(1) You are given a data-set in the file FMLA1Q1Data train.csv with 10000 points in (R 2 , R) (Each row corresponds to a datapoint where the first 2 components are features and the last component is the associated y value).

Q1. Write a piece of code to obtain the least squares solution wML to the regression problem using the analytical solution.
Analytical Solution:
The closed-form solution for the least squares problem is given by:
w_ML = (X^T X)^(-1) X^T y
This equation is derived from minimizing the mean squared error (MSE) by setting the gradient of the loss function to zero.

Q2. Code the gradient descent algorithm with suitable step size to solve the least squares algorithms and plot // wt − wML // 2 as a function of t. What do you observe?

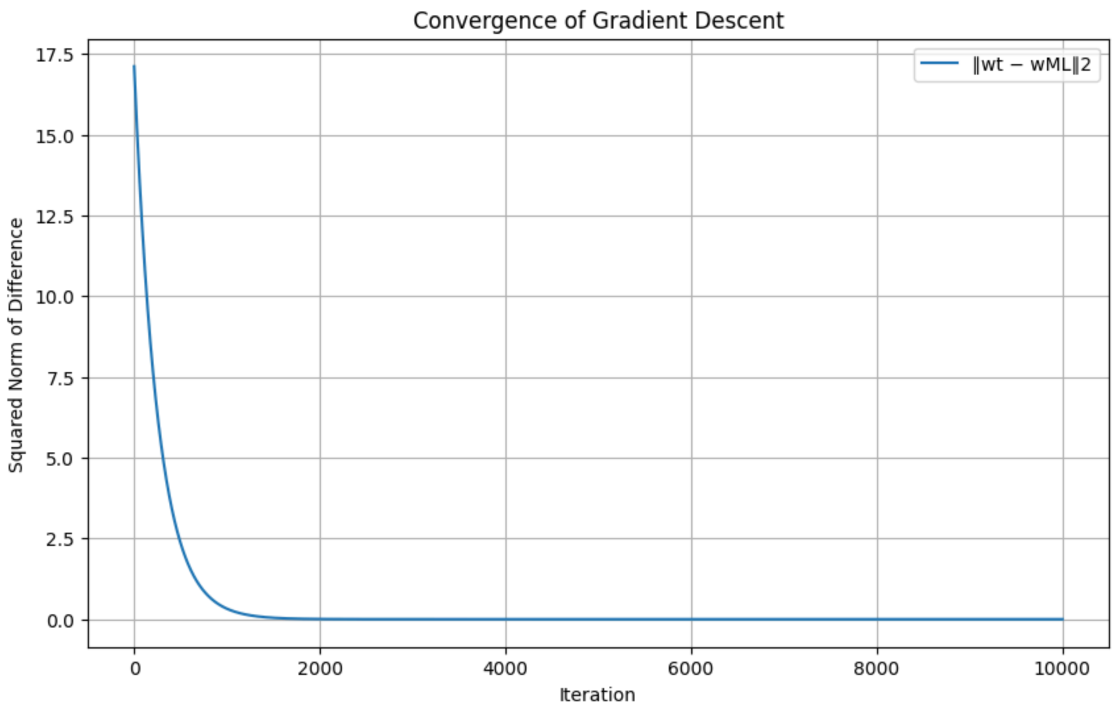
Step to follow to perform the following taks.
1. Compute the analytical solution w_ML for comparison.
2. Implement the gradient descent algorithm to iteratively update weights w_t.
3. Track the norm |w_t - w_ML|^2 for each iteration.
4. Plot the norm(Y-axis) over iterations(X-axis) and analyze the convergence behavior.

Gradient Descent Algorithm for Least Squares
The objective is to minimize the mean squared error (MSE):

L(w) = (1/2m)sum(i=1 to m) (y^i - X^i w)^2
The gradient of the MSE with respect to w is:
L(w) = (1/m)(X^T (X w - y))

Closed-form Solution w_{ML: The exact solution for least squares is computed using the formula w_ML = (X^T X)^(-1) X^T y

Gradient Descent:
The weights are initialized to zero. For each iteration, the gradient of the loss function is computed and used to update the weights. After each weight update, |w_t - w_ML|^2 is computed to track how far the current weights w_t are from the closed-form solution w_ML.
Plot: The norm |w_t - w_ML|^2 (Y-axis) is plotted as a function of iterations(X-axis) to visualize how the gradient descent converges.



Observation:

Convergence Behavior:

- As the number of iterations increases,|w_t - w_ML|^2 decreases, it means the gradient descent solution w_t gets closer to the analytical solution w_ML.
- Initially, the norm decreases rapidly, showing fast convergence in the early stages. Over time, the updates become smaller as the solution approaches the optimal point.

Learning Rate Impact:

- If the learning rate is too small, convergence will be slow, and it will take many iterations for w_t to approach w_ML.
- If the learning rate is too large, the algorithm might overshoot and oscillate around w_ML, leading to unstable or divergent behavior.

Final Weight Difference:

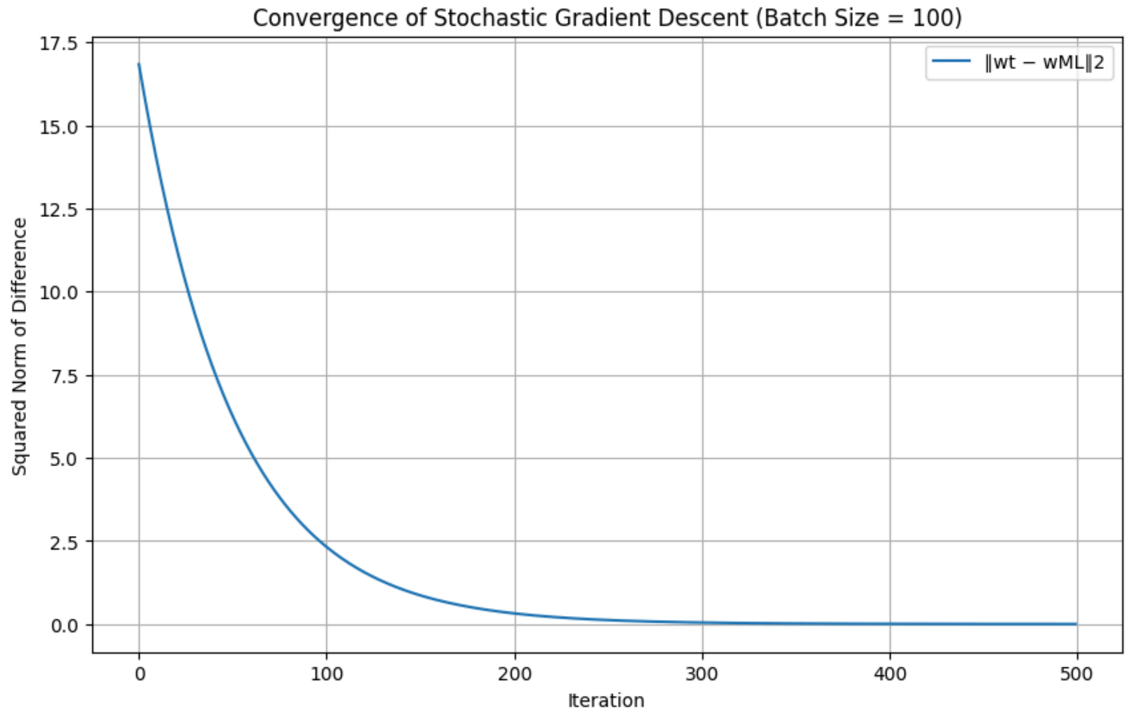
- After completing the gradient descent process, the norm ||wt-w_ML||^2 should be close to zero, indicating that the iterative solution is near the optimal solution.

Q3. Code the stochastic gradient descent algorithm using batch size of 100 and plot // wt − wML // 2 as a function of t. What are your observations

Learning Rate: I have used an learning rate for smoother convergence.
Iterations: Tune the number of iterations for faster and stable convergence with a batch size of 100 and I have taken 500 number of iterations .

Steps to follow to perform the following.

1. Calculate w_ML: The analytical least squares solution is computed using the closed-form equation: w_ML= (X^T X)^(-1) X^T y
2. Stochastic Gradient Descent (SGD):
   The dataset is shuffled for each iteration.A batch size of 100 is used(as given in the question), and for each batch, the gradient is computed and weights are updated with the help of previous weights.After every full pass through the dataset (say,one epoch), compute ||w_t - w_ML||^2, the Euclidean norm, which tells about how far the current weights w_t are from the analytical solution w_ML.
3. Convergence plot:
   Convergence plot is plotted against the norm ||w_t - w_ML||^2 on the Y-axis vs the interactions on the X-Axis.
4. Observations:
   Stochastic Gradient Descent with batch size 100 converges smoothly towards the analytical solution w_ML as the number of iterations.
5. Weight Difference:
   The final printed weight difference |w_t - w_ML|^2 should approach zero as Stochastic Gradient Descent converges, indicating that the iterative solution has reached close to the closed-form solution w_ML.



Q4. Code the gradient descent algorithm for ridge regression. Cross-validate for various choices of λ and plot the error in the validation set as a function of λ. For the best λ chosen, obtain wR. Compare the test error (for the test data in the file FMLA1Q1Data test.csv) of wR with wML. Which is better and why?

In this task, we have to implement a ridge regression model using gradient descent. We have to perform cross-validation to determine the best regularization parameter lambda for some values of lambda from 0.0001 to 100.

Ridge Regression Overview:
Ridge Regression: Ridge regression is a linear model where we add a regularization term to penalize large coefficients.
The cost function for ridge regression is:
J(w) = sum from i=1 to m{(1/2m)(yi - Xiw)^2} + lambda*w^2.

w represents the model parameters (weights).
lambda is the regularization parameter that controls the strength of the penalty applied to large weights. A larger lambda leads to more regularization.

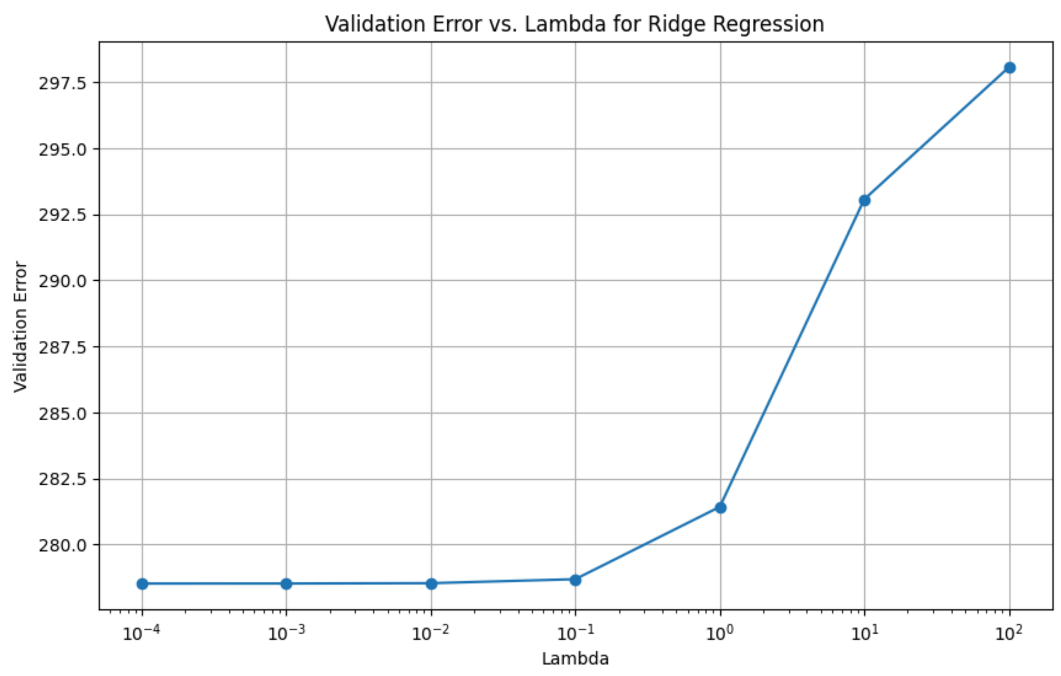
Gradient Descent for Ridge Regression:
To solve the ridge regression problem, we use the gradient descent algorithm. The update rule for the weights is:
w = w - alpha((1/m) X^T (Xw - y) + 2 lambda w), where
alpha is the learning rate , **m** is the number of samples in the dataset, **lambda** is the regularization parameter.

Cross-Validation and Hyperparameter Tuning:
Cross-Validation: I have used 5-fold cross-validation to select the best lambda from a range of values between 0.0001 to 100. In cross-validation, the training data is split into 5 subsets (folds). Each fold is used once as validation data while the remaining folds are used for training.

Error Metric: The Mean Squared Error (MSE) is used to evaluate the performance of the model.

Ordinary Least Squares (OLS) for Comparison with ridge regression:
We have computed the OLS solution w_ML using the given equation:
w_ML = (X^T X)^(-1) X^T y
OLS provides a baseline model to compare against ridge regression.

Results
Best lambda: After cross-validation, the best lambda(lambda=0.0001) was selected based on the minimum validation error.



Test Error Comparison:

Test error for ridge regression is a little bit higher than the ordinary least square.
Test error for ridge regression is 143.78
Test error for ordinary least squares is 142.76
Based on the test error, I have found that OLS is performing better as compared to Ridge regression.

Regularization Effect: Ridge regression adds a regularization term to prevent overfitting by penalizing large weights. The best lambda found during cross-validation balances the trade-off between bias and variance.

Q5. Assume that you would like to perform kernel regression on this dataset. Which Kernel would you choose and why? Code the Kernel regression algorithm and predict for the test data. Argue why/why not the kernel you have chosen is a better kernel than the standard least squares regression.

Kernel Choice:
Chosen Kernel: Radial Basis Function (RBF) Kernel.
K=exp(-||X1-X2||^2 /2sigma^2)
Justification for RBF:
Non-linearity: RBF kernels are more powerful for capturing non-linear relationships, as they can model complex patterns by focusing on local variations.
Smoothness: The RBF kernel ensures smooth predictions, which is often important in regression tasks where the goal is to predict continuous values.

Flexibility: The kernel’s flexibility is controlled by the `gamma` parameter, which allows the model to adjust how much influence nearby points have on predictions.

Kernel Regression Algorithm:

Kernel Ridge Regression (RBF): The Radial Bias Kernel algorithm calculates the kernel matrix using the RBF kernel,it regularizes the solution using ridge regression, and solves for the dual coefficients.

Parameters Used:

gamma = 0.5 , I have used gamma=0.5 because best gamma=1/(number of features)

Comparison with Ordinary Least Squares (OLS):

OLS (wML): For comparison, the ordinary least squares (OLS) method was applied using the normal equation for solving linear regression.

Evaluation Metrics:

Metric Used: Mean Squared Error (MSE) is used for both kernel regression and OLS.

Results:

Test Error for Kernel Ridge Regression (RBF) is much less than the Test Error for Ordinary least Square.

Observations:

Kernel Ridge Regression with RBF performed better when the underlying relationship between features and target variable is non-linear(As our data is non linear). The RBF kernel can model more complex interactions, which ordinary least squares might miss.