Question 2 (Coding)
2(a) How would you preprocess the data? Explain your reasoning for using this pre-processing.

To preprocess the data for training and evaluation, the following steps were implemented:

1. **Dropping Non-Numeric or Irrelevant Columns**
   Columns containing date or time in their names were removed from the dataset. These columns often represent temporal information in formats that are not inherently numeric, which may not contribute meaningfully to predictive modeling in their raw form. Dropping them ensures the model focuses on relevant features while avoiding unnecessary noise or complications during training.
2. **Handling Missing Values**
   Rows with missing values were dropped. Missing values can negatively impact the model's training process by introducing bias or reducing the accuracy of predictions. Removing such rows ensures a clean and consistent dataset for training, validation, and testing.
3. **Separating Features and Target**
   The target variable (Appliances) was separated from the feature variables in all datasets (training, validation, and testing). This separation is critical for supervised learning tasks, as the target variable is used to compute the loss and guide model optimization.
4. **Scaling Features**
   The features were scaled using StandardScaler, which normalizes each feature to have a mean of zero and a standard deviation of one. This step is essential for algorithms like Ridge and Lasso regression, which are sensitive to the scale of input features. Scaling ensures that all features contribute equally to the optimization process and helps improve convergence during training.

---

**Justification for Preprocessing Steps:**

- **Relevance and Noise Reduction:** Dropping irrelevant columns ensures the model focuses only on variables that are likely to contribute meaningful information.
- **Data Consistency:** Handling missing values guarantees that the model does not encounter inconsistencies during training and evaluation.
- **Feature-Target Separation:** This is a standard practice in machine learning to properly define the predictive relationship.

- **Normalization:** Scaling features prevents any one variable from disproportionately influencing the model, especially in regression-based tasks.

These preprocessing steps collectively ensure that the data is clean, consistent, and ready for effective model training and evaluation.

2(c) Apply the standard linear regression. Your must return following metrics and the associated values are the numeric values.

Standard Linear Regression Metrics Table

| | Metric | Value |
|---|---|---|
| 1 | Train RMSE | 98.23103848878637 |
| 2 | Train $R^2$ | 0.18675401399493108 |
| 3 | Validation RMSE | 97.53986578772172 |
| 4 | Validation $R^2$ | 0.0026790084850937257 |
| 5 | Test RMSE | 100.03023430387991 |
| 6 | Test $R^2$ | -0.21168451232273933 |

2 (f) Report (using a table) the RMSE and R 2 for training, validation, and test for all the different λ values you tried. What would be the optimal parameter you would select based on the validation data performance?

Ridge Regression Metrics Table

| | Alpha | Train RMSE | Train $R^2$ | Validation RMSE | Validation $R^2$ | Test RMSE | Test $R^2$ |
|---|---|---|---|---|---|---|---|
| 1 | 0.001 | 98.2310384892513 | 0.18675401398723268 | 97.53993670205654 | 0.0026775583215 15061 | 100.03039801358983 | -0.2116884784172 65 |
| 2 | 0.01 | 98.23103853520469 | 0.18675401322634 5 | 97.54057486319942 | 0.0026645081897 89472 | 100.03187131883303 | -0.2117241715699 5582 |
| 3 | 0.1 | 98.23104305576229 | 0.18675393837575 87 | 97.54694924126379 | 0.0025341501060 512917 | 100.04659488028 08 | -0.2120809020409 8973 |
| 4 | 1.0 | 98.23142955118874 | 0.18674753884070405 | 97.60954538295918 | 0.0012535862888 750637 | 100.19175366963609 | -0.2156006987179 424 |
| 5 | 10.0 | 98.24434040599783 | 0.18653374831548 43 | 98.03467419193045 | -0.007465243709 017688 | 101.1850541871464 | -0.2398230946511 10063 |

Optimal Ridge alpha based on validation RMSE: **0.001**

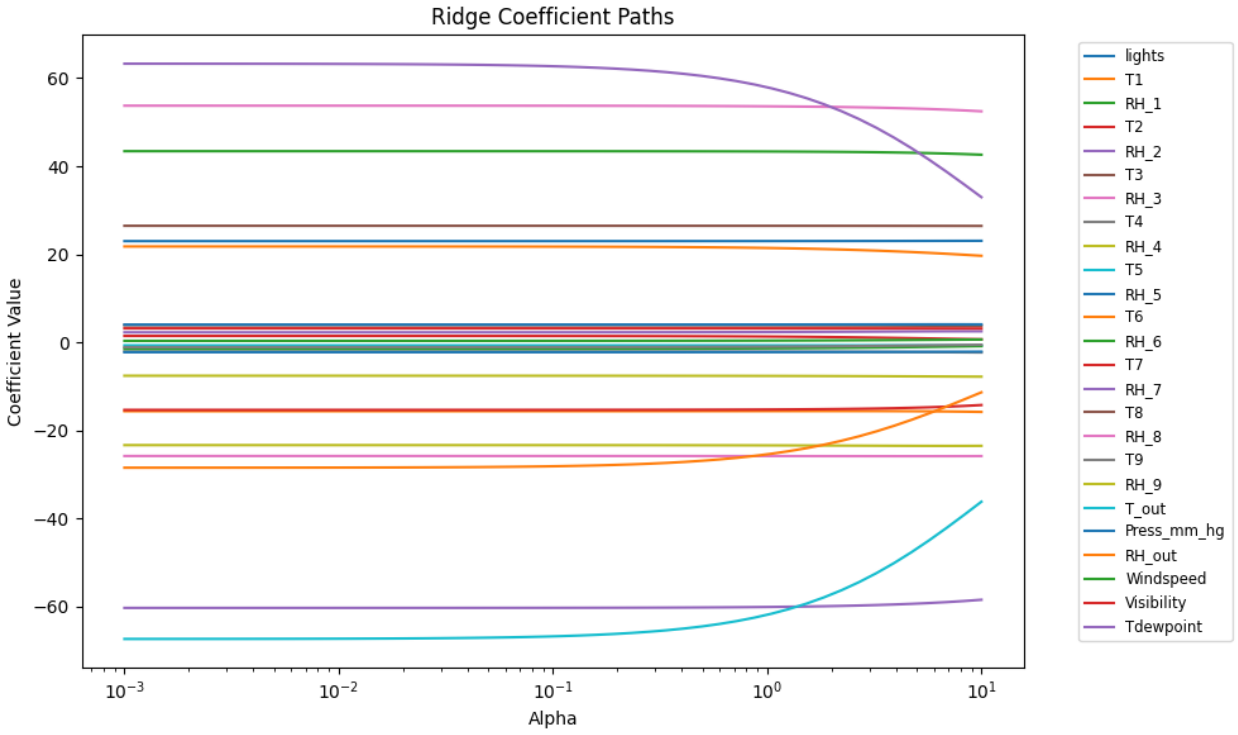
**Lasso Regression Metrics Table**

| | Alpha | Train RMSE | Train $R^2$ | Validation RMSE | Validation $R^2$ | Test RMSE | Test $R^2$ |
|---|---|---|---|---|---|---|---|
| 1 | 0.001 | 98.23105379930793 | 0.18675376048603287 | 97.55156053691982 | 0.002439842314897467 | 100.05640914229632 | -0.2123187164920579 |
| 2 | 0.01 | 98.23256902935141 | 0.18672867138047888 | 97.67596462510818 | -0.00010608707251891225 | 100.33933967409682 | -0.21918458220153347 |
| 3 | 0.1 | 98.28591031917716 | 0.18584520227805845 | 98.508612433841 7 | -0.017229757430997764 | 101.992596259909 | -0.25969173171725 |
| 4 | 1.0 | 98.84194814819675 | 0.17660722689707897 | 96.53103923609712 | 0.023202327125585676 | 96.22388050960129 | -0.12122486114921216 |
| 5 | 10.0 | 104.84016605524218 | 0.07363986381810783 | 96.24229697924464 | 0.029037153843525 12 | 89.4742148432645 | 0.030555889133546454 |

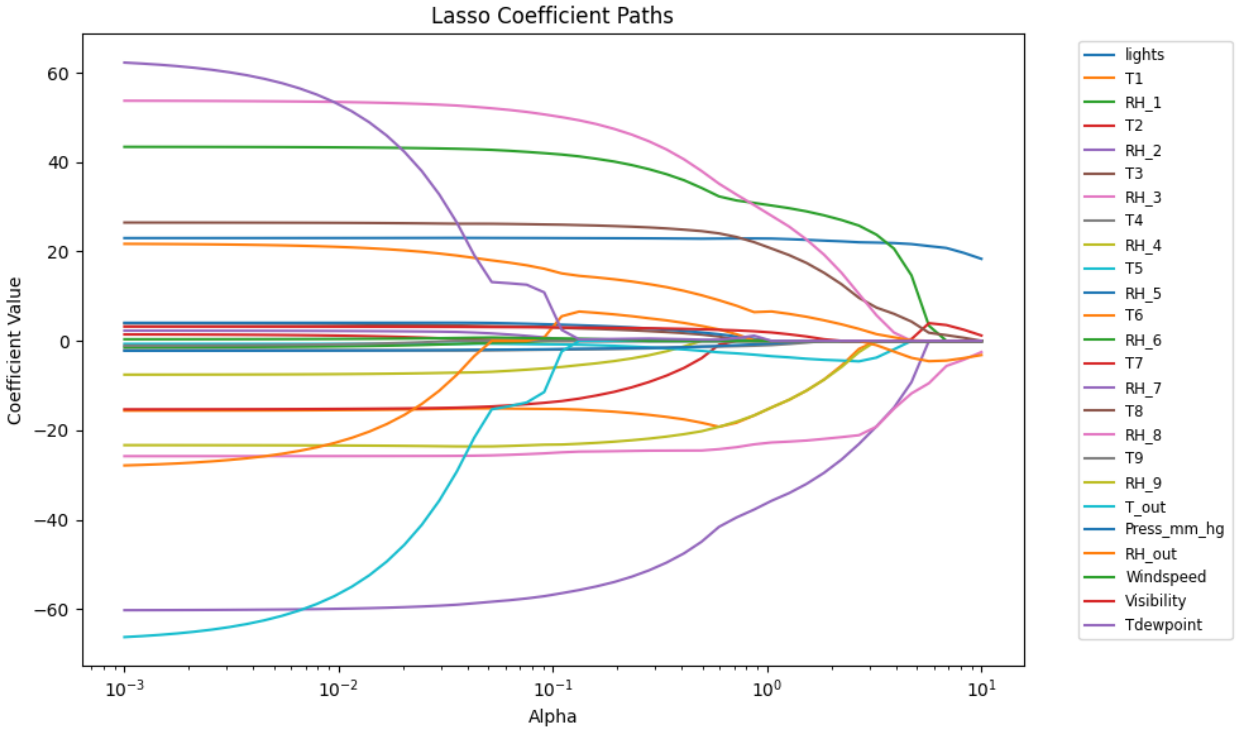
Optimal Lasso alpha based on validation RMSE: 10.0

2(g) Generate the coefficient path plots (regularization value vs. coefficient value) for both ridge and lasso. Make sure that your plots encompass all the expected behavior (coefficients should shrink towards 0).

Plotting Ridge Coefficient Paths:



Plotting Lasso Coefficient Paths:




2 (h) What are 3 observations you can draw from looking at the coefficient path plots, and the metrics?

Observations:

1. For Ridge regression, coefficients shrink smoothly as alpha increases, but rarely become exactly zero.
2. For Lasso regression, many coefficients are driven exactly to zero for moderate to high alpha values, effectively performing feature selection.
3. The optimal regularization parameter shows a balance between underfitting and overfitting. Here, we selected the alpha value with the lowest validation RMSE as optimal for each model.

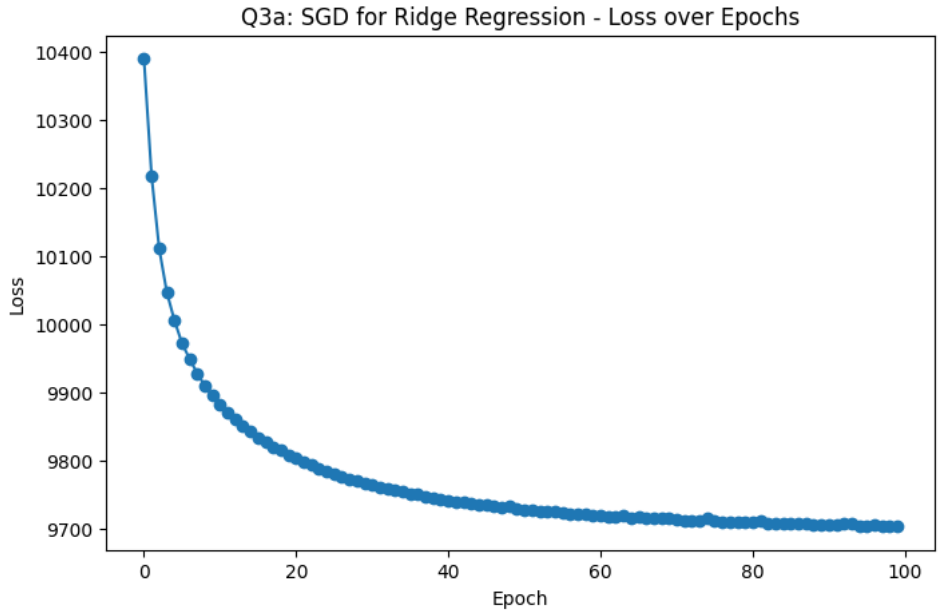
3(a) Your function should return a dictionary (paired value) where the key denotes the epoch number and the value of the loss associated with that epoch.

Training Ridge Regression with SGD

**Epoch And Loss Table**

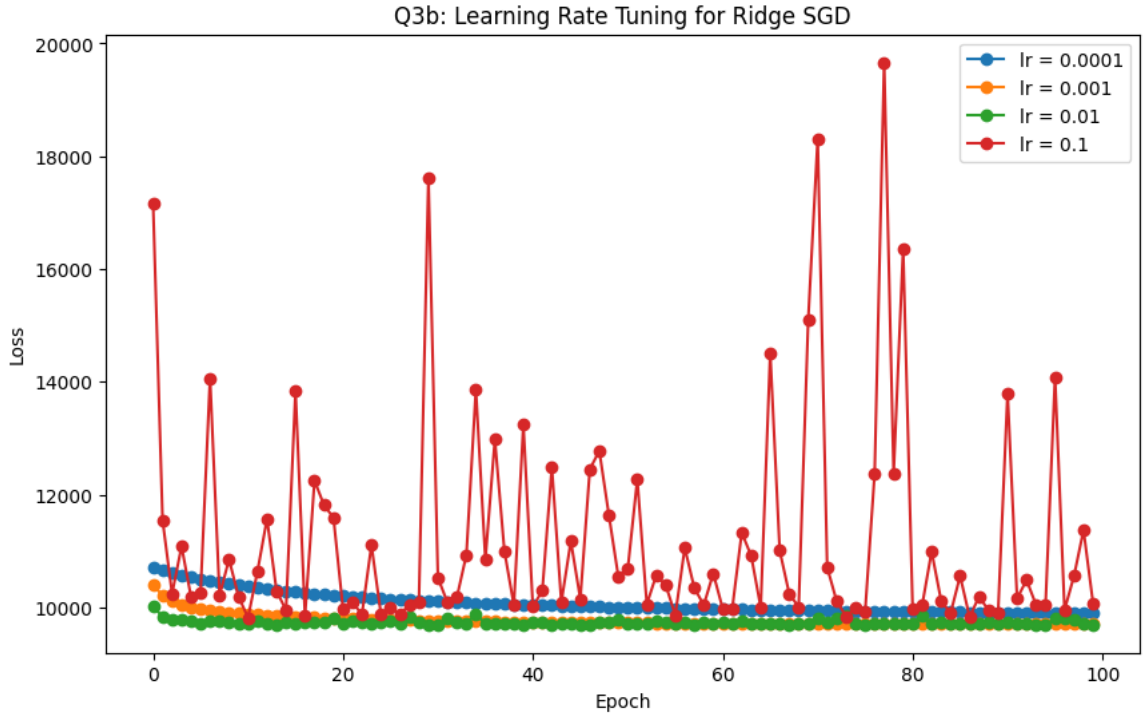
| | Epochs 0-19 | Epochs 20-39 | Epochs 40-59 | Epochs 60-79 | Epochs 80-99 |
|---|---|---|---|---|---|
| 1 | Epoch 0: Loss = 10390.297318 | Epoch 20: Loss = 9802.436043 | Epoch 40: Loss = 9740.496927 | Epoch 60: Loss = 9718.03998 | Epoch 80: Loss = 9708.10849 |
| 2 | Epoch 1: Loss = 10217.931808 | Epoch 21: Loss = 9797.848455 | Epoch 41: Loss = 9738.897588 | Epoch 61: Loss = 9717.34557 | Epoch 81: Loss = 9709.989157 |
| 3 | Epoch 2: Loss = 10111.190504 | Epoch 22: Loss = 9794.046016 | Epoch 42: Loss = 9738.889945 | Epoch 62: Loss = 9716.380519 | Epoch 82: Loss = 9707.474109 |
| 4 | Epoch 3: Loss = 10047.088184 | Epoch 23: Loss = 9787.906817 | Epoch 43: Loss = 9735.634274 | Epoch 63: Loss = 9717.889696 | Epoch 83: Loss = 9707.048309 |
| 5 | Epoch 4: Loss = 10005.490923 | Epoch 24: Loss = 9784.104924 | Epoch 44: Loss = 9734.546138 | Epoch 64: Loss = 9715.482879 | Epoch 84: Loss = 9707.594857 |
| 6 | Epoch 5: Loss = 9971.888771 | Epoch 25: Loss = 9779.586177 | Epoch 45: Loss = 9733.738301 | Epoch 65: Loss = 9716.715968 | Epoch 85: Loss = 9706.555311 |
| 7 | Epoch 6: Loss = 9948.745677 | Epoch 26: Loss = 9776.619717 | Epoch 46: Loss = 9731.610591 | Epoch 66: Loss = 9715.568223 | Epoch 86: Loss = 9706.221082 |
| 8 | Epoch 7: Loss = 9927.784754 | Epoch 27: Loss = 9772.443094 | Epoch 47: Loss = 9731.366568 | Epoch 67: Loss = 9713.901922 | Epoch 87: Loss = 9706.593537 |
| 9 | Epoch 8: Loss = 9909.444897 | Epoch 28: Loss = 9769.897083 | Epoch 48: Loss = 9731.628807 | Epoch 68: Loss = 9714.609583 | Epoch 88: Loss = 9705.461615 |
| 10 | Epoch 9: Loss = 9895.157143 | Epoch 29: Loss = 9766.049398 | Epoch 49: Loss = 9728.961283 | Epoch 69: Loss = 9714.628088 | Epoch 89: Loss = 9705.394584 |
| 11 | Epoch 10: Loss = 9881.248693 | Epoch 30: Loss = 9764.027997 | Epoch 50: Loss = 9727.466405 | Epoch 70: Loss = 9712.343529 | Epoch 90: Loss = 9705.435255 |
| 12 | Epoch 11: Loss = 9870.037684 | Epoch 31: Loss = 9760.204947 | Epoch 51: Loss = 9726.150465 | Epoch 71: Loss = 9711.462181 | Epoch 91: Loss = 9704.806167 |
| 13 | Epoch 12: Loss = 9859.380092 | Epoch 32: Loss = 9758.044277 | Epoch 52: Loss = 9725.111576 | Epoch 72: Loss = 9710.982946 | Epoch 92: Loss = 9706.039763 |
| 14 | Epoch 13: Loss = 9850.15511 | Epoch 33: Loss = 9755.274456 | Epoch 53: Loss = 9724.223616 | Epoch 73: Loss = 9711.54128 | Epoch 93: Loss = 9706.803385 |
| 15 | Epoch 14: Loss = 9842.440806 | Epoch 34: Loss = 9753.699656 | Epoch 54: Loss = 9725.123868 | Epoch 74: Loss = 9714.212076 | Epoch 94: Loss = 9703.494998 |
| 16 | Epoch 15: Loss = 9833.288707 | Epoch 35: Loss = 9750.68249 | Epoch 55: Loss = 9722.025756 | Epoch 75: Loss = 9710.106991 | Epoch 95: Loss = 9703.838888 |
| 17 | Epoch 16: Loss = 9826.939439 | Epoch 36: Loss = 9749.402241 | Epoch 56: Loss = 9721.02007 | Epoch 76: Loss = 9709.770157 | Epoch 96: Loss = 9704.762504 |
| 18 | Epoch 17: Loss = 9819.324565 | Epoch 37: Loss = 9746.351178 | Epoch 57: Loss = 9720.315394 | Epoch 77: Loss = 9709.421006 | Epoch 97: Loss = 9703.460662 |
| 19 | Epoch 18: Loss = 9814.05971 | Epoch 38: Loss = 9744.179939 | Epoch 58: Loss = 9719.707373 | Epoch 78: Loss = 9708.522328 | Epoch 98: Loss = 9703.763208 |
| 20 | Epoch 19: Loss = 9807.681563 | Epoch 39: Loss = 9742.278675 | Epoch 59: Loss = 9718.758556 | Epoch 79: Loss = 9709.302836 | Epoch 99: Loss = 9702.511405 |

Plot of Loss Over Epochs




3 (b) Tuning the learning rate:

I trained with following learning rate: 0.000, 0.001, 0.01and 0.1



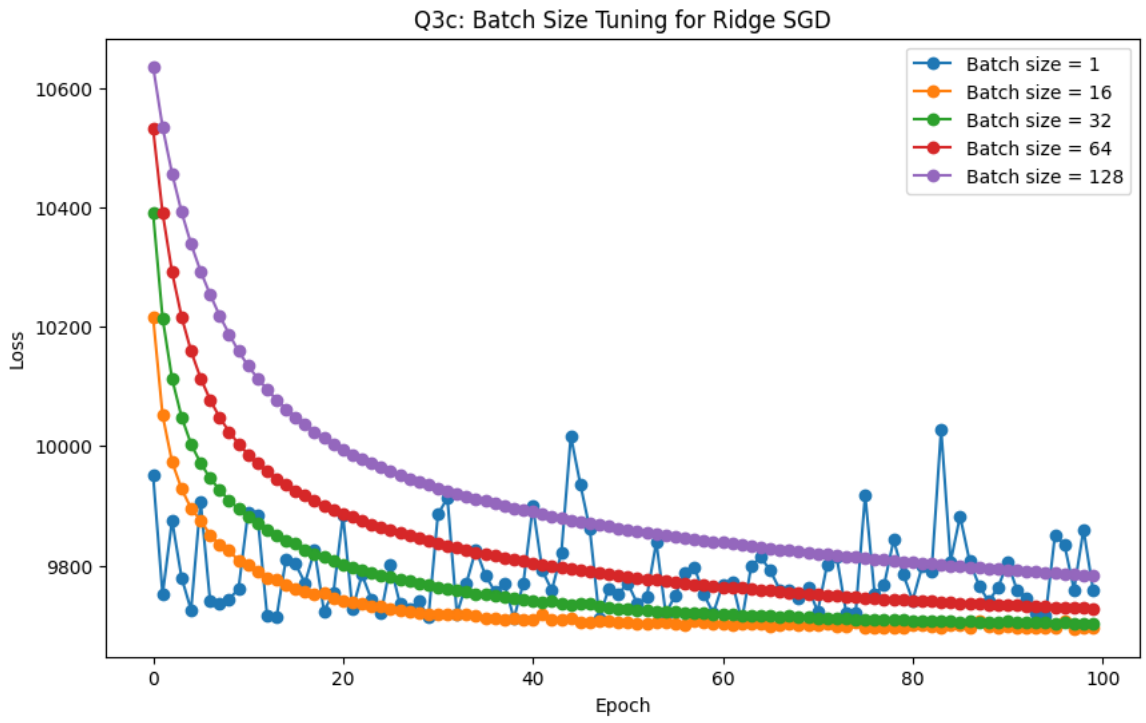
**Learning Rate Selection for Ridge Regression with SGD**
Four learning rates were tested (lr=0.0001lr = 0.0001lr=0.0001, 0.0010.0010.001, 0.010.010.01, and 0.10.10.1) over 100 epochs, and the results are visualized in the graph.

**Observations:**

- **lr=0.0001:** Converges very slowly, making training inefficient.
- **lr=0.001:** Steady and smooth convergence, optimal balance between speed and stability.
- **lr=0.01:** Faster convergence but minor oscillations in loss.
- **lr=0.1:** Significant instability and divergence, unsuitable for training.

**Conclusion:**
The best learning rate is **lr=0.001** due to its smooth and reliable convergence without instability. It provides an effective balance between training speed and model stability.

3(C)
I trained with Batch size:1,16,32,64,128



**Batch Size Tuning for Ridge Regression with SGD**

The impact of batch size on training was evaluated using batch sizes: 1,16,32,64,128 with the selected learning rate from **Q3b (lr=0.001)**. The loss was tracked over 100 epochs, as shown in the graph.

**Observations:**

- **Batch Size = 1:** Results in noisy loss curves due to high variance in gradient updates. While the model converges well, the instability may hinder reproducibility.
- **Batch Sizes = 16 and 32:** Achieve smooth and fast convergence with relatively low loss values. These sizes balance computational efficiency and effective gradient estimation.
- **Batch Size = 64:** Shows slower convergence compared to smaller batch sizes but maintains stability. Suitable for larger datasets or when computational resources are limited.
- **Batch Size = 128:** Converges the slowest and results in higher loss values initially. Larger batches can struggle to capture the nuances in the data due to lower update frequency.

**Conclusion:**

Batch sizes **16 and 32** are optimal as they provide a good trade-off between convergence speed and stability. Smaller batch sizes (e.g., 1) are prone to noise, while larger sizes (e.g., 64 and 128) reduce update frequency, slowing convergence.

RUN ALL CODES ON JUPITER NOTBOOK TO GENERATE RESULT