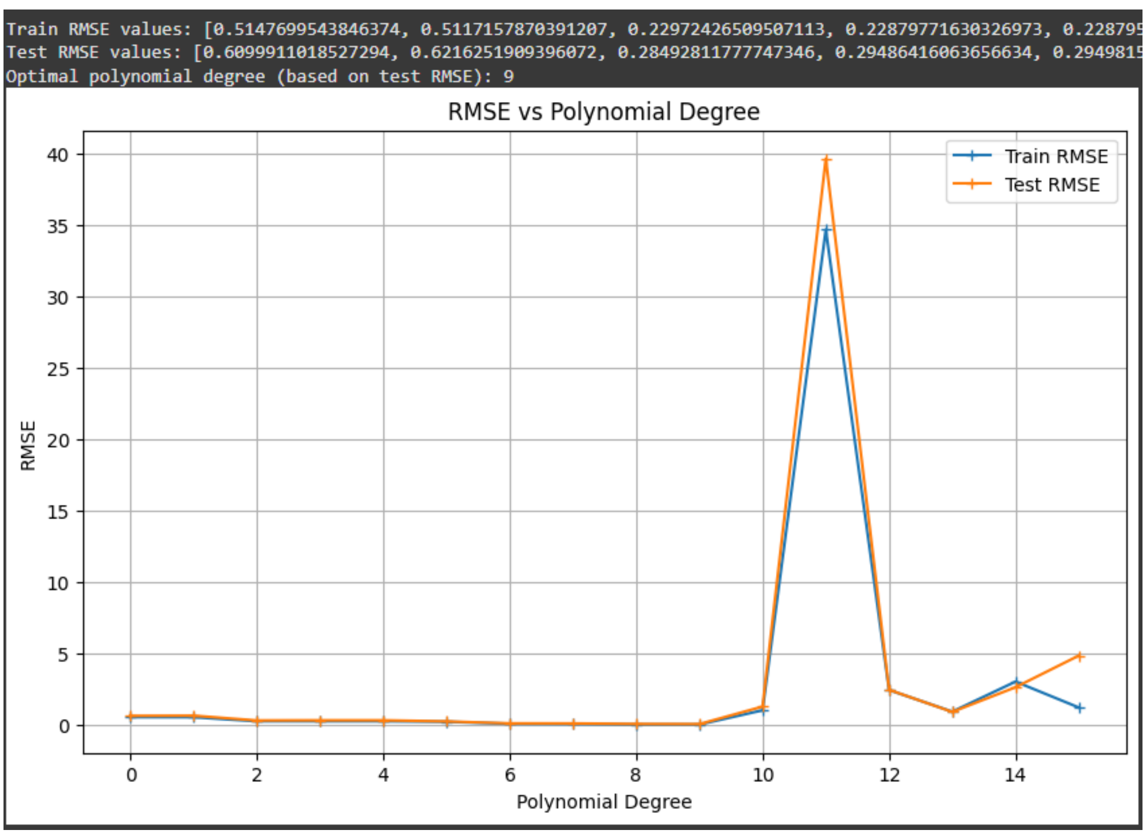






1.4 Model Evaluation





Train RMSE values: [0.5147699543846374, 0.5117157870391207, 0.22972426509507113, 0.22879771630326973, 0.22879573735413525, 0.19502327062655744, 0.05342435077718177, 0.03818517735253735, 0.003910516742283594, 5.538327228678245e-10, 0.999183995488885, 34.68901085509442, 2.4195160890609984, 0.9029673413927789, 3.0081770554024114, 1.1934142035010373]

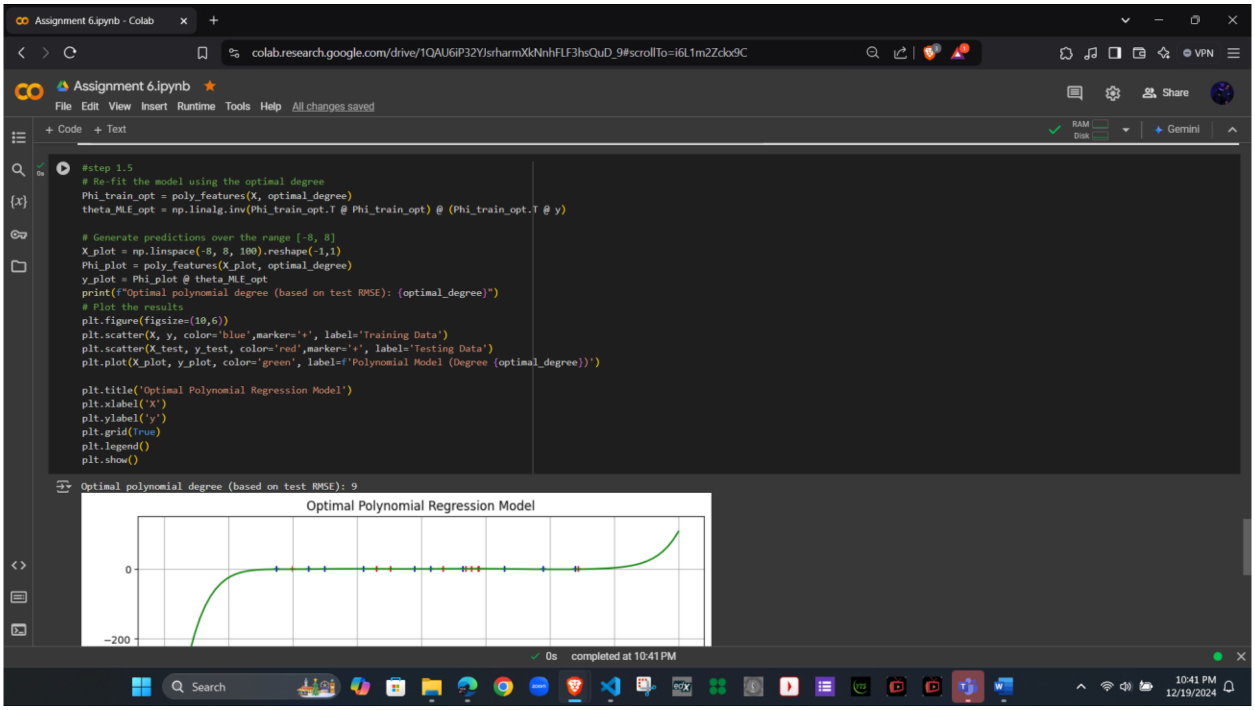
Final Train RMSE values: 1.1934142035010373

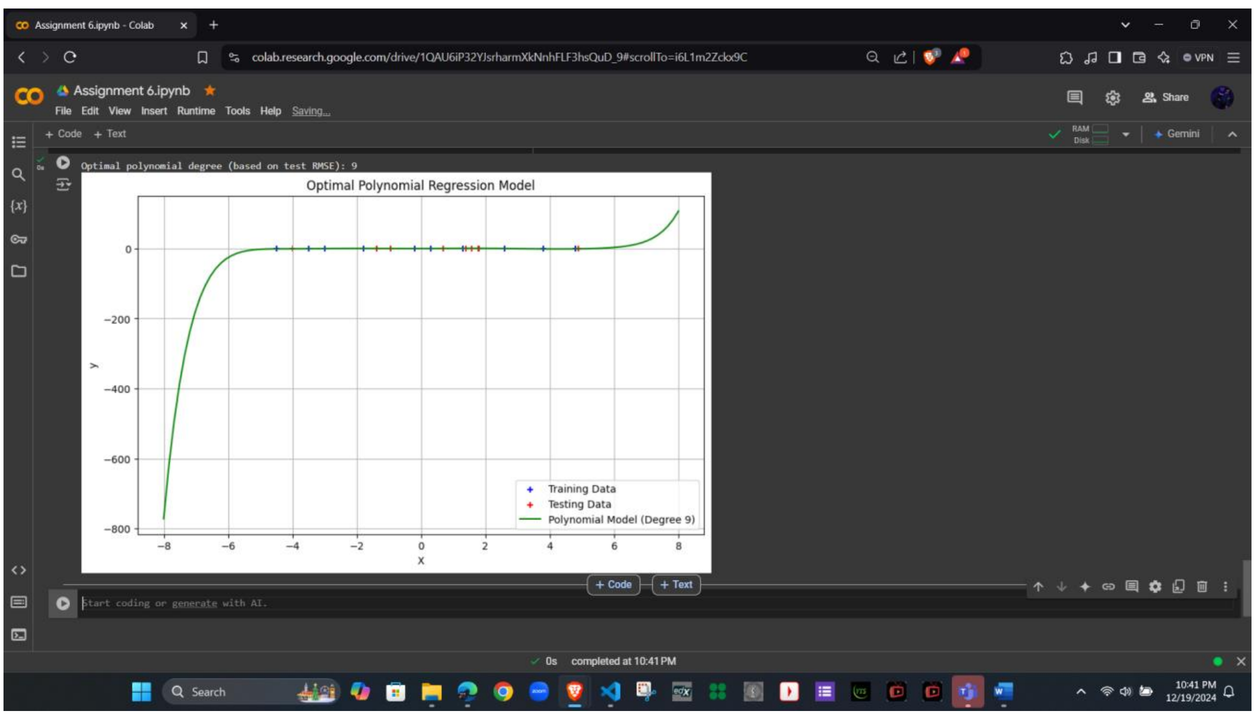
Test RMSE values: [0.6099911018527294, 0.6216251909396072, 0.28492811777747346, 0.29486416063656634, 0.29498152071402095, 0.21742862799598514, 0.08213402882291138, 0.07763539925156139, 0.03924484661505437, 0.030724661923518273, 1.2579162752970154, 39.66507331006011, 2.435318758318806, 0.8844692052280537, 2.613756810379 45, 4.830158928594374]

Final Test RMSE values: 4.830158928594374

Optimal polynomial degree (based on test RMSE): 9

1.5 Selecting the Best Model





Optimal polynomial degree (based on test RMSE): 9

Question 2: Gradient Descent for Polynomial Regression

2.1 Mathematical Derivation

$J(\theta)=(y-\Phi T\theta)2$

Step 1: Introduce a temporary variable

$h=\Phi T\theta$

$J(\theta)=(y-h)2$

Step 2: Differentiate with respect to h

$dJ/dh = 2(y-h)(-1)=-2(y-h)=2(h-y)$

Step 3: Differentiate h with respect to θ

$h=\Phi T\theta=i\sum\phi i\theta i$

$\partial h/\partial\theta = \Phi$

Step 4: Apply the chain rule

$\nabla\theta J(\theta)= dJ/dh\cdot\partial h/\partial\theta$

$dJ/dh=2(h-y)$

$\partial h/\partial\theta = \Phi$

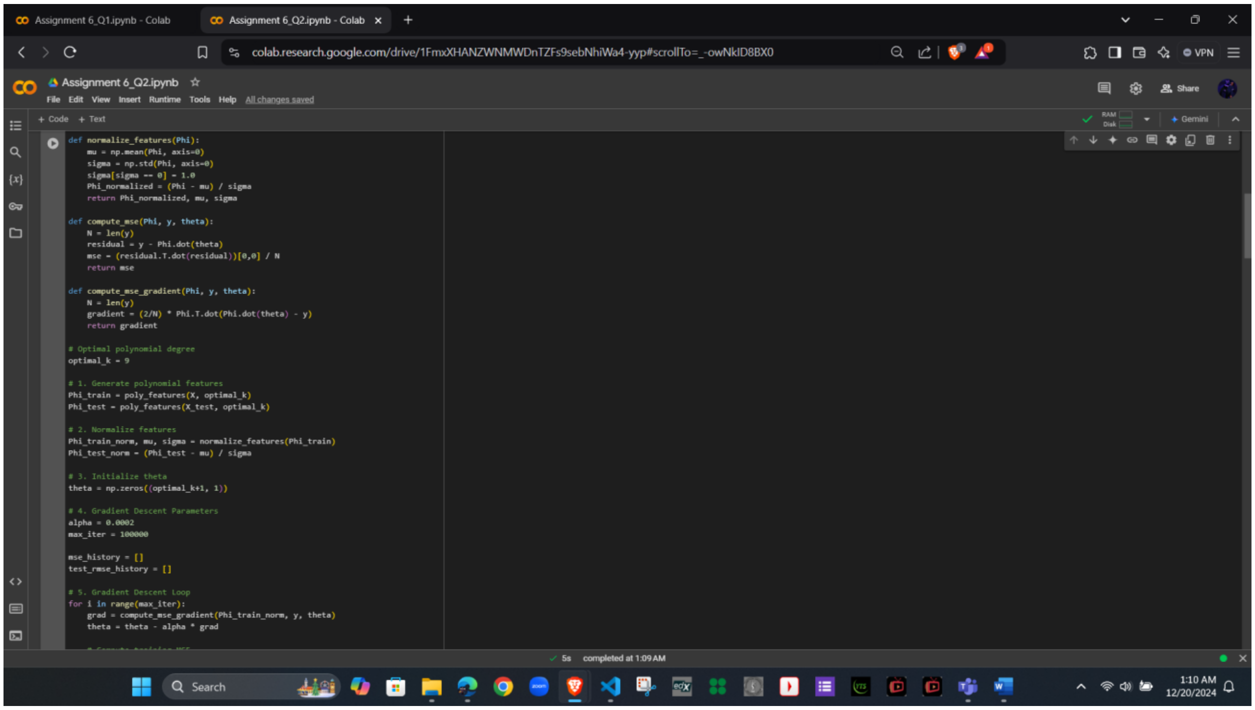
Thus , $\nabla\theta J(\theta)=2(h-y)\Phi$

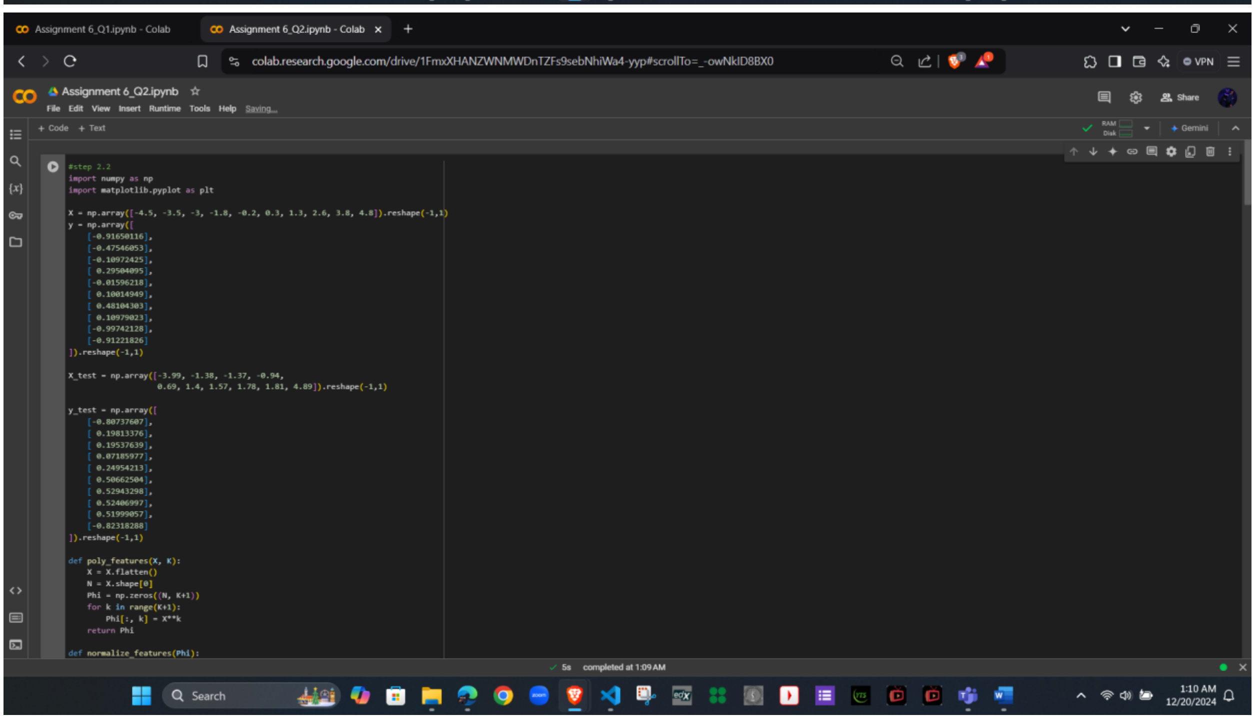
Step 5: **Substitute h=ΦTθ**

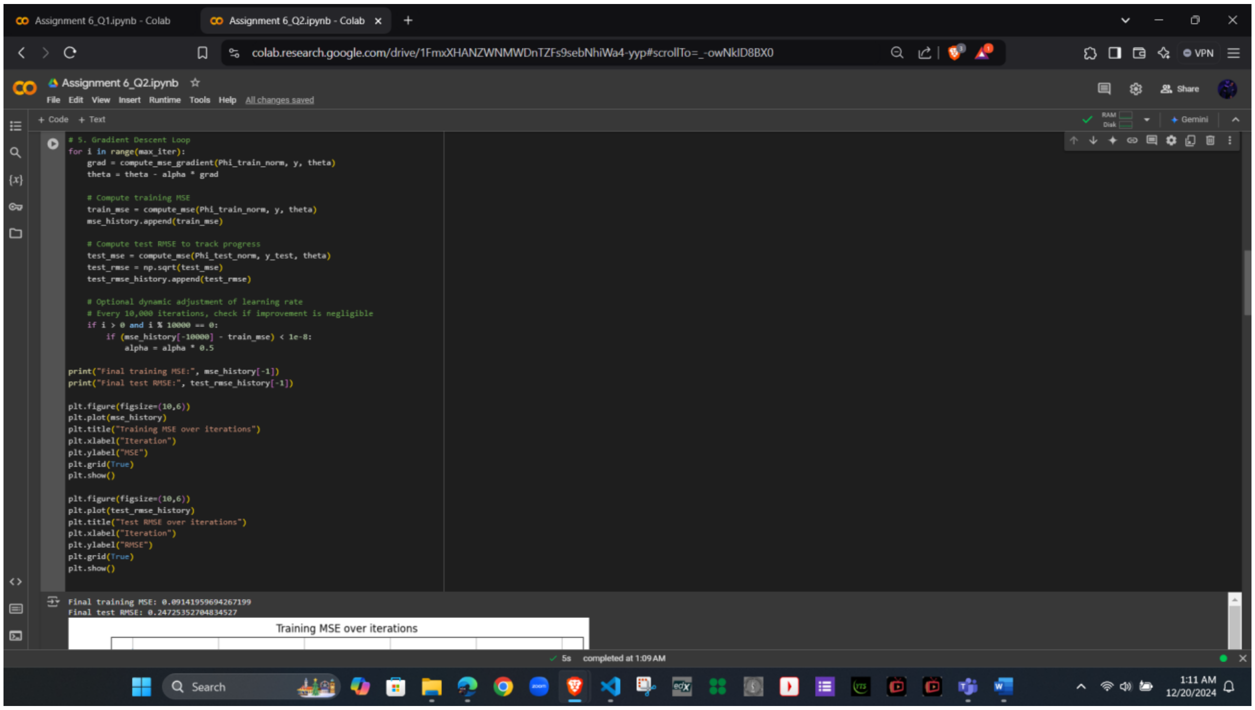
$\nabla\theta J(\theta)=2(\Phi T\theta-y)\Phi$

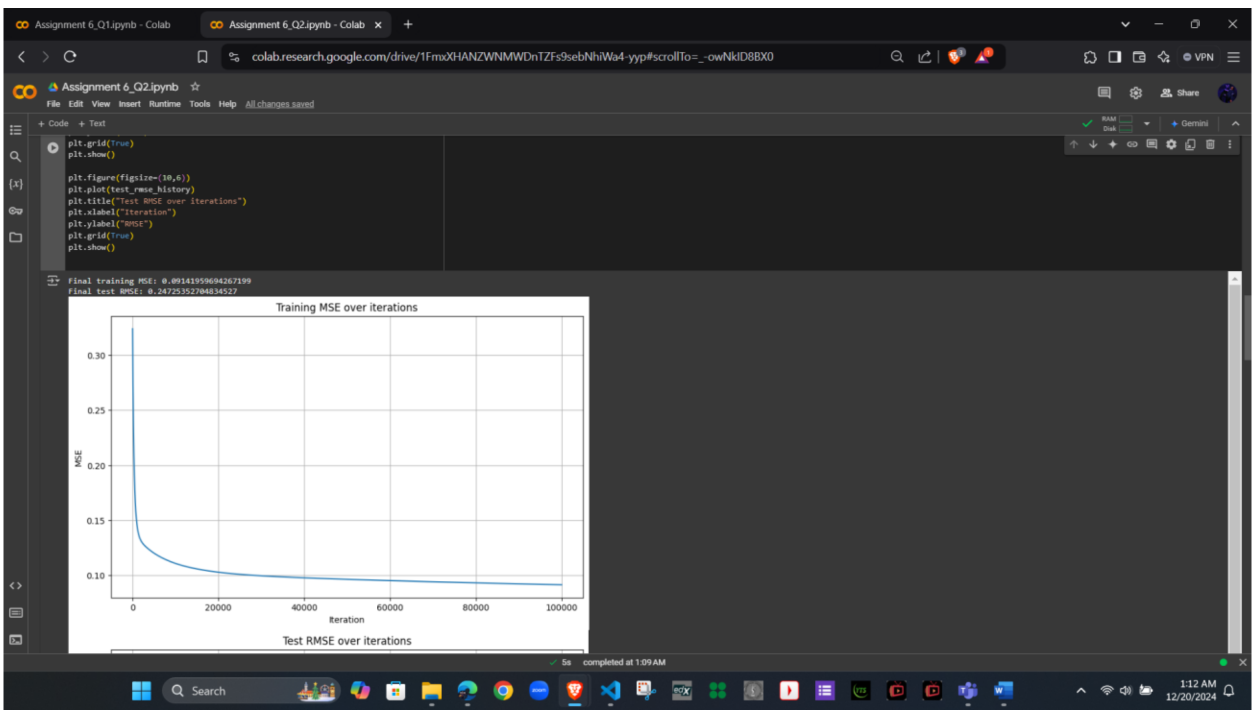
Final answer: $\nabla\theta J(\theta)=2(\Phi T\theta-y)\Phi$

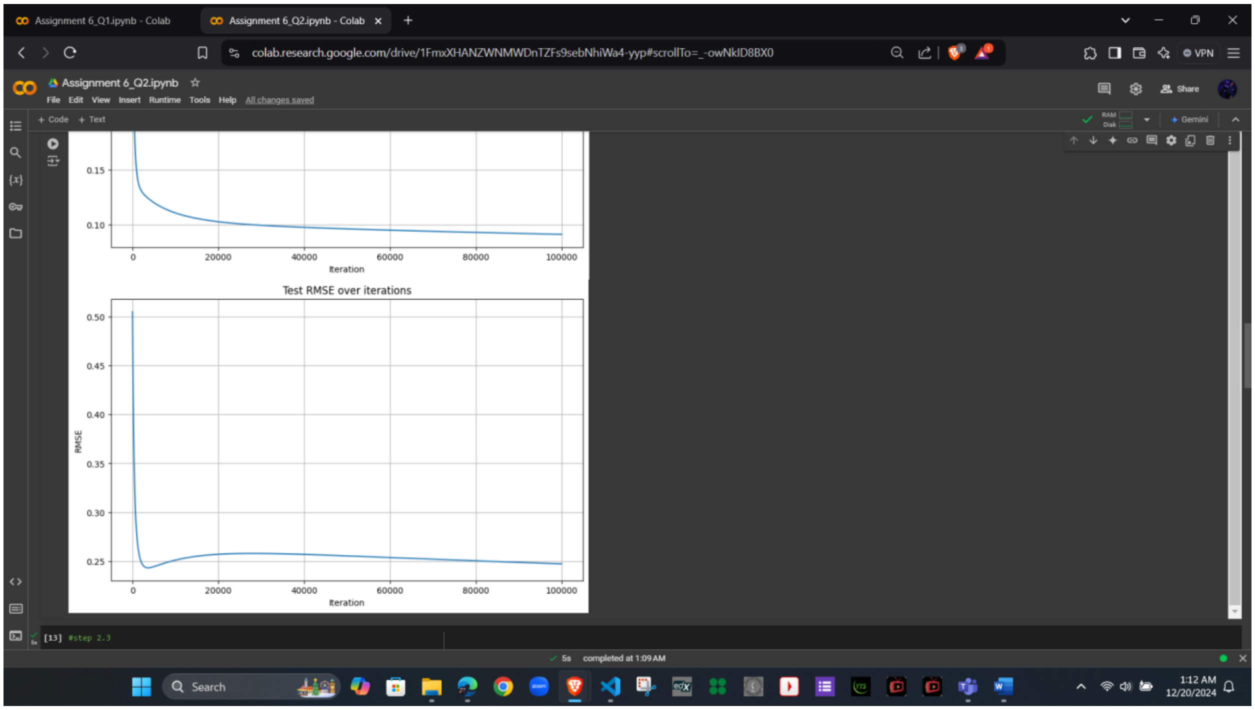
2.2 Feature Transformation and Gradient Descent Implementation





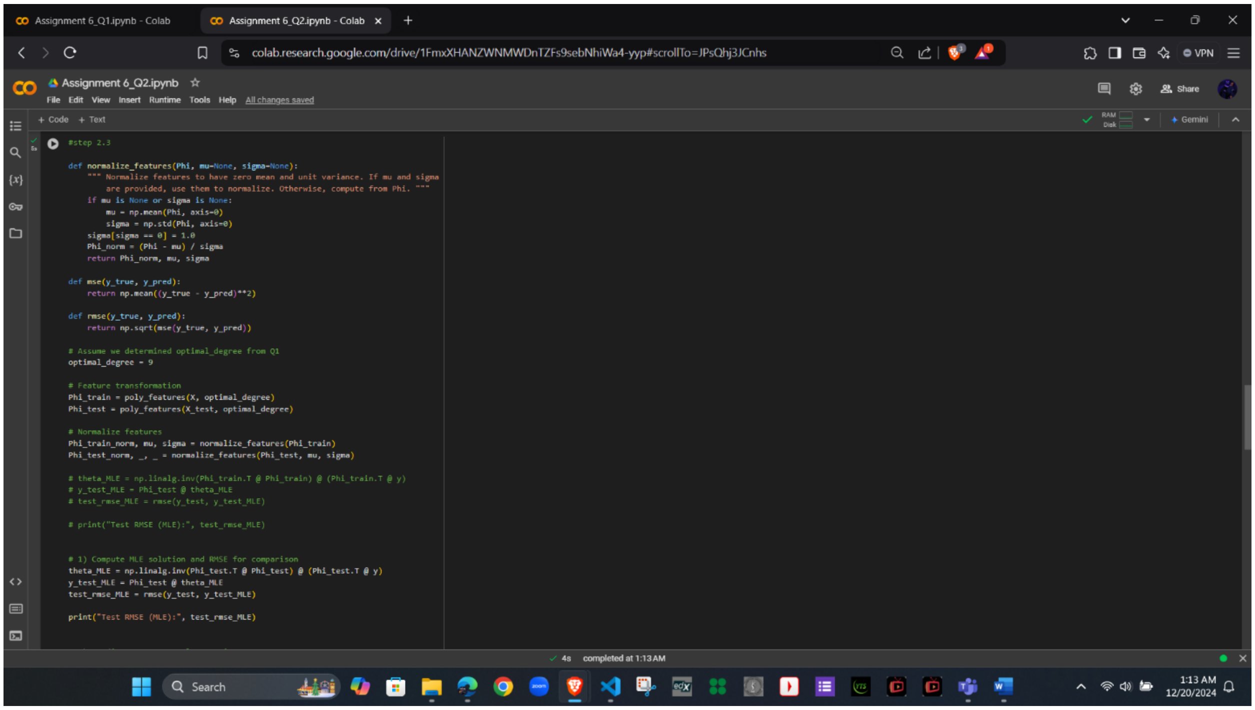

Assignment 6_Q2.ipynb
File Edit View Insert Runtime Tools Help All changes saved
# 5. Gradient Descent Loop
for i in range(max_iter):
    grad = compute_mse_gradient(Phi_train_norm, y, theta)
    theta = theta - alpha * grad
    # Compute training MSE
    train_mse = compute_mse(Phi_train_norm, y, theta)
    mse_history.append(train_mse)
    # Compute test RMSE to track progress
    test_mse = compute_mse(Phi_test_norm, y_test, theta)
    test_rmse = np.sqrt(test_mse)
    test_rmse_history.append(test_rmse)
    # Optional dynamic adjustment of learning rate
    # Every 10,000 iterations, check if improvement is negligible
    if i > 0 and i % 10000 == 0:
        if (mse_history[-10000] - train_mse) < 1e-8:
            alpha = alpha * 0.5
print("Final training MSE:", mse_history[-1])
print("Final test RMSE:", test_rmse_history[-1])
plt.figure(figsize=(10,6))
plt.plot(mse_history)
plt.title("Training MSE over iterations")
plt.xlabel("Iteration")
plt.ylabel("MSE")
plt.grid(True)
plt.show()
plt.figure(figsize=(10,6))
plt.plot(test_rmse_history)
plt.title("Test RMSE over iterations")
plt.xlabel("Iteration")
plt.ylabel("RMSE")
plt.grid(True)
plt.show()
Final training MSE: 0.09141959694267199
Final test RMSE: 0.24725352704834527
Training MSE over iterations
completed at 1:09 AM
1:11 AM
12/20/2024

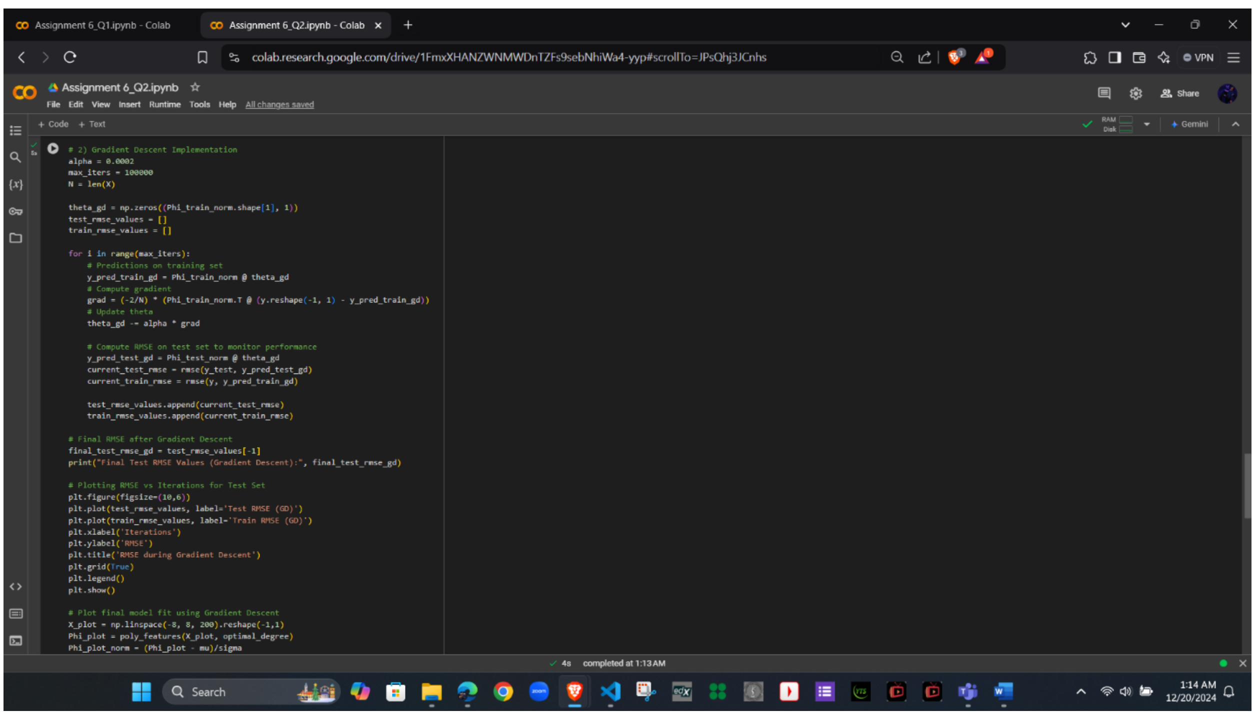

Assignment 6_Q2.ipynb
File Edit View Insert Runtime Tools Help All changes saved
plt.grid(True)
plt.show()
plt.figure(figsize=(10,6))
plt.plot(test_rmse_history)
plt.title("Test RMSE over iterations")
plt.xlabel("Iteration")
plt.ylabel("RMSE")
plt.grid(True)
plt.show()
Final training MSE: 0.09141959694267199
Final test RMSE: 0.24725352704834527
Training MSE over iterations
MSE
0.30
0.25
0.20
0.15
0.10
0
20000
40000
60000
80000
100000
Iteration
Test RMSE over iterations
completed at 1:09 AM
1:12 AM
12/20/2024

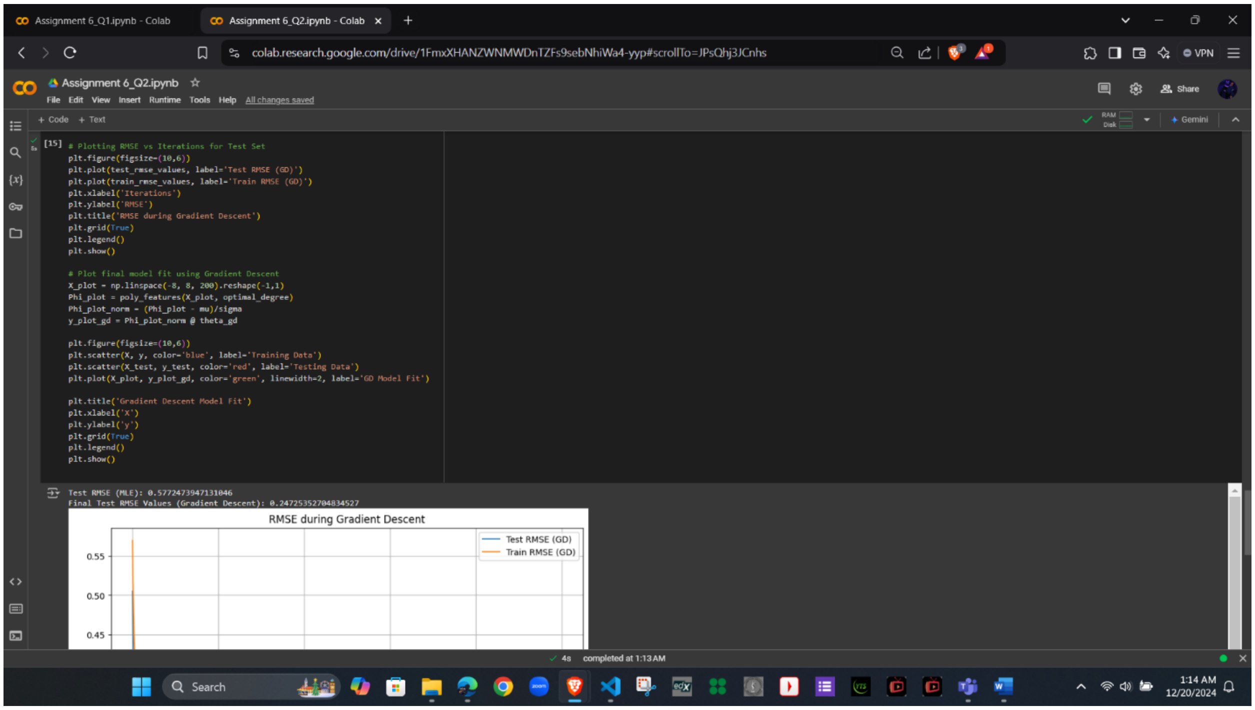


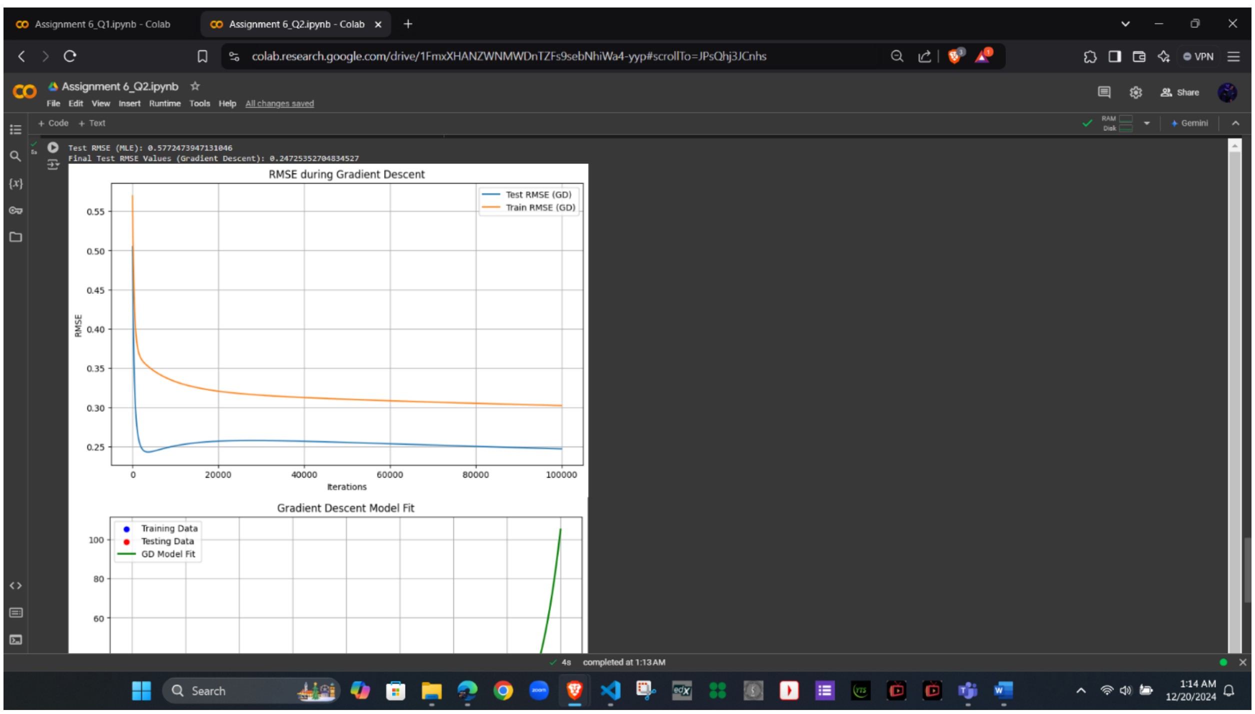
Assignment 6_Q1.ipynb - Colab
Assignment 6_Q2.ipynb - Colab
colab.research.google.com/drive/1FmxXHANZWNMWDnTZFs9sebNhiWa4-yyp#scrollTo=_-owNkID8BX0
Assignment 6_Q2.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Iteration
Test RMSE over iterations
RMSE
Iteration
[13] #step 2.3
5s completed at 1:09 AM
Search
1:12 AM
12/20/2024

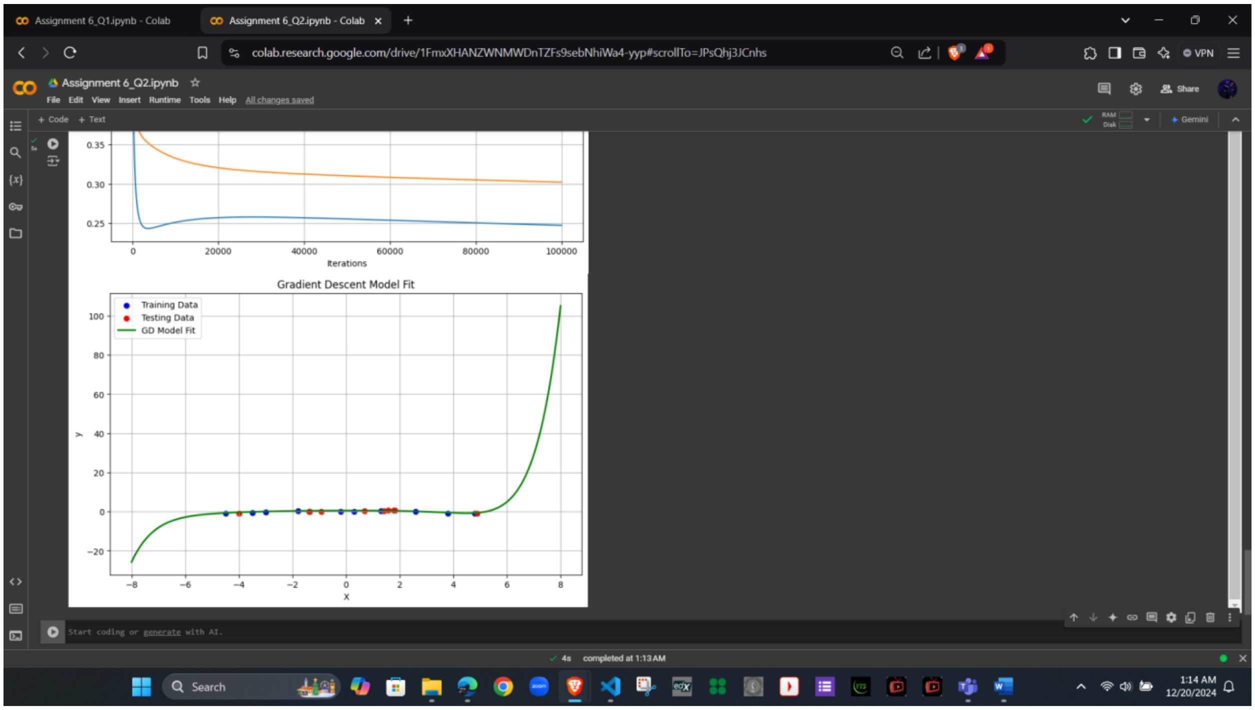
2.3 Model Evaluation and Comparison






Assignment 6_Q2.ipynb
# Plotting RMSE vs Iterations for Test Set
plt.figure(figsize=(10,6))
plt.plot(test_rmse_values, label='Test RMSE (GD)')
plt.plot(train_rmse_values, label='Train RMSE (GD)')
plt.xlabel('Iterations')
plt.ylabel('RMSE')
plt.title('RMSE during Gradient Descent')
plt.grid(True)
plt.legend()
plt.show()
# Plot final model fit using Gradient Descent
X_plot = np.linspace(-8, 8, 200).reshape(-1,1)
Phi_plot = poly_features(X_plot, optimal_degree)
Phi_plot_norm = (Phi_plot - mu)/sigma
y_plot_gd = Phi_plot_norm @ theta_gd
plt.figure(figsize=(10,6))
plt.scatter(X, y, color='blue', label='Training Data')
plt.scatter(X_test, y_test, color='red', label='Testing Data')
plt.plot(X_plot, y_plot_gd, color='green', linewidth=2, label='GD Model Fit')
plt.title('Gradient Descent Model Fit')
plt.xlabel('X')
plt.ylabel('y')
plt.grid(True)
plt.legend()
plt.show()
Test RMSE (MLE): 0.5772473947131046
Final Test RMSE Values (Gradient Descent): 0.24725352704834527
RMSE during Gradient Descent
Test RMSE (GD)
Train RMSE (GD)


Assignment 6_Q2.ipynb
Test RMSE (MLE): 0.5772473947131046
Final Test RMSE Values (Gradient Descent): 0.24725352704834527
RMSE during Gradient Descent
Test RMSE (GD)
Train RMSE (GD)
RMSE
Iterations
Gradient Descent Model Fit
Training Data
Testing Data
GD Model Fit



Test RMSE (MLE): 0.5772473947131046

Final Test RMSE Values (Gradient Descent): 0.24725352704834527

Discussion:

- The training MSE curve should decrease over iterations, indicating successful convergence.
- The final GD solution should produce a test RMSE similar to the MLE solution.
- Gradient Descent convergence speed and final accuracy depend on the learning rate and normalization.
- Normalization and a sufficiently small learning rate (0.0002) help ensure smooth convergence.