# Homework 3

Narendra Gautam Sontu 002241534

Link

Q1) 1. **Formula for Function fun**: The function **fun** in the script is defined as:

$$f(x)=1.3+2x_0-1.1x_1+0.7x_2+1.2x_3+0.4x_0^2-1.5x_1x_3-0.7x_4^2$$

This function depends on all coordinates of the vector $x \in R5$ (i.e., $x_0$ , $x_1$ , $x_2$ , $x_3$ , $x_4$ ). It is not a linear function due to the presence of quadratic terms ($x_0^2$ and $x_4^2$ ) and a product term ($x_1 x_3$ ).

2. **Description of generate_data.py**: The code generates a dataset of features and labels for machine learning purposes. It initializes a set of parameters, including the number of samples ($N$), the dimensionality of the feature vectors ($d$), and the noise variance ($\sigma$). It then generates a random $N\times d$ feature matrix $X$ and computes the labels $y$ by applying the function **fun** to each row of $X$ and adding Gaussian noise with variance $\sigma$. Finally, it saves the feature matrix and labels to files.

3. **Effect of Changing N to 2000:** Increasing N to 2000 would generate a larger dataset with 2000 samples instead of the original 1000. This would provide more data for training or testing in a machine learning context but would also increase the computational cost for processing and may require more storage space for the dataset.

4. **Effect of Setting d to 10:** Changing d to 10 alters the dimensionality of the feature vectors from 5 to 10. However, since the function fun explicitly depends on the first five coordinates of the input vector x, setting d to 10 would not affect the computation of y directly through fun, as fun only uses the first five dimensions ($x_0$ to $x_4$). The extra dimensions in $x \in R^{10}$ would not be utilized by fun and, therefore, would not influence its output.

Q2) 1. The RMSE formula relates to the training objective of linear regression by quantifying the average magnitude of errors between predicted values ($y$^) and actual values ($y$). Linear regression aims to minimize the sum of squared residuals, defined as

$$\sum_{i=1}^{n}(y_i-\hat{y}_i)^2$$

which is the Mean Squared Error (MSE). The RMSE is the square root of MSE,

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}$$

making it directly related to the least-squares estimate that linear regression minimizes. This minimization process results in the best-fitting line through the data, with the least amount of error.

2. sklearn.linear_model.LinearRegression:

Methods:

- fit: This method is used to train the linear model, using the given training data. It finds the coefficients (weights) for the features in the training data that minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation.

- predict: Once the model has been fitted, the predict method can be used to make predictions using the linear model's coefficients.

Attributes:

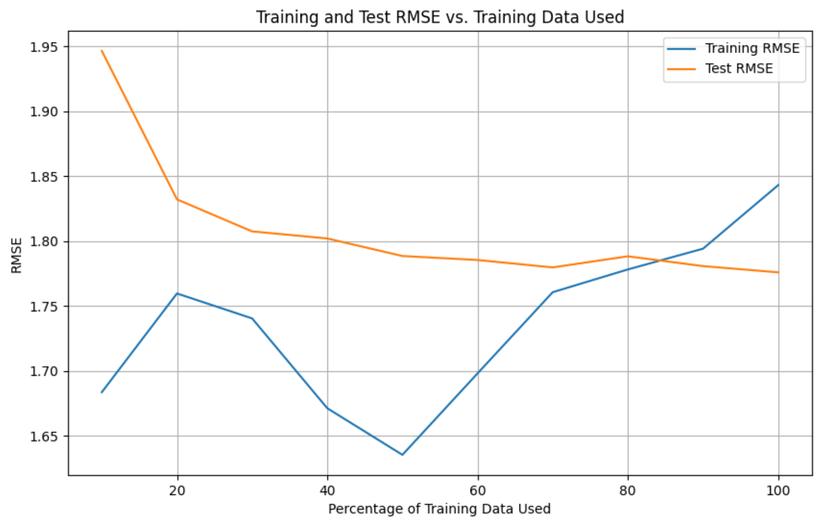
- coef_: This attribute holds the coefficients of the features in the linear model. These coefficients represent the values that multiply each feature value in the prediction equation.
- intercept_: This attribute represents the intercept of the model, which is the prediction value when all the feature values are set to zero.

3. The LinearRegression.py was modified it to create a new script called LinearRegressionSweepN.py that does the following:

- It reads a dataset generated by generate_data.py with N = 1000, d = 5, and σ = 0.01
- Splits this into a training and a test set, with a %70/%30 split ratio.
- Trains multiple ten different linear regression models, using only a fraction fr of the training set each time, where fr=10%, 20%, . . . , 100%.

For each fraction value, the code computes the RMSE of predictions both on the training samples used to train the model, as well as the RMSE on the entire test set.

Plotting the train and test RMSE as a function of the number of training samples given to model.



Based on the plot, increasing the number of samples beyond 1000 may not help significantly in lowering the errors because:

- The model has likely reached a point where it has captured most of the underlying patterns in the data, and additional data will not provide new information.

- The noise level (σ = 0.01) in the data generation may define a lower bound on the RMSE that cannot be improved by simply adding more data.
- The fluctuations in the RMSE, especially in the test set, suggest that the model's performance is affected by the variance within the data itself rather than the amount of data.

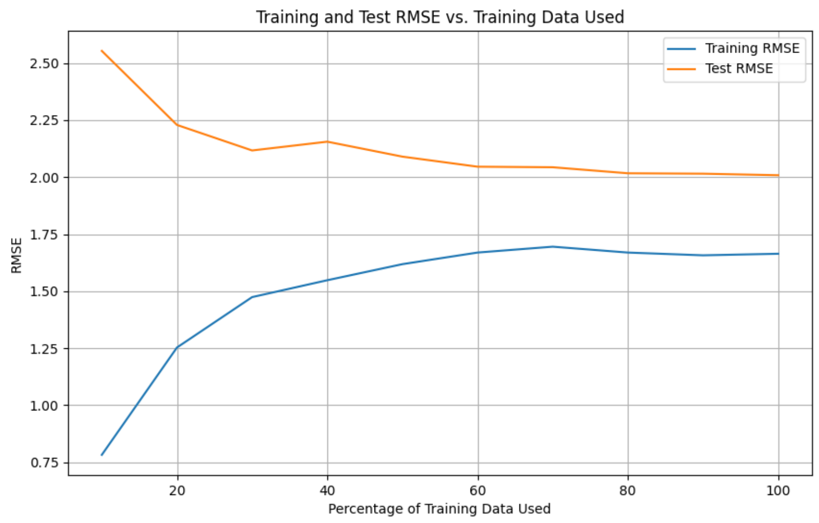
However, if the data has high variability or if the additional data introduces new patterns that were not present in the initial dataset, then it might still be beneficial to increase the sample size. It would also be important to consider other model complexities, feature engineering, or different types of models that might capture the data's structure better.

4. The LinearRegression.py was modified it to create a new script called LinearRegressionSweepN_2.py that does the following:

- It reads a dataset generated by generate_data.py with N = 1000, d = 40, and σ = 0.01
- Splits this into a training and a test set, with a %70/%30 split ratio.
- Trains multiple ten different linear regression models, using only a fraction fr of the training set each time, where fr=10%, 20%, . . . , 100%.

For each fraction value, the code computes the RMSE of predictions both on the training samples you used to train the model, as well as the RMSE on the entire test set.

Plotting the train and test RMSE as a function of the number of training samples given to our model.



Based on the plot, with d=40, we can observe that the training RMSE is quite low compared to the test RMSE, which is significantly higher and relatively flat across the percentage of training data used. This gap suggests that the model is overfitting the training data.
Unlike the first scenario where both errors decreased initially and then stabilized, in this plot, the test RMSE remains relatively constant and does not show an improvement as the percentage of training data

increases. This is typical of an overfitted model; it doesn't generalize well to new data, so adding more training data does not improve its performance on the test set.

Increasing the number of samples beyond 1000 is unlikely to significantly reduce the test error if the model is already overfitting with d=40. Instead of reducing the error, adding more data might just give the model more opportunities to fit the noise in the training set, rather than learning the true underlying pattern.

Given this behavior, the issue might not be the amount of data but the complexity of the model relative to the true underlying pattern and the noise level ($\sigma=0.01$). The model might be too complex to capture the actual signal in the presence of such a low level of noise, leading to overfitting.

In conclusion, with d=40, the model's complexity is too high for the given data, leading to overfitting. Adding more data is not likely to improve the test error unless other measures are taken to address the overfitting issue.

Q3) 1.

```
def lift(x):
    x_prime = [x[i] * x[j] for i in range(len(x)) for j in range(i + 1)]
    return np.array(x_prime)
```

2. Consider a function f(x) defined on the original vector x. If we want to represent this function as a linear combination of the elements in x′, we need to consider how each term in the function relates to the elements of x′.

- The terms in f(x) could be linear terms or interaction terms
- Since x′ includes both the original elements of x and their products, any function of x that includes linear terms and products of its elements can be directly mapped to x′.
- The vector β will have weights corresponding to each element of x′. For linear terms in f(x) that directly match elements of x, β will have non-zero weights where the terms correspond. For product terms xi ·xj , β will have non-zero weights at positions corresponding to these products in ′x′.
- If f(x) includes a constant term, this will be represented by c. Otherwise, c can be set to 0.

Consider the quadratic function f(x) defined on R^2:

$$f(x)=3+2x_0+x_1-x_0x_1+0.5x_0^2$$

Here, x is the original vector $[x_0, x_1]$. To represent this as a linear function of the lifted vector x', we first lift x to x':

$$x'=[x_0, x_1, x_0^2, x_0x_1, x_1^2]$$

Now, we can write f(x) as a linear function of x' with an appropriate vector β and constant c:

$$f(x')=\beta T x'+c$$

For our function f(x), β and c would be:

$$\beta=[2,1,0.5,-1,0]$$

$$c=3$$

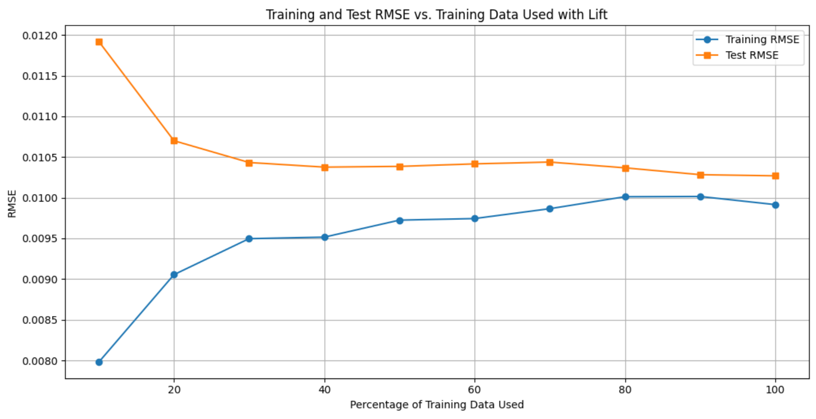
So the linear combination is:

$$f(x') = [2, 1, 0.5, -1, 0] \begin{bmatrix} x_0 \\ x_1 \\ x_0^2 \\ x_0 x_1 \\ x_1^2 \end{bmatrix} + 3$$

This shows that any quadratic function of x can be represented as a linear function of the lifted vector x', where β contains the coefficients of the corresponding terms, and c is the constant term from the original function f(x).

3.
```python
import numpy as np

def lift(x):
    x_prime = list(x)
    d = len(x)
    for i in range(d):
        for j in range(i + 1):
            x_prime.append(x[i] * x[j])
    return x_prime



def liftDataset(X):
    # Initializing an empty list to store the lifted rows
    X_prime = []

    # Iterating over each row in X and apply the lift function
    for row in X:
        lifted_row = lift(row)
        X_prime.append(lifted_row)

    # Converting X_prime to a matrix if required
    return X_prime
```

4. Modified LinearRegressionSweepN so that it lifts matrix X generated by generate_data right after reading it. Implemented on dataset generated by generate_data.py with N = 1000, d = 5, and σ = 0.01.



Observations,

Training RMSE:

- The plot with the lifted matrix (Q3.4) shows lower training RMSE values across all percentages of training data used compared to the plot without lifting (Q2.3).
- In the lifted plot, the training RMSE decreases sharply with the initial increase in training data percentage and then stabilizes.
- In contrast, the non-lifted plot shows more variability in the training RMSE, with a notable dip and subsequent increase as more data is used.

Test RMSE:

- The test RMSE in the lifted plot is relatively flat across the different percentages of training data used, whereas the non-lifted plot shows a general trend of decreasing test RMSE with more data, except for an increase from 90% to 100%.
- The lifted plot generally has lower test RMSE values, indicating better generalization performance.

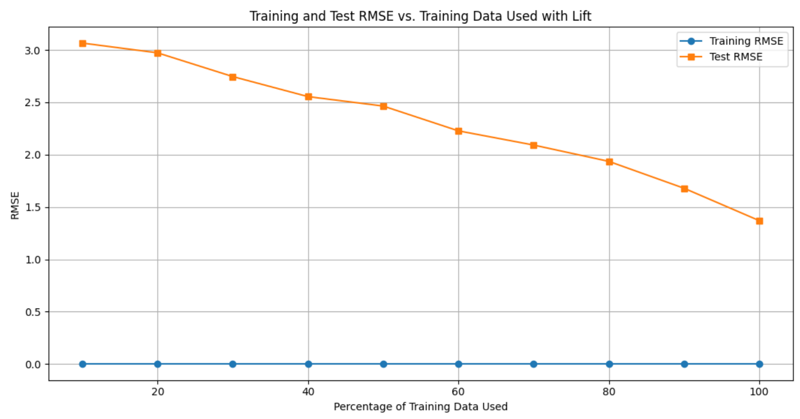
Why These Differences Occur:

- Lifting the matrix introduces additional features that capture interactions and non-linear relationships between the original features. This can enable the model to fit the data better and potentially generalize better if the underlying true relationship is indeed non-linear.
- However, the fact that the test RMSE is flat in the lifted plot could suggest that the model is not significantly improving with additional training data. This could be due to the regularization effect of Lasso, which may prevent overfitting by keeping the test error relatively constant.
- The initial sharp decrease in the training RMSE in the lifted plot suggests that the additional features are helping the model capture more of the variance in the data. However, as the model complexity increases with more training data, Lasso regularization likely plays a role in maintaining a balance between fitting the training data and maintaining generalization to the test data.

In summary, the lifting process seems to help reduce the RMSE, particularly on the training data, and to a lesser extent, stabilize the test RMSE. Regularization via Lasso likely contributes to preventing overfitting, which can be especially important when the number of features is large after lifting.

```
Final Model Coefficients: [ 1.99993561e+00 -1.09995902e+00  6.99675668e-01  1.20014825e+00
  4.37882617e-04  4.00142959e-01  1.16669521e-05  2.54883210e-04
 -5.82237321e-05  6.08512001e-04 -1.02522859e-04 -1.35417016e-04
 -1.49966623e+00 -2.22254061e-05 -3.04521986e-04 -7.28872468e-04
 -3.60964924e-04 -3.13564099e-04 -9.76089628e-04 -7.00110625e-01]
Final Model Intercept: 1.3001027369960907
```

5. Modified LinearRegressionSweepN so that it lifts matrix X generated by generate_data right after reading it. Implemented on dataset generated by generate_data.py with N = 1000, d = 40, and σ = 0.01.



```
Final Model Coefficients: [ 1.68936473e+00 -8.77450101e-01  5.81555233e-01  9.23714170e-01
  1.11079096e-01 -8.47598860e-02  5.09878257e-02 -9.79452362e-03
  1.86333958e-02  4.33182934e-02  1.25065585e-02 -3.91732451e-02
  2.39461715e-02  2.76801181e-02 -4.35568801e-02  1.28413798e-02
 -2.21817861e-02  4.10040145e-02  1.56370546e-02 -8.81852428e-03
 -1.38345892e-02 -2.20628251e-02 -3.37631043e-02  2.48041602e-02
 -1.60626405e-02 -2.04102238e-02 -5.62974736e-02  8.12637807e-02
 -2.48057396e-02  3.12106128e-02 -1.00154143e-02 -4.00248249e-02
 -5.58933940e-03 -3.16283762e-02 -4.65224102e-02 -9.55005774e-04
  7.99121706e-02 -8.21481101e-02 -1.00229792e-02 -1.80536601e-03
  3.47503927e-01 -2.45198615e-02 -2.03986862e-02 -1.70586109e-02
  5.87333365e-02  6.08266254e-03  5.65427809e-02  1.24844449e-02
 -2.70906453e-02  2.14160157e-02 -1.81732339e-03  2.96516358e-02
 -1.57593806e-02 -4.38184204e-02  1.80527964e-02 -9.53332901e-03
```

```
  9.90617180e-03  8.51893604e-02 -2.87877381e-02 -4.64505976e-02
  3.44452182e-02 -3.05865679e-02 -5.31760683e-02 -5.17359019e-02
 -2.82040387e-02 -3.45040918e-02  1.07110044e-02 -4.87224319e-02
 -9.49785733e-02 -1.21205609e-02 -7.02229634e-02 -2.14202804e-02
 -3.64394152e-02 -1.36832813e-02 -1.32166154e-02  2.94275148e-02
 -1.62139394e-03  3.31995800e-02  3.12005339e-02 -2.71324541e-03
  8.56926935e-02  8.01922920e-02 -1.07496150e+00 -1.27015290e-02
  3.76198795e-02  1.95722344e-02  4.75692871e-02 -3.32550786e-02
  4.54866790e-02 -3.95925175e-02 -2.59457622e-02  5.62840107e-02
 -2.12602121e-02 -2.52005938e-02  7.00208759e-02  6.87414885e-02
  7.36852588e-02  3.60778741e-02 -7.55668728e-02 -6.00521983e-03
  4.05069902e-02 -6.86219851e-02  2.33074614e-02 -2.68521908e-02
 -5.73403872e-02 -6.12979684e-02 -6.61499654e-03 -1.08832281e-02
 -1.17309181e-01  1.18924263e-02  5.80680798e-02 -7.36577164e-03
 -2.64822316e-02 -2.87167222e-02 -7.72805485e-02 -3.94782367e-02
  1.98437606e-02 -8.41905297e-02  1.79589770e-02  4.11024313e-03
 -7.36737757e-02 -5.70342704e-02  1.19948580e-02 -2.35491365e-02
 -9.40544841e-02  1.56341649e-02  5.52981148e-02 -2.57919617e-02
  2.37625242e-02  7.24304352e-02 -1.63044284e-02  8.25168370e-03
  9.90071512e-02 -3.50814541e-02  2.88422132e-02  6.87867503e-02
 -3.46935026e-02  1.42281260e-02 -2.90033071e-03 -2.21621727e-02
  5.49767485e-02 -3.99877701e-02 -1.16111366e-02 -4.35228899e-02
 -2.11661791e-02 -5.80601025e-02  4.20031744e-02 -1.18154096e-01
 -1.13337886e-02  2.23431187e-02 -9.79826849e-02 -5.24390796e-02
 -4.68482342e-04  9.25214709e-02 -1.75319457e-03 -6.76172158e-02
  5.76734660e-02  6.25920819e-02  3.65721774e-02 -2.99198571e-02
  6.36836495e-02  5.89718984e-03 -2.52977789e-02  1.73426635e-02
  8.22206071e-03  1.10699945e-01  3.94451211e-02  4.75010982e-02
  5.30260624e-03  6.13437111e-02 -2.77721174e-02 -2.62311912e-02
 -1.56410907e-02  2.42058052e-02 -3.98778804e-03  2.46900343e-02
 -3.72690048e-02  9.79578379e-03 -6.57797633e-02 -3.39040809e-02
 -2.02601357e-02  2.66569802e-02  1.03246269e-02  3.71384354e-02
 -5.29233368e-03  5.53812960e-02 -1.31538011e-02  6.96389601e-02
  7.25853614e-02 -3.74875110e-02  1.00799198e-01 -6.69110282e-02
 -1.57888157e-02  1.43877332e-02 -6.19399233e-01 -3.57560676e-02
  1.81749032e-02 -3.13801936e-02 -1.96012596e-02  2.73725764e-02
 -2.91299932e-02 -4.96215258e-02 -3.27197956e-02  1.64626848e-02
  2.32967751e-02  4.04524321e-02  2.55832840e-02  6.06116955e-02
  2.57511858e-02 -5.76620027e-02  6.73171034e-02  5.46447754e-02
 -1.43953888e-02  6.72872206e-02 -6.10179660e-03  1.13567975e-02
 -2.97032215e-02  4.21037930e-03  1.09705627e-03 -3.27738447e-02
 -1.64722711e-03  1.95093609e-02 -5.69307586e-03 -2.74919390e-02
 -8.64512865e-02  1.15367067e-01  3.83800251e-02  6.77256816e-02
 -3.58402997e-03  7.29615077e-02 -7.84544111e-02 -1.91595008e-02
```

```
  7.52781625e-02 -4.78333595e-02  1.78430077e-02 -5.09164688e-02
 -1.81891452e-02 -4.36673275e-02 -1.26448570e-02  9.64631775e-03
 -4.17881547e-04 -1.95876864e-02 -8.61575608e-03  1.84316839e-02
  8.08705614e-02 -6.99339303e-02 -2.99397361e-02 -3.36597000e-03
  2.52443997e-05  8.42027166e-03 -5.40916761e-02  4.58628489e-02
  2.56321328e-02  3.54243806e-02  5.31340302e-02 -4.12078926e-02
 -2.41521704e-02 -6.70714345e-02 -1.59786813e-02  4.82072995e-02
 -3.28087470e-02  3.83103300e-02  6.25903366e-02  1.21686719e-02
  2.97599916e-02 -2.77045831e-02  6.77761747e-02  8.13097746e-02
  7.37903131e-02 -1.92170115e-02  4.98074526e-02  2.21314694e-02
 -5.53737681e-02 -4.06312480e-02  5.89515188e-02  4.78831848e-02
  2.60707580e-02 -3.84845385e-03  2.35079576e-02  3.90127075e-02
 -2.65555691e-02  6.09055344e-02  9.74617258e-03  1.44278051e-03
  4.81817207e-02 -9.04929663e-02  1.21620642e-02  1.62430470e-02
  6.65027613e-02  4.31220360e-02 -4.64731431e-02 -6.74615771e-02
 -2.28262689e-02 -3.97448885e-03  1.95440482e-02  5.98718019e-02
 -6.62596828e-03  4.19151765e-03  4.95366050e-02 -3.12989735e-02
  1.18064755e-02 -1.63658951e-03 -1.24201975e-02 -1.43549518e-02
  1.15619036e-02  1.49552675e-02 -6.26852959e-02 -5.92234029e-03
 -1.63775287e-02  5.72905065e-02  4.26746351e-02 -3.24860002e-02
 -6.85533803e-02  2.69067728e-02  2.74932853e-02  2.09061747e-02
  1.07050810e-02 -6.01689616e-03 -3.20026233e-02  2.83680111e-02
  2.76545510e-02 -1.17042137e-02  9.89797002e-03  3.48793265e-02
 -1.67979491e-02  5.08033859e-02  4.32437156e-03  5.39104689e-02
  5.06675121e-03 -1.19973070e-02 -2.50232204e-02 -2.75528985e-02
  1.19941840e-02 -2.06832633e-02  5.77401094e-03  3.18883406e-02
 -7.63095883e-02 -2.38140317e-02 -6.20026467e-02  8.73191224e-03
 -1.48803428e-03  2.31440070e-02 -3.26171909e-02 -1.02953587e-01
 -1.54597291e-02 -1.21365490e-02 -8.73213306e-02  4.02237073e-02
  6.78792736e-04 -4.44945471e-02 -4.60174290e-03 -1.83216570e-02
  3.63585634e-02  2.31691019e-02  2.48734958e-02 -3.23671550e-02
 -2.52439277e-03  5.49775427e-02 -6.16588501e-02  1.54414896e-01
 -3.73621883e-02 -6.81008456e-03 -1.56903616e-02  7.81851764e-02
  4.41235202e-02  7.13337387e-03  3.60641208e-02 -3.54734041e-02
 -2.15512686e-02 -4.52069440e-02  3.32416358e-02  2.06969261e-02
  5.51595135e-03  4.96885055e-02 -3.42126935e-02  3.84055457e-02
 -5.96852796e-02 -3.81575045e-02  1.27397298e-02 -1.15554876e-02
 -3.44180483e-02 -7.12418618e-03 -2.96078509e-02  1.31254503e-02
  7.80740545e-03 -2.76786845e-02  2.17081738e-02 -4.44241025e-02
  6.24686813e-04 -1.16299832e-02 -6.59417209e-02 -2.95454990e-02
  5.10295381e-03  8.00026491e-02  5.95827711e-02  1.45531443e-02
  6.78687247e-02 -4.00481693e-02  4.71987697e-02 -1.93797296e-02
  4.60131878e-02 -3.58776879e-02  2.41479941e-02  8.60588605e-03
  1.84864922e-02 -1.16524130e-02 -2.93396133e-02 -6.65706342e-02
  3.31526733e-02 -2.43893577e-02  5.44502047e-02  7.64518574e-02
 -6.09052740e-02  3.13016291e-03  5.99358711e-02 -5.71134780e-03
 -3.57284076e-02  1.93856489e-02  1.96283679e-02  2.79496586e-02
 -3.76375483e-02  8.37194321e-03  1.49487877e-02  4.25417582e-02
 -4.06888615e-02 -5.94213083e-04 -7.01200897e-02  5.61823286e-02
 -4.08684355e-02 -3.93714137e-02 -2.30423343e-02 -5.84372150e-02
  1.29644823e-02 -6.91098652e-02 -2.16679826e-02  5.10618071e-04
  3.66676436e-02 -5.82112110e-02 -3.50608217e-03  3.04775004e-02
 -5.04788301e-02  2.60444097e-02  2.40259960e-02 -3.55945769e-02
 -1.83948614e-02  2.46329479e-02 -2.36141384e-02  1.84269072e-02
  8.40668031e-03  5.33196657e-03 -4.37680435e-02 -1.14122416e-01
  8.27490087e-02 -9.14047307e-02 -4.05090084e-02  9.49370310e-02
  5.00750524e-02  9.07444954e-03 -1.93315530e-02 -2.07368779e-02
 -2.05931589e-02 -3.24414883e-02 -1.09601655e-02 -6.93826507e-03
 -5.67489306e-02  6.98383968e-02  3.23980124e-02  1.38467552e-02
 -2.43638576e-02  4.61594897e-02 -1.03014224e-03  1.39573623e-02
  2.23173769e-03  2.43791853e-02 -6.26947910e-02  4.52495227e-02
 -2.14129219e-02 -1.47308129e-02 -9.19723995e-03 -2.34605928e-02
 -6.44053230e-02  1.61113073e-02 -2.07719578e-02 -2.61331541e-02
  9.10495459e-02  2.74081075e-02 -5.95468239e-02 -4.47412097e-02
 -3.88522417e-02  3.57650658e-02 -6.89605571e-02  3.45893940e-02
  5.71253833e-02  9.84534994e-02  7.17472423e-02  5.16780704e-02
 -6.03525582e-03  5.08797062e-03 -7.17414048e-02  1.04472990e-01
  2.29570100e-02  4.13412392e-02 -1.01462262e-02  1.35873377e-02
  2.02834923e-02  6.96536714e-03 -5.56335588e-03 -1.66224860e-02
  1.29188605e-02  3.42100218e-02 -1.20003455e-02  2.34673898e-02
  1.03467206e-01  4.47956693e-02  3.50368871e-02  3.61680263e-02
  1.49102005e-02  4.33507027e-02 -9.24021794e-03 -2.91244765e-02
  9.48250202e-02 -1.74206268e-03  4.55829030e-02 -3.78495516e-02
  7.37440584e-02  3.74293592e-02 -1.44480001e-02  4.93817504e-02
  2.37468737e-02  1.69055020e-03 -9.19196005e-03 -4.43570738e-02
  3.06057672e-02 -9.02638929e-02 -9.26801038e-03  2.55211432e-02
  1.64583022e-02  8.54572218e-03  5.56516846e-02 -5.14889947e-03
 -2.31530126e-02 -3.26673613e-02 -4.01147282e-02  2.65006830e-03
  6.99880993e-03 -8.08327352e-02 -3.25991873e-02  3.59263743e-02
 -9.80031207e-03 -2.25761558e-02  4.81389747e-03  2.16973693e-02
  7.25782658e-02 -3.50027137e-03 -8.58374781e-03 -2.24419799e-02
 -6.90276071e-03 -5.99227603e-04  5.49999276e-02 -5.19617195e-02
 -1.10506248e-02 -6.14056034e-02  5.10421477e-03 -1.32087849e-02
 -5.83515461e-02 -7.84542524e-02 -1.42729406e-02  1.56778279e-02
 -3.15243298e-02  5.26361450e-03  3.04433147e-06 -7.87173036e-02
 -1.23740328e-02 -2.50245294e-03 -3.06029191e-02  1.81416946e-02
  5.78932995e-02  2.21121145e-02  4.96501064e-02  3.85978489e-02
  9.79573587e-03 -5.85763578e-02  1.44777146e-02  1.08739738e-02
```

```
-4.77224179e-02  1.43573379e-02 -3.50854564e-02  3.96163284e-02
  4.78258286e-05 -9.17773323e-04 -1.60676618e-02 -2.87812637e-02
-8.89390231e-02 -6.26680795e-02 -6.71029124e-02 -7.44685899e-02
  3.14874859e-02  2.93824971e-02  3.53913935e-02 -3.05471006e-02
-3.96852459e-02 -1.20199495e-02 -1.29035435e-02 -1.48124568e-02
  3.55575646e-02  1.00289555e-02 -8.41952708e-04 -2.37384093e-03
-7.88075770e-02 -2.92175036e-02  8.23529698e-02  3.08878986e-02
-3.53594775e-02 -2.23594414e-02 -7.57440823e-02  1.80996739e-02
-2.51366676e-02 -2.82460850e-02  5.33412090e-02 -3.08217220e-02
-8.80832211e-03 -2.61798474e-02  2.56490654e-04  2.91103295e-02
  2.51476394e-03  6.23864449e-02  5.00124249e-03 -6.99696525e-04
  1.69912286e-01 -4.40589308e-02 -6.29095724e-02  7.90359911e-03
  2.49306711e-03 -4.51077105e-02  6.37122021e-03 -2.41273022e-02
  6.22232885e-02 -5.75569565e-02 -4.62788400e-02 -1.72842804e-02
  2.11485975e-02 -1.52726918e-02 -6.51233540e-02  3.84877230e-02
  4.80116499e-02 -4.47054821e-02 -1.66725258e-03  1.28153991e-02
-1.15155274e-01 -4.96865326e-02 -2.79591417e-02  1.85772371e-02
-3.69449069e-02  7.41945836e-03  3.76688397e-03  1.06150227e-02
-6.39988053e-02  5.61546436e-03 -1.39062552e-03  8.07620956e-02
  4.69212160e-02  4.79560604e-03 -9.81832948e-03 -4.90603031e-02
-8.42170258e-02  6.17987752e-02  3.50390612e-02  2.40790847e-02
-8.09238246e-03 -6.01544141e-02  2.96947475e-02  1.48070953e-03
  7.67354311e-02  4.23145615e-02 -6.55445633e-02 -1.89405147e-02
-3.45230639e-02  8.14842258e-03 -3.23914087e-02 -1.64787817e-02
-2.08577818e-02 -7.74493540e-02 -8.47634091e-02  2.22040052e-02
-6.08950424e-02  7.43701205e-02  1.76504871e-02  8.04175904e-02
  4.45454949e-02  2.59535341e-03  6.55732726e-03 -2.35109947e-02
  2.08470144e-02  4.43586977e-02  1.94422459e-03  7.86856610e-03
-6.68815465e-03  3.13157923e-02 -1.76282399e-02 -5.13637529e-02
-5.46107815e-02  2.71987533e-02 -1.37691218e-02 -3.21067058e-02
  4.49361372e-02 -3.62241939e-02 -5.21517020e-02 -3.11069571e-02
  1.91034405e-02 -3.90630869e-02 -1.21221495e-03  2.75241437e-02
  9.03782473e-02 -8.18161035e-02 -4.02502338e-02 -1.60297317e-04
-1.53394586e-02 -3.56776364e-02  1.37746528e-02  4.92355969e-02
-1.53799748e-03  6.15542827e-02  5.09288487e-02  5.96208931e-03
-4.14428499e-02 -1.05875072e-01 -4.67056254e-02 -2.47306699e-02
-1.06628585e-03 -1.83431568e-02  2.94899765e-02 -6.95921651e-03
-5.88980612e-03  6.89297986e-02 -7.94280543e-03  1.76058467e-02
  4.28620159e-02  1.08508124e-02  1.64323690e-02  7.56702328e-02
-7.28911436e-03  5.44146471e-02 -3.38475372e-03 -1.63854236e-02
  4.55916076e-02  3.49227887e-02  3.18100150e-03  1.42977661e-02
  1.45103542e-02  1.58229029e-02  4.15641880e-02  4.92437428e-03
  1.67002835e-02 -5.73168471e-03 -8.49009947e-02 -2.72197254e-03
  2.72164079e-02  3.41151753e-02  1.85716636e-02  6.95874392e-02
  6.19519665e-02 -7.27363508e-02 -1.98736389e-02  5.68521630e-02
 -7.99546879e-02 -4.78547906e-02 -1.09711068e-02 -4.12537676e-02
  7.15298510e-02  1.30439660e-02 -4.44852151e-03  9.28158421e-03
  7.32472664e-02 -1.27350284e-02  8.54511073e-03 -7.93586393e-03
 -2.68005453e-02 -1.81457788e-02 -1.97854776e-02  4.33593066e-02
 -2.68273229e-02 -3.15135191e-02  7.05242718e-03 -3.82320732e-02
  4.38079437e-02 -6.91116323e-03  3.47575149e-02 -1.02488007e-01
 -2.63868141e-02  8.85126878e-03  2.31958662e-02  8.05287127e-04
  1.70826310e-02 -2.72259195e-02  7.30518750e-02 -2.85815132e-02
  6.49064772e-02  5.46350519e-03  9.97653367e-02  1.82765078e-02
  5.14814111e-02  1.85966821e-02 -2.44531831e-02 -3.33735666e-03
 -2.13609454e-02  1.91566314e-02 -3.74391636e-03  1.28946182e-02
  4.48340819e-02 -2.21940081e-03 -3.49587442e-02  1.21357835e-02
  3.58292215e-02 -1.53867917e-02 -2.64984286e-02  1.78075235e-02
 -7.75671764e-02  7.20214457e-03  5.87131008e-02 -7.93803553e-02
 -3.47515349e-03  4.18363710e-02 -1.91486528e-02 -5.26435572e-02
  1.09534494e-02 -1.97553518e-02  8.20873311e-04  3.54203586e-02
  3.35542808e-02 -6.07936690e-03  5.29720893e-02  4.19516995e-02
  9.54817562e-03  7.81108497e-02 -3.41680764e-02  4.93994956e-02
  2.83951604e-02 -3.57463175e-02  1.61744220e-02 -1.27693754e-02
  3.22561898e-02 -3.55644883e-02  3.64245714e-02 -9.89998264e-02
  4.76662013e-02  2.61641169e-02  3.45562984e-02  6.73224033e-02
 -9.21421120e-02 -5.51773726e-02 -1.01258990e-02  4.86827163e-02
  5.02497913e-02 -1.95634080e-02 -3.81626265e-02 -7.89989152e-02
 -3.35593293e-02  5.28812026e-02  2.22009800e-02  4.81356916e-03]
Final Model Intercept: 1.2024726260703398
```

Increasing the number of samples beyond 1000 may improve the test error and the model's approximation to the true function fun, particularly if the existing data does not capture the full variability of the underlying process. However, the benefits of additional data tend to diminish after a certain point, especially when the model already has a low RMSE and generalizes well. It is also important to consider computational costs and the potential for diminishing returns on model performance with more data.

Q4) 1.

Loads the input features X and output labels y from .npy files.and prints the dataset dimensions for samples and features, and the number of labels.

Split Dataset:

- Splits the dataset into training and test sets with a 70/30 split ratio using train_test_split.

Model Initialization and Cross-Validation:

- Initializes a Lasso regression model with a regularization parameter alpha.
- Sets up a K-Fold cross-validator with 5 splits, a fixed random state for reproducibility, and shuffling enabled.

- Performs cross-validation on the training data and computes the negative root mean squared error for each fold.
- Prints the mean and standard deviation of the cross-validation RMSE scores for the given alpha.

Model Training:

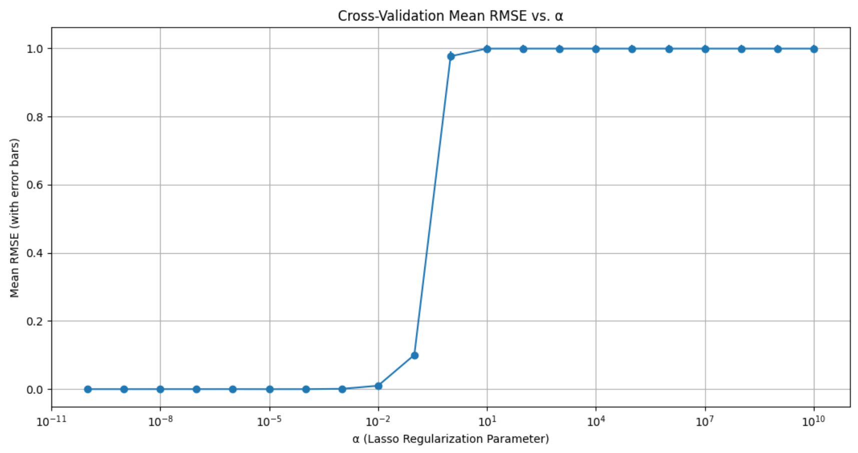
- Fits the Lasso model on the entire training set.

Model Evaluation:

- Calculates and prints the RMSE on both the training and test sets to evaluate the performance of the model

2. Running your code on a dataset generated by generate_data.py with N = 1000, d = 40, and σ = 0.01 and plotting the cross-validation mean RMSE as a function of α

Optimal α: 1e-05
Train RMSE: 1.0342798979866096e-05, Test RMSE: 1.0317869037422895e-05

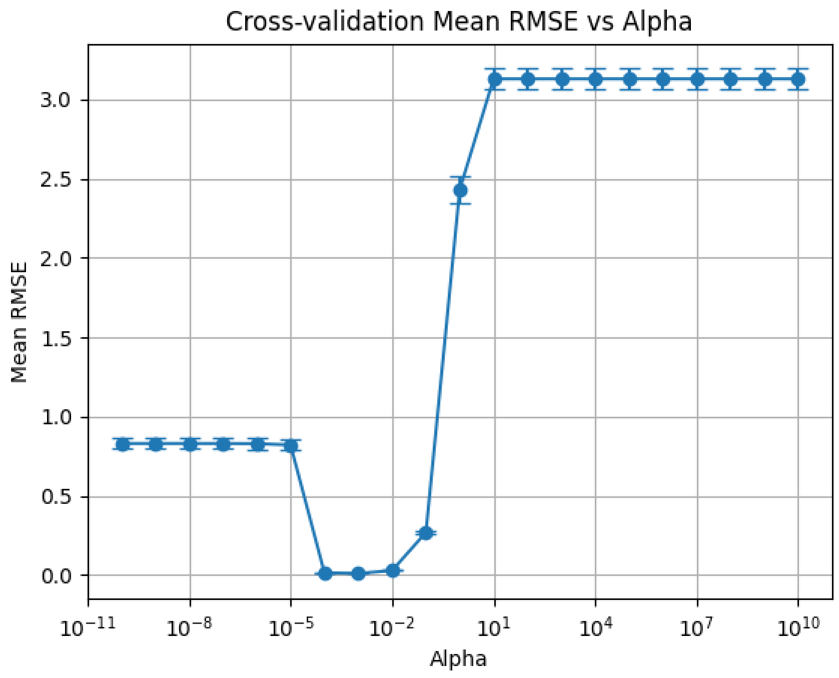


The Mean RMSE during cross-validation is minimized for an optimal α value of $10^{-5}$. This indicates that a small amount of regularization is beneficial for the model, possibly by reducing overfitting.

As α increases beyond $10^{-5}$, the RMSE remains relatively constant. This plateau suggests that further increasing the regularization strength does not significantly affect the model due to either the data already being well-fitted or because the Lasso regularization is forcing too many coefficients to zero, hence not improving the model's performance.
The very low RMSE values on both the training and test sets indicate that the model is performing exceptionally well. This could be a result of the high-dimensional lifted feature space capturing the underlying data structure effectively.

The consistency of RMSE values between the training and test sets suggests that the model is generalizing well and not overfitting to the training data, which is supported by the optimal regularization parameter found.

3.



```
Train RMSE with optimal alpha 0.001: 0.0100093528878736
Test RMSE with optimal alpha 0.001: 0.010200695839430245
Parameters with absolute value larger than 1e-3: [ 1.2989214   1.99909155 -1.09949989  0.69863278
1.19921516  0.39982011
-1.4993236  -0.698939  ]
```

So, based on the output and the plots, it appears that the learning process for the function fun has improved in Q4.3 compared to Q3.5 despite the presence of nuisance variables.

Because without regularization, there could have been a risk of overfitting, especially with a high-dimensional feature space after lifting. However, with Lasso regularization, the model can ignore irrelevant features (nuisance variables), which might have caused overfitting or distraction from the true relationship.
The improvement in learning fun can be attributed to the combination of feature lifting and Lasso regularization. Feature lifting expands the feature space to include interactions that may capture the complexity of fun, and Lasso regularization then selects the most relevant features, reducing the impact of nuisance variables and preventing overfitting. This leads to a model that not only performs well on the training data but also generalizes well to unseen test data.