University of Tehran

School of Mechanical Engineering

Artificial Intelligence

Home Work 2

Professor:

Dr. Masoud Shariat Panahi

Author:

Masoud Pourghavam

April, 2023

Table of Contents

List of Figures 2
List of Tables 3
1. Tumor type prediction 4
A. Missing data & outlier detection 4
i. Detection of missing values and outliers 4
ii. Changing 2 and 4 values 6
iii. Normalizing the data between 0 and 1 6
B. Data distribution 7
C. Parameters and tumor type relation using heatmap 9
D. Validation, testing, and training data 10
i. Splitting data 10
ii. Logistic regression and k-fold cross validation 11
E. Confusion matrix 11
F. Analysis of false predictions 13
2. Service life prediction of dielectric material 15
A. Regression using 4 kernels 15
B. L2-Regularization 17
C. Change regularization parameters and use gridSeachCv 18
D. Fit the function to data in MATLAB 20
E. Using Cftool in MATLAB 22

List of Figures

Fig 1. Box plot of outlier detection 5
Fig 2. Printed results of outlier detection 6
Fig 3. Results of values replacement 6
Fig 4. Results of normalization 7
Fig 5. Box plot of data distribution 7
Fig 6. Histogram plot of data distribution 8
Fig 7. Number of tumor types 8
Fig 8. Total distribution of data 9
Fig 9. Heatmap representation of the correlation matrix 10
Fig 10. Illustration of confusion matrix concept 11
Fig 11. Confusion matrix of our model 12
Fig 12. Regression with linear kernel 15
Fig 13. Regression with RBF kernel 16
Fig 14. Regression with 2nd order polynomial 16
Fig 15. Regression with sigmoid kernel 17
Fig 16. Nonlinear regression with fitted function 21
Fig 17. Cftool fitting curve app 22
Fig 18. Regression with new function fitting 23

List of Tables

Table 1. Libraries used in section 1. 4
Table 2. Missing values. 5
Table 3. Number of samples in training, validation, and testing sets. 10
Table 4. Factors calculated for confusion matrix. 12
Table 5. Libraries used in section 2. 15
Table 6. R2-score and MAE for 4 different kernels using k-fold cross validation 17
Table 7. R2-score and MAE for 4 different kernels using R2-regularization 17
Table 8. R2-score for linear kernel with different regularization parameters 18
Table 9. R2-score for 2nd order poly kernel with different regularization parameters 18
Table 10. R2-score for 3rd order poly kernel with different regularization parameters 19
Table 11. R2-score for 4th order poly kernel with different regularization parameters 19
Table 12. R2-score for RBF kernel with different regularization parameters 20

1. Tumor type prediction

In this section, a dataset with the format of *CSV* containing the data of 684 patients diagnosed with breast cancer is used. In this dataset, each sample contains 9 tumor characteristics in columns 1 to 9 and an output containing tumor type in column 10. In column 10, there are two types of tumors, the number 2 represents a benign tumor and the number 4 represents a malignant tumor. In the following, we will try to satisfy the demands of the question by using the Python programming language. For coding, a 3.7 Python interpreter has been used. Also, the libraries used in this section, are given in Table 1.

Table 1. Libraries used in section 1.

No.	Library title
1	pandas
2	numpy
3	matplotlib
4	sklearn
5	seaborn

A. Missing data & outlier detection

i. Detection of missing values and outliers

To get started, we read the dataset using pd.read_csv command. After reading the data, we can check if there are missing data and also detect the outliers. It is important to find the missing values and outliers because missing values affects the accuracy of the result and also may lead to biased results. Outliers are the abnormal values that happen to have deviated from the normal distribution pattern of data distribution. We checked the number of missing values in each column and the results are given in Table 2.

Table 2. Missing values.

Column	Characteristics	Missing data
1	Clump Thickness	0
2	Uniformity of Cell Size	0
3	Uniformity of Cell Shape	0
4	Marginal Adhesion	0
5	Single Epithelial Cell Size	0
6	Bare Nuclei	0
7	Bland Chromatin	0
8	Normal Nucleoli	0
9	Mitoses	0
10	Class	0

To show the outliers existing in data, we used 2 different ways. First, we used a box plot shown in Figure 1 which represents the data distributions and the outliers existing in dataset. Then, the outliers in each column are printed with their row number. For instance, a part of the printed results in Python ide is given below. You can see the full printed results after running the code.

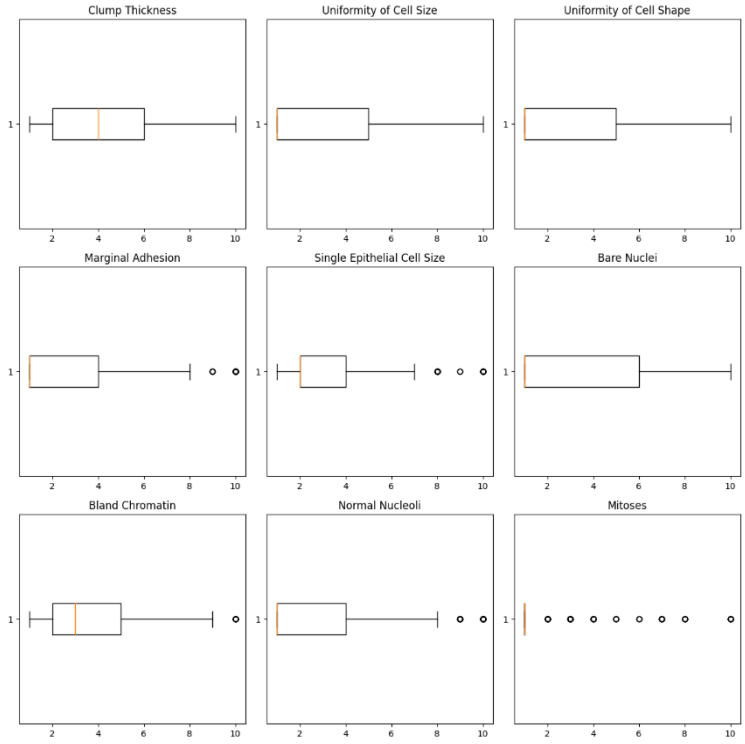


Fig 1. Box plot of outlier detection

```
Outliers in Clump Thickness: []
Outliers in Uniformity of Cell Size: []
Outliers in Uniformity of Cell Shape: []
Outliers in Marginal Adhesion: [14, 20, 37, 71, 97, 102,
Outliers in Single Epithelial Cell Size: [31, 40, 41, 42
Outliers in Bare Nuclei: []
Outliers in Bland Chromatin: [167, 208, 253, 276, 410, 4
Outliers in Normal Nucleoli: [21, 35, 42, 50, 54, 58, 61
Outliers in Mitoses: [8, 14, 18, 20, 31, 39, 40, 43, 47,
```

Fig 2. Printed results of outlier detection

As you can see in Figure 2, there are no outliers in *Clump Thickness*, *Uniformity of Cell Size*, *Uniformity of Cell shape*, and *Bare Nuclei* columns.

ii. Changing 2 and 4 values

Then, we replace the numbers 0 and 1 with the numbers 2 and 4 using the following command, and in this way, benign tumors are identified with the number 0 and malignant tumors with the number 1. So, for instance, in the first 5 rows of dataset, we will have the results in Figure 3.

```
df["Class"] = df["Class"].replace({2: 0, 4: 1})
```

	Clump Thickness	Uniformity of Cell Size	...	Mitoses	Class
0	5	1	...	1	0
1	5	4	...	1	0
2	3	1	...	1	0
3	6	8	...	1	0
4	4	1	...	1	0

Fig 3. Results of values replacement

iii. Normalizing the data between 0 and 1

To normalize the data in columns 1 to 9, we will use the following commands and as a result, the first 5 rows of dataset after normalizing are given in Figure 4.

```
scaler = MinMaxScaler(feature_range=(0, 1))

df.iloc[:, :9] = scaler.fit_transform(df.iloc[:, :9])
```

	Clump Thickness	Uniformity of Cell Size	...	Mitoses	Class
0	0.444444	0.000000	...	0.0	0
1	0.444444	0.333333	...	0.0	0
2	0.222222	0.000000	...	0.0	0
3	0.555556	0.777778	...	0.0	0
4	0.333333	0.000000	...	0.0	0

Fig 4. Results of normalization

B. Data distribution

Here, we will use some plots to show the data distribution. The basic advantage of data distribution is to estimate the probability of any specific observation in a sample space. In Figure 5 the data distribution is shown for each column with different colors.

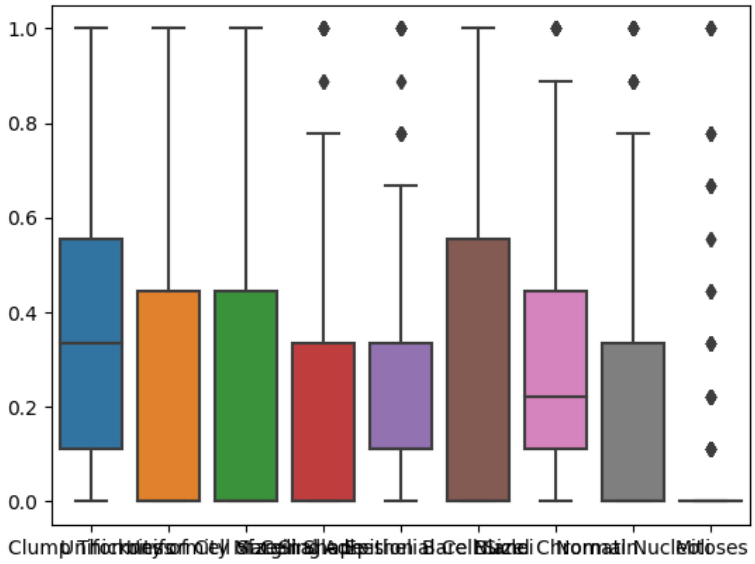


Fig 5. Box plot of data distribution

In another way, we used a histogram plot given in Figure 6 to show the distribution of each value between 0 and 1 in each column. Also, the amount of data with different type of tumors available in column 10, are shown in Figure 7. Where the blue color represents the benign tumor and the red color represents the malignant tumor. Finally, as shown in Figure 8 we used a command to show the total distribution of all the values, in all the columns 1 to 9.

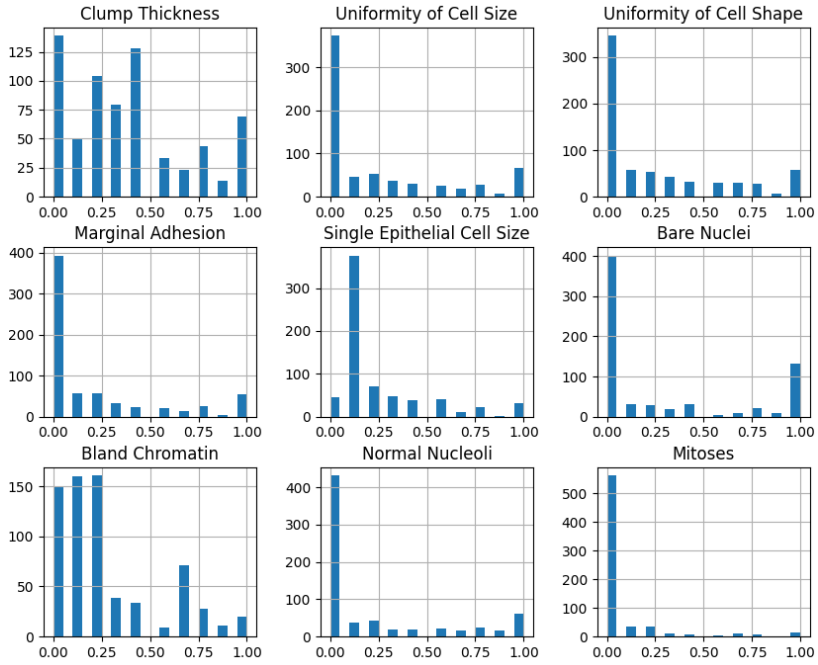


Fig 6. Histogram plot of data distribution

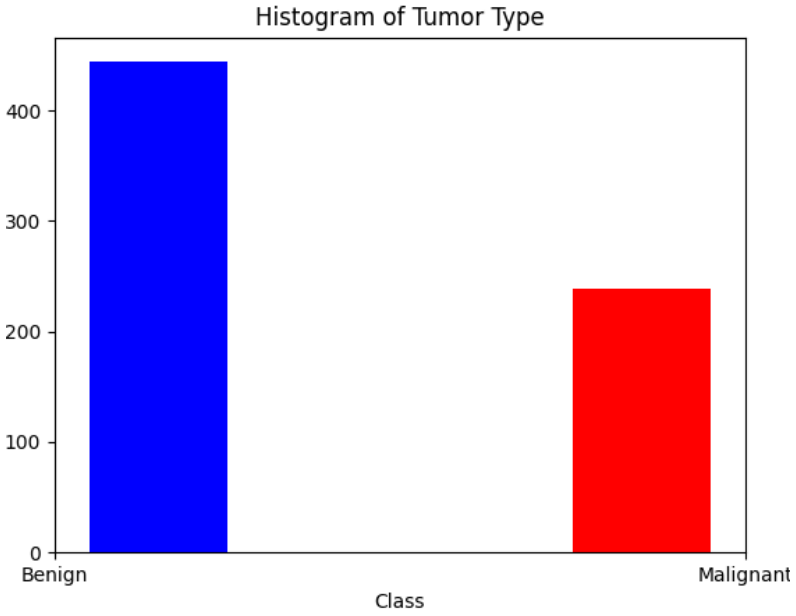


Fig 7. Number of tumor types

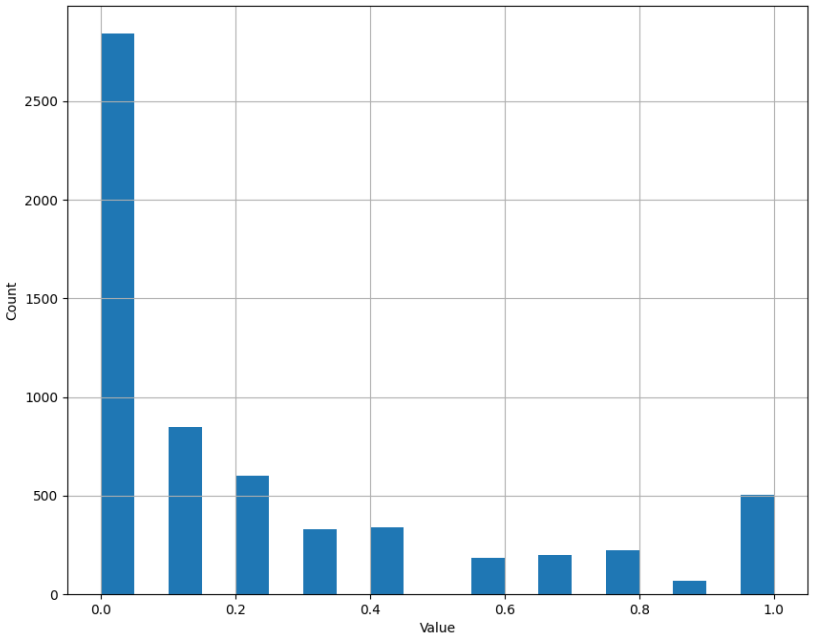


Fig 8. Total distribution of data

We can see from the results in Figure 5 that 9th column which is the ***Mitoses*** characteristic, has the most outliers. But, the 1st, 2nd, 3rd, and 6th columns do not have any outlier values. From Figure 6, it can be seen that the values between 0 and 0.25 are the most frequent in most columns. We can see through the Figure 7 that the number of benign tumors in is much more than the number of malignant tumors.

C. Parameters and tumor type relation using heatmap

In this part, it is asked to show the relationship and influence of each parameter on the type of tumor. So, we used the ***Seaborn*** library to create the correlation matrix, then, we plotted it using the heatmap as shown in Figure 9.

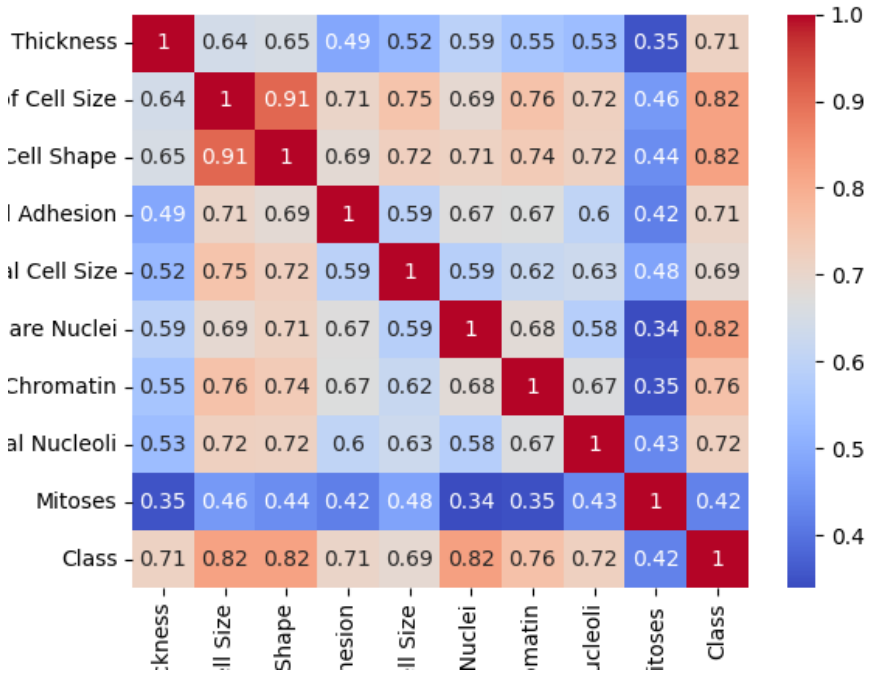


Fig 9. Heatmap representation of the correlation matrix

D. Validation, testing, and training data

i. Splitting data

To split the data into validation, testing, and training sets, we use the following code and the printed results can be seen in Table 3. We allocated 10% of data to validation and 10% to testing, so 80% left of data is automatically allocated to training.

```
train_val, test = train_test_split(df, test_size=0.1, random_state=42)
train, val = train_test_split(train_val, test_size=0.1111, random_state=42)
```

Table 3. Number of samples in training, validation, and testing sets.

Sets	Number of samples
Training	545
Validation	69
Testing	69

ii. Logistic regression and k-fold cross validation

We use logistic regression algorithm to train the following model to predict the output. Where, the validation accuracy equals 0.9130434782608695.

```
lr = LogisticRegression(random_state=42)

lr.fit(X_train, y_train)
```

Now, we use the k-fold cross validation method with a k of 5, to obtain the best performance of model. Where the results are as follows:

- Cross-validation scores: [0.90510949 0.94890511 0.97810219 0.97794118 0.98529412]
- Mean accuracy: 0.959070416487763
- Standard deviation: 0.029735876095083796

From the results, we can see that we had an appropriate value of mean accuracy using 5-fold cross validation.

E. Confusion matrix

Confusion matrix is one of the easiest and most intuitive metrics used for finding the accuracy of a classification model, where the output can be of two or more categories. This is the most popular method used to evaluate logistic regression. The definition of confusion matrix is given in Figure 10.

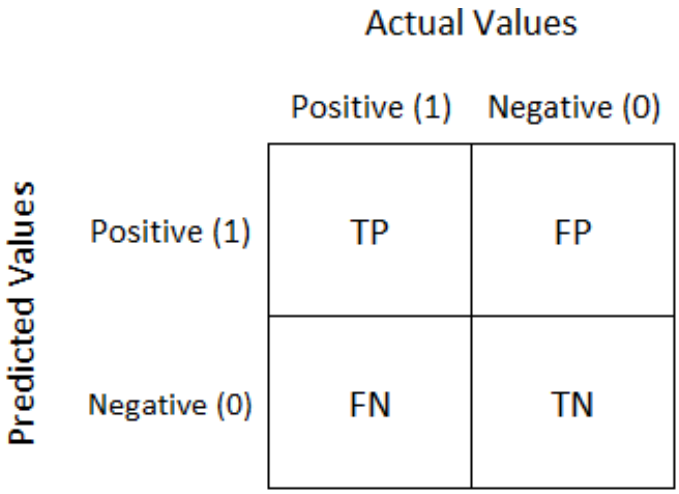


Fig 10. Illustration of confusion matrix concept

Where, TP stands for true positive, FP stands for false positive, FN stands for false negative, and TN represents the true negative predictions. So, the obtained confusion matrix for our model is given in Figure 11. Where TP = 39, FP = 0, FN = 4, and TN = 26 for our model.

39	0
4	26

Fig 11. Confusion matrix of our model

We can calculate the accuracy, error rate, precision, sensitivity, and specificity factors to evaluate the performance of our model. The equations for calculating these factors are as follows. We use a Python code to calculate these factors and the results are given in Table 4.

$$\text{accuracy}=\frac{TP+TN}{total=TP+TN+FP+FN}$$

$$\text{error rate}=\frac{FP+FN}{total=TP+TN+FP+FN}$$

$$\text{precision}=\frac{TP}{TP+FP}$$

$$\text{sensitivity or recall}=\frac{TP}{TP+FN}$$

$$\text{specificity}=\frac{TN}{TN+FP}$$

Table 4. Factors calculated for confusion matrix.

Factor	Number of samples	Percentage
Accuracy	0.9420	94.20%
Error rate	0.0579	5.79%
Precision	1.0000	100%
Sensitivity	0.9069	90.69%
Specificity	1.0000	100%

The accuracy of the model is 94.20%. This means that the model was able to correctly predict the class of 94.20% of the instances. Whether this accuracy is appropriate or not depends on the specific problem and the requirements of the application. In some cases, an accuracy of 94.20% may be considered very good, while in other cases it may not be sufficient. Therefore, it is important to understand the context and the purpose of the model before evaluating its

accuracy. In our case, as we are going to predict the breast cancer and it is really an important matter for saving the patients lives, we should try to increase the accuracy as much as we can.

My suggestions for increasing the accuracy are to collect more data of patients with breast cancer instead of 684 patients, because collecting more data can help the model to learn more patterns and generalize better to new instances. Another way is to have an appropriate feature engineering because good feature engineering can help the model focus on the most important information and ignore the noise. We can use the ensemble methods to combine the predictions of multiple models to improve the accuracy. This can be done by training multiple models with different hyperparameters, using different algorithms, or using different subsets of the data. And finally, choosing the right model for the problem can have a big impact on the accuracy. So, we can improve our model to get better results.

F. Analysis of false predictions

Fortunately, we had not any false positive predictions, but false negatives occurred because the model predicted a benign tumor when the true result was a malignant tumor. From the results, I guess that false negatives happened because as I mentioned before, we had insufficient data or features. If the dataset is too small or lacks important features that distinguish between the two classes, the model may have difficulty accurately predicting the positive class. So, if we increase the samples of patients with breast cancer, we can have more accurate predictions.

My other guess is because of the regression model. Some models, such as logistic regression that we used in our model or decision trees, have a threshold that determines the predicted class. If the threshold is set too high, the model may predict fewer positive cases, resulting in more false negatives.

So, as I mentioned above, the following strategies can reduce the false negatives in a breast cancer prediction model:

- Threshold adjustment of regression model for tumor type prediction
- Changing the model
- An appropriate feature engineering
- Increasing the number of 684 patient samples

In section A, we detected the missing values and outliers and it resulted in many outliers in some dataset columns. In other hands, we saw form Figure 7 that the number of our benign tumors was much more than the malignant tumors. So, we can conclude that our dataset is almost imbalance. If the dataset is imbalanced, one strategy is to oversample the minority class (malignant tumors) or undersample the majority class (benign tumors) to balance the dataset. This can help the model learn to distinguish between the two classes better.

2. Service life prediction of dielectric material

In this section, a dataset with the format of *Xlsx* containing the data of 128 experiments for determining the maximum voltage. In this dataset, each sample contains 3 characteristics in columns 1 to 3. In column 1, we have the service (useful life) data (y). In column 2, we have the operating temperature data (x1). In column 3, we have the maximum allowed voltage data (x2). In the following, we will try to satisfy the demands of the question by using the Python programming language. For coding, a 3.10 Python interpreter has been used. Also, the libraries used in this section, are given in Table 5.

Table 5. Libraries used in section 2.

No.	Library title
1	pandas
2	numpy
3	matplotlib
4	sklearn

A. Regression using 4 kernels

To get started, we read the dataset using pd.read_excel command. First, we create the regression using the linear, 2nd order polynomial, RBF, and sigmoid kernels which are shown in Figures 12 to 16. The mean absolute error (MAE) and R2-score are calculated using k-fold cross validation with a k of 4 and the results are given in Table 6.

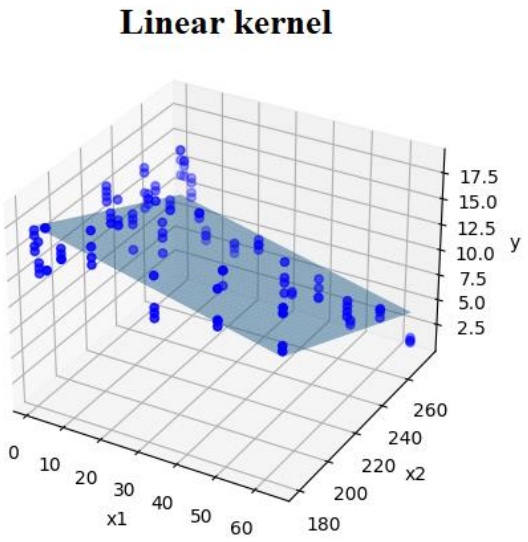


Fig 12. Regression with linear kernel

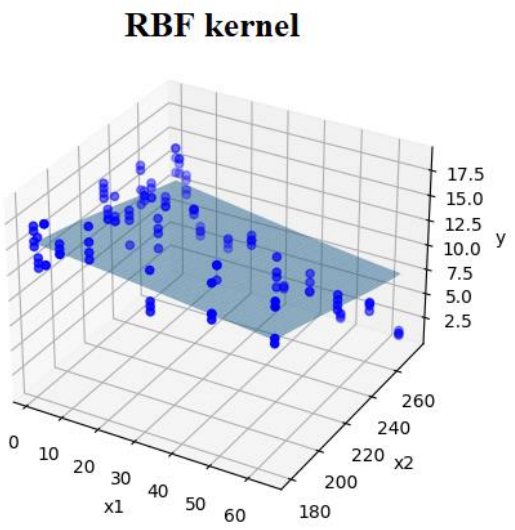


Fig 13. Regression with RBF kernel

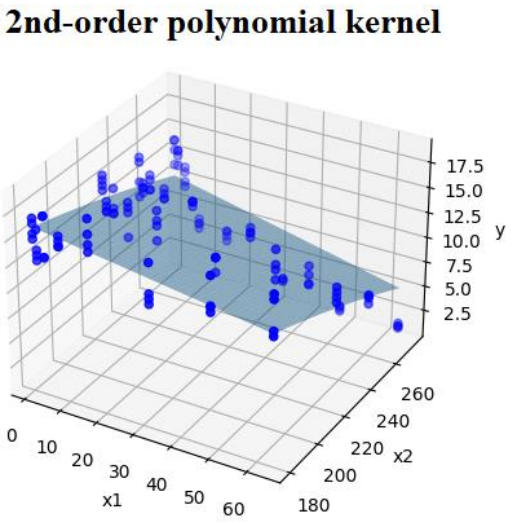


Fig 14. Regression with 2nd order polynomial

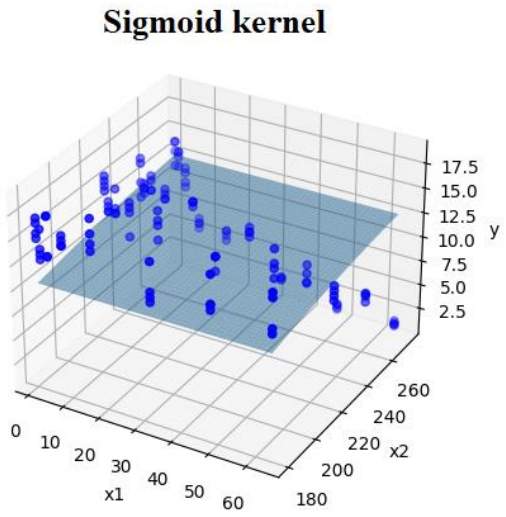


Fig 15. Regression with sigmoid kernel

Table 6. R2-score and MAE for 4 different kernels using k-fold cross validation

Kernel	R2-score	Mean absolute error (MAE)
Linear	0.6492	1.8374
RBF	0.5370	1.9805
2nd order poly	0.7693	1.5894
Sigmoid	-0.3053	3.4531

Where we can see through the Table 6 that 2nd order poly has the highest R2-score with a value of 76.93% and a MAE of 1.5894 which is the lowest among all the kernels.

B. L2-Regularization

Now, we repeat the previous question using R2-Regularization method once for *alpha=1*, and once for *alpha=2*. The results are shown in Table 7.

Table 7. R2-score and MAE for 4 different kernels using R2-regularization

Kernel	Alpha parameter	R2-score	Mean absolute error (MAE)
Linear	alpha=1	-2.3835	4.3158
	alpha=2	-2.3835	4.3158
RBF	alpha=1	-22.8704	10.9321
	alpha=2	-23.2006	10.9745
2nd order poly	alpha=1	-52.5048	16.7044
	alpha=2	-57.5092	17.3518
Sigmoid	alpha=1	-1.2864	3.4858
	alpha=2	-1.3559	3.5144

If we compare the results of Tables 6 and 7, we will see that using k-fold cross validation method will result in higher R2-scores. Also, both the R2-score and MAE values are increased with alpha=2 in compare with alpha=1.

C. Change regularization parameters and use gridSeachCv

Now, we change the parameters between {0.2, 0.8, 1, 5, 10, 20, 50, 300} for linear, 2nd, 3rd, 4th order polynomial, and RBF kernels. The results are shown in Tables 8 to 12.

Table 8. R2-score for linear kernel with different regularization parameters

Regularization parameter	R2-score
0.2	0.6737
0.8	0.6738
1	0.6738
5	0.6738
10	0.6738
20	0.6738
50	0.6737
300	0.6738

Using the gridsearchCv command, the best regularization parameter for linear kernel is 0.2 and the best R2-score is 0.0565.

Table 9. R2-score for 2nd order poly kernel with different regularization parameters

Regularization parameter	R2-score
0.2	0.5903
0.8	0.7043
1	0.7098
5	0.7382
10	0.7453
20	0.7572
50	0.7655
300	0.7994

Using the gridsearchCv command, the best regularization parameter for 2nd order poly kernel is 10 and the best R2-score is 0.2497.

Table 10. R2-score for 3rd order poly kernel with different regularization parameters

Regularization parameter	R2-score
0.2	0.7536
0.8	0.7862
1	0.7897
5	0.8119
10	0.8266
20	0.8327
50	0.8383
300	0.8287

Using the gridsearchCv command, the best regularization parameter for 3rd order poly kernel is 300 and the best R2-score is 0.3098.

Table 11. R2-score for 4th order poly kernel with different regularization parameters

Regularization parameter	R2-score
0.2	0.8215
0.8	0.8388
1	0.8430
5	0.8590
10	0.8570
20	0.8589
50	0.8572
300	0.8581

Using the gridsearchCv command, the best regularization parameter for 4th order poly kernel is 300 and the best R2-score is 0.4886.

Table 12. R2-score for RBF kernel with different regularization parameters

Regularization parameter	R2-score
0.2	0.2732
0.8	0.5569
1	0.5931
5	0.7507
10	0.7993
20	0.8360
50	0.8538
300	0.8722

Using the gridsearchCv command, the best regularization parameter for RBF kernel is 300 and the best R2-score is 0.3583.

After obtaining the best regularization parameters and R2-scores for each kernel, again, we use the gridsearchCv to find the best kernel and regularization parameter between all of them. After running the code, we will see that the best kernel is polynomial with a regularization parameter of 300.

D. Fit the function to data in MATLAB

In this section, we will use the MATLAB to read the dataset again using xlsread command. After that, we will define the given function in MATLAB. Now, we can fit the following function to data points using lsqcurvefit command. The visualization of regression with fitted function is shown in Figure 16. We assumed the initial *beta* as [1 1 1].

$$\log(y) = b_1 - b_2.x_1.\exp(-b_3.x_2)$$

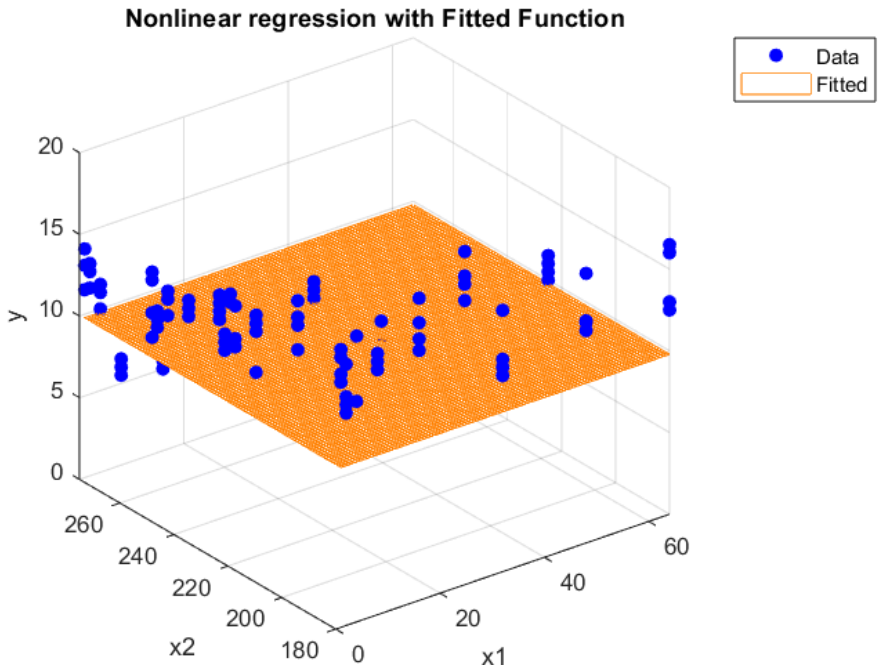


Fig 16. Nonlinear regression with fitted function

The parameters calculated through the MATLAB code are given below:

- b1 = 2.2806
- b2 = 1
- b3 = 1
- R2-score = 0
- RMSE = 0.652

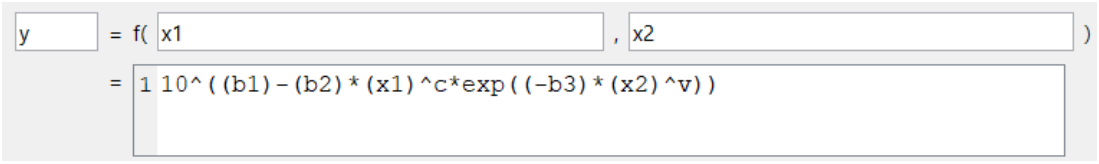
R-squared is a measure of how closely the data in a regression line fit the data in the sample. The closer the r-squared value is to 1, the better the fit. An r-squared value of 0 indicates that the regression surface does not fit the data at all, while an r-squared value of 1 indicates a perfect fit. So, in the next section we will try to increase the R2-score to make the regression better fit the data.

E. Using Cftool in MATLAB

In this section, we will try to increase the R2-score using Cftool in MATLAB. Cftool is a curve fitting tool where we can change the parameters to obtain the best and most optimal solution. So, for this purpose, first of all, we will import the xlsx data to the MATLAB using the following commands. Then, we can just type cftool and run the code to open the Cftool app.

```
load('X1.mat');
load('X2.mat');
load('Y.mat');
```

Finally, in Cftool environment, using the file menu, load the CFTOOL.sfit file which is available through the folder of Question 2. We will use the custom equation option and write the function in the following form:



Where, ***c*** and ***v*** are the different degrees for x1 and x2. By changing the degrees with different values, we can find the best R2-score. You can see the software environment through the Figure 17. Where the new regression with fitted function is showed in Figure 18 with more details.

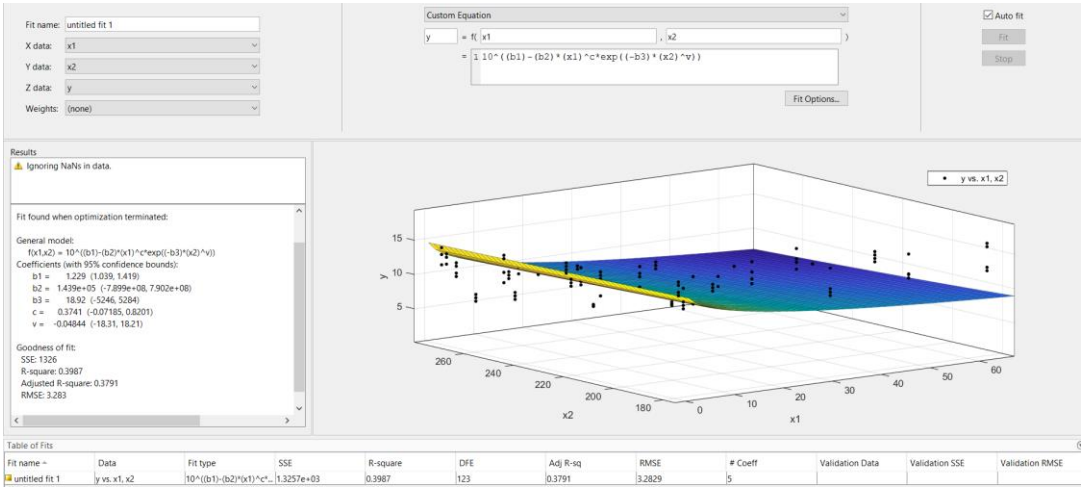


Fig 17. Cftool fitting curve app

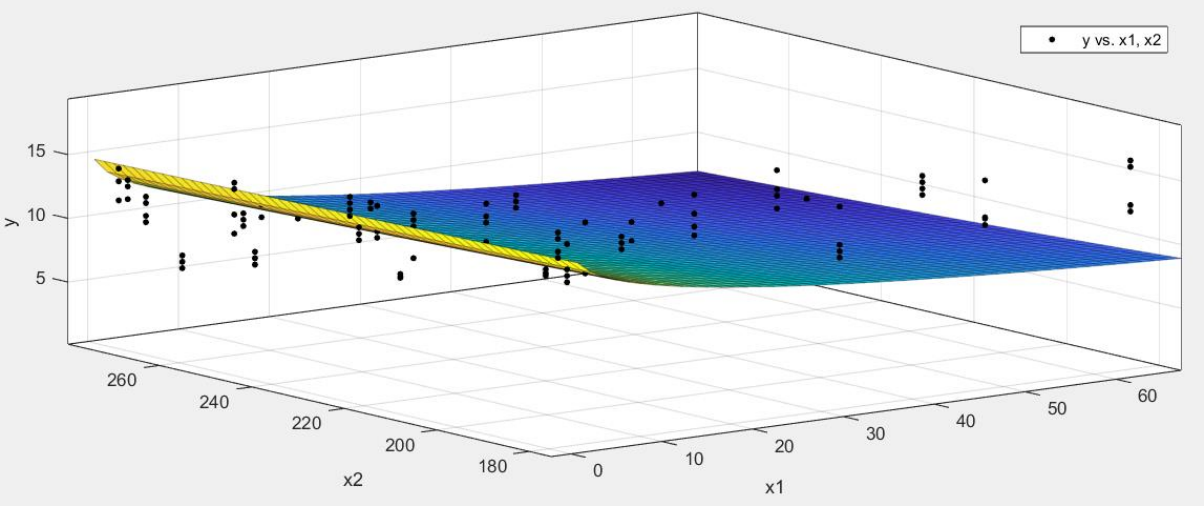


Fig 18. Regression with new function fitting

Where the b1, b2, b3, SSE, R2-score, RMSE, and the degrees of x1 and x2 are given below.

- b1 = 1.229
- b2 = 1.439e5
- b3 = 18.92
- SSE = 1326
- R2-score = 0.3987
- RMSE = 3.283
- C (degree of x1) = 0.3741
- V (degree of x2) = -0.04844

We can conclude that the R2-score has been increased from 0 to 0.3987 and it means that surface is fitting the data better and more efficient.

Thanks for your Time