EGR 598: Machine Learning and Artificial Intelligence (Fall 2022)

Instructor: Shenghan Guo

Assignment 1

This assignment is worth 200 points. It will be due on Thursday, 9/29/2022, and submitted through Canvas. Code should be written in Python. Other programming language can be used upon instructor's approval. A written report should be submitted as a separate document along with the code through Canvas.

Data: A dataset, "Raison_Dataset.csv", is provided to you. Data description can be found in "Raison_Dataset.txt". Please read through the document and then use the data to do the following tasks.

Note: You may consider each column a variable. The *input attributes* include: "Area", "MajorAxisLength", "MinorAxisLength", "Electricity", "ConvexArea", "Extent", "Perimeter". The samples are independent and identically distributed (iid).

Part 1 (70 pts)

1. What is the number of *classes* in this dataset? (2 pts)

Two, "Besni" (C_1) and "Kecimen" (C_2).

2. Calculate the *log odds* for the data. Write the *discriminant function* in terms of the log odds. (6 pts)

There are two classes and 900 data points, with 450 for "Besni" and 450 for "Kecimen". First, use the data to estimate the probability that a sample point comes from each class:

$$\hat{P}(C_1|\boldsymbol{x}) = \frac{450}{900} = 0.5$$

$$\hat{P}(C_2|\boldsymbol{x}) = \frac{450}{900} = 0.5$$

The log odds is therefore the ratio of these probabilities:

$$\log\frac{\hat{P}(C_1|\boldsymbol{x})}{\hat{P}(C_2|\boldsymbol{x})} = \log\frac{0.5}{0.5} = 0$$

The discriminant function in term of the log odds is a function based on $P(C_1|\boldsymbol{x})$ and $P(C_2|\boldsymbol{x})$ (not their estimate with the data), which is

$$g(\boldsymbol{x}) = \log\frac{P(C_1|\boldsymbol{x})}{P(C_2|\boldsymbol{x})} \ and\ choose \begin{cases} C_1, if\ g(x) > 0 \\ C_2, otherwise \end{cases}$$

3. Assume that the input attributes are *multivariate normal*. Further assume that the input attributes in each class follow a different multivariate distribution. Calculate the mean vector and covariance matrix for the input attributes in each class. (Hint: consider your answer in 1. You should obtain this many sets of mean vector and covariance matrix.) (8 pts)

You will obtain $(\boldsymbol{m}_1, \boldsymbol{S}_1)$ for class 1 and $(\boldsymbol{m}_1, \boldsymbol{S}_1)$ for class 2. See Python code for solutions. Notice that $(\boldsymbol{m}_1, \boldsymbol{S}_1)$ are the sample-based estimate for $(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$, and $(\boldsymbol{m}_2, \boldsymbol{S}_2)$ are the sample-based estimate for $(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$. We would never know the true parameters when working with real data. We would use sample estimate to proceed with the calculation of discriminant functions, etc.

4. Given your answer in 3, generate 10 samples from each of the multivariate distributions. (Hints: the number of samples generated should be 10 times number of classes.) (10 pts)

Take the mean vector and covariance matrix for each class from 3, and put them into "numpy.random.multivariate_normal" function for multivariate random number generation. Each sample will be a vector of 7 elements, and each class is associated with 10 samples. See Python code for solutions.

5. Given the assumption that input attributes are multivariate normal, visualize the joint distribution of "MajorAxisLength" and "MinorAxisLength" for each class. Based on the "multivariate normal" assumption, do you think that "MajorAxisLength" and "MinorAxisLength" are both *univariate normal*, and why? (Hint: use your results from 3 and visualize the parametric form of distribution. Create grids for $[0, 800] \times [0, 800]$ for 3D plots.) (10 pts)

See Figure 1 and Python code for solutions.

Assuming that the input attributes are multivariate normal, then "MajorAxisLength" and "MinorAxisLength" both follow a normal distribution, i.e., univariate normal, respectively. This is because the input attributes are jointly normal, so they should be individually normal as well (check textbook). However, notice that a bunch of univariate normal variables do not necessarily form a multivariate normal distribution.

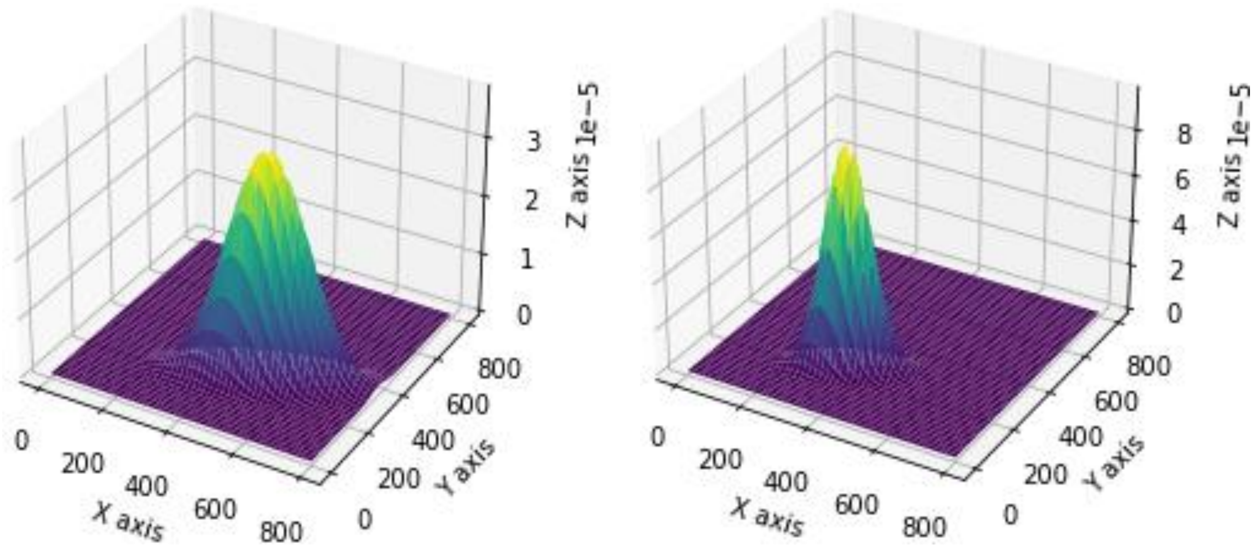

Figure 1. Part 1-5: class 1 (left) and class 2 (right).

6. Given your answers in 3, write the functional form of the *likelihood ratio*. You may define notations for the mean and covariance of each class. (6 pts)

There are two classes, so the likelihood ratio is $p(\boldsymbol{x}|C_1)/p(\boldsymbol{x}|C_2)$, where $p(\cdot)$ is the density of multivariate normal distribution. $p(\boldsymbol{x}|C_1) \sim MVN(\boldsymbol{m}_1, \boldsymbol{S}_1), p(\boldsymbol{x}|C_2) \sim MVN(\boldsymbol{m}_2, \boldsymbol{S}_2)$.

$$p(\boldsymbol{x}|C_1) = (2\pi)^{-\frac{7}{2}}|\boldsymbol{S}_1|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{m}_1)^T\boldsymbol{S}_1^{-1}(\boldsymbol{x}-\boldsymbol{m}_1)\right)$$

$$p(\boldsymbol{x}|C_2) = (2\pi)^{-\frac{7}{2}}|\boldsymbol{S}_2|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{m}_2)^T\boldsymbol{S}_2^{-1}(\boldsymbol{x}-\boldsymbol{m}_2)\right)$$

See Eq. (5.9) and (5.16) in textbook. So, the likelihood ratio is:

$$\frac{p(\boldsymbol{x}|C_1)}{p(\boldsymbol{x}|C_2)} = (|\boldsymbol{S}_1|/|\boldsymbol{S}_2|)^{-\frac{1}{2}}\exp\left(-\frac{1}{2}[(\boldsymbol{x}-\boldsymbol{m}_1)^T\boldsymbol{S}_1^{-1}(\boldsymbol{x}-\boldsymbol{m}_1)-(\boldsymbol{x}-\boldsymbol{m}_2)^T\boldsymbol{S}_2^{-1}(\boldsymbol{x}-\boldsymbol{m}_2)]\right)$$

7. Given your answers in 3 and 6, write the *discriminant function* for each class. Then, calculate the discriminant functions it for each sample point and label each of them with the class name. (Hint: see Eq. (4.20) in textbook. The "label" here is based on your calculated discriminant. You may store the labels in an Excel or .csv file.) (10 pts)

$$g_1(\boldsymbol{x}) = \log p(\boldsymbol{x}|C_1) + \log \hat{P}(C_1) = \log p(\boldsymbol{x}|C_1) + \log 0.5$$

$$g_2(\boldsymbol{x}) = \log p(\boldsymbol{x}|C_2) + \log \hat{P}(C_2) = \log p(\boldsymbol{x}|C_2) + \log 0.5$$

Label a data point as C_1 if $g_1(\boldsymbol{x}) > g_2(\boldsymbol{x})$ and C_2 otherwise. See Python code for labeling results.

8. Given your answers in 3 and 6, if pooling the covariance of all classes, write the *discriminant function* for each class. Then, calculate the discriminant functions it for each sample point and label each of them with the class name. (Hint: see Eq. (5.21) and (5.22) in textbook. The "label" here is based on your calculated discriminant. You may store the labels in an Excel or .csv file.) (10 pts)

$$\boldsymbol{S} = \sum_{i=1}^{2} \hat{P}(C_i)\boldsymbol{S}_i = 0.5 \times (\boldsymbol{S}_1 + \boldsymbol{S}_2)$$

$$g_1(\boldsymbol{x}) = -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{m}_1)^T\boldsymbol{S}^{-1}(\boldsymbol{x}-\boldsymbol{m}_1) + \log 0.5$$

$$g_2(\boldsymbol{x}) = -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{m}_2)^T\boldsymbol{S}^{-1}(\boldsymbol{x}-\boldsymbol{m}_2) + \log 0.5$$

Label a data point as C_1 if $g_1(\boldsymbol{x}) > g_2(\boldsymbol{x})$ and C_2 otherwise. See Python code for labeling results.

9. Use a *confusion matrix* to show the classification results with the *discriminant function* in 7 and 8, respectively. Calculate the classification accuracy for both and compare the results. Briefly describe your findings. (Hint: you will obtain 2 confusion matrices, one for the result in 7 and the other for 8.) (8 pts)

See Python code for result. Based on the classification accuracy, discrimination based on the pooled covariance performed slightly better, with an accuracy of 86%. Discrimination with different covariance had an accuracy of 84.2%.

Part 2 (60 pts)

Do 4-fold cross validation for the dataset and perform classification analysis: (1) randomly shuffle the samples, (2) partition the data into 4 folds, (3) choose 3 out of the 4 folds as *training data* and the rest 1 as *testing data* (you can do this for 4 times by choosing 3 different folds each time).

For each of the 4 replicates, do the following in your Python program:

1. Assume that the input attributes are *multivariate normal*. Calculate the mean vector and covariance matrix for the input attributes in each class using the training data. (10 pts)

In each replicate of 4-fold cross-valuation, you will choose 1 fold as the testing data and the rest 3 folds as the training data. It is recommended to do this in a "for" or "while" loop in Python for 2~5. See Python code for solutions.

2. Given your answers in 1, calculate the *discriminant function* for the testing data. Then label each testing sample with the class name. Finally, create a *confusion matrix* to show the classification result for testing data. (Hint: You may store the labels in an Excel or .csv file.) (15 pts)

3. Given your answers in 1, if pooling the covariance of all classes, calculate the *discriminant function* for the testing data. Then label each testing sample with the class name. Finally, create a *confusion matrix* to show the classification result for testing data. (Hint: You may store the labels in an Excel or .csv file.) (15 pts)

4. For the *discriminant functions* in 2 and 3, respectively, calculate the average *false positive rate*, *false negative rate*, *true positive rate*, and *true negative rate* for the classification results throughout the 4 replicates that you have completed. (Hint: you will get four rates for each classification method.) (15 pts)

See Python code for solutions to 2~4.

5. Briefly describe the performance of each discrimination method and identify the best one for this dataset based on the average performance across 4-fold cross validation. (5 pts)

If letting "Besni" be the positive and "Kecimen" be the negative, then the discrimination without pooled covariance has a higher average TP rate and a lower FN rate, which indicate better accuracy in predicting "Besni". Discrimination with pooled covariance has lower TP rate but a higher TN rate, indicating better accuracy in predicting "Kecimen".

Due to the randomness in shuffling the data points for 4-fold cross validation, each student's result for part 2 may be slightly different. May give credit as long as the comment is reasonable for one's specific results.

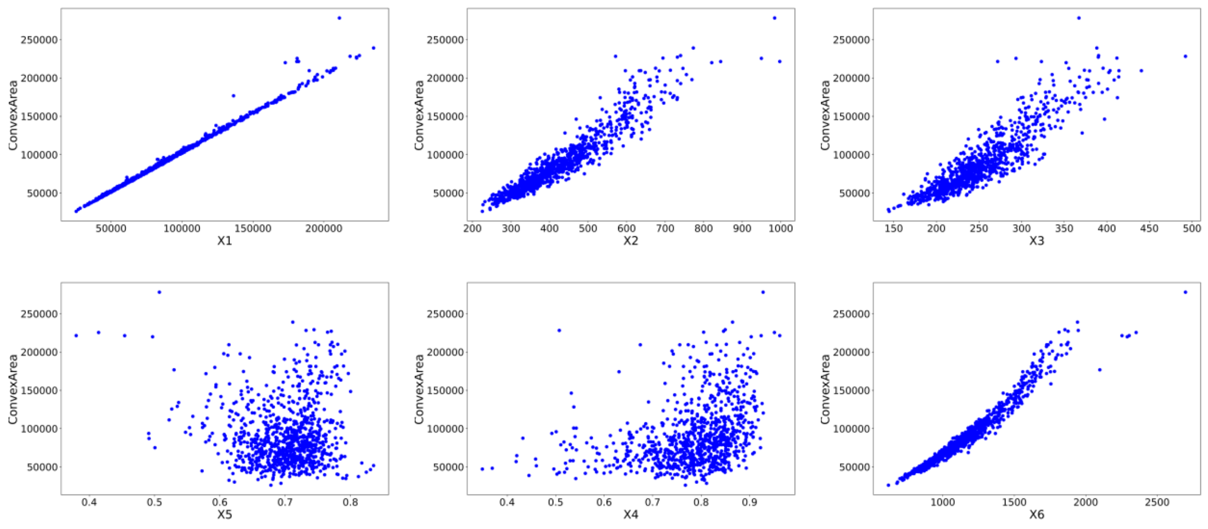

Figure 2. Visualization of "ConvecArea" against independent variables.

Part 3 (70 pts)

For this part, take “Area”, “MajorAxisLength”, “MinorAxisLength”, “Electricity”, “Extent”, “Perimeter” as independent variables, and “ConvexArea” as dependent variables.

1. Visualize “ConvexArea” against each independent variable and describe the trend and patterns in your plots. (Hints: you will get 6 plots, each with “Area” as the vertical axis and an independent variable as the horizontal axis.) (8 pts)

See Figure 2 for demonstrations. See Python code for visualization method.

2. Use the first 600 samples in the dataset as the training data and the rest as the testing data. Calculate the correlation matrix for all dependent and independent variables for the training data. Based on the correlation matrix, identify which independent variables have major impact to the dependent variable. Does the impact imply a causal relationship and why? (Hint: Save the correlations in an Excel or .csv file.) (8 pts)

See Python code for covariance calculation. The 1st, 2nd, 3rd, and 6th independent variables are highly correlated with the dependent variables. The correlation coefficients do NOT imply a causal relationship, but simply a linear dependence between the variables.

3. Use Python to fit a linear regression model using the training data. Summarize the model coefficients. Based on the coefficients, which independent variables have more impact to the dependent variable? (10 pts)

See Python code for solution. The 3rd, 4th, and 5th independent variables have coefficients with large magnitude, implying their bigger impact on the dependent variable.

4. Use the model fitted in 3 to make predictions for testing data. Calculate the *mean squared error* for the testing samples with respect to the predictions. Do you think the model has a good prediction performance? (Hint: input the testing samples of independent variables into your fitted model and then evaluate the prediction against the true sample values of dependent variable.) (8 pts)

See Python code for solution. The model does NOT have a good prediction performance, as revealed by the large MSE.

5. Based on result in 3, do you think that the independent variables are mutually linearly independent? What’s the influence on the linear regression model with the appearance of linear dependence among independent variables? (5 pts)

No, they have linear dependency. The term is “Multicollinearity”.

Multicollinearity occurs when two or more independent variables are highly correlated with one another in a regression model. This means that an independent variable can be predicted from another independent variable in a regression model. Multicollinearity can be a problem in a regression model because we would not be able to distinguish between the individual effects of the independent variables on the dependent variable.

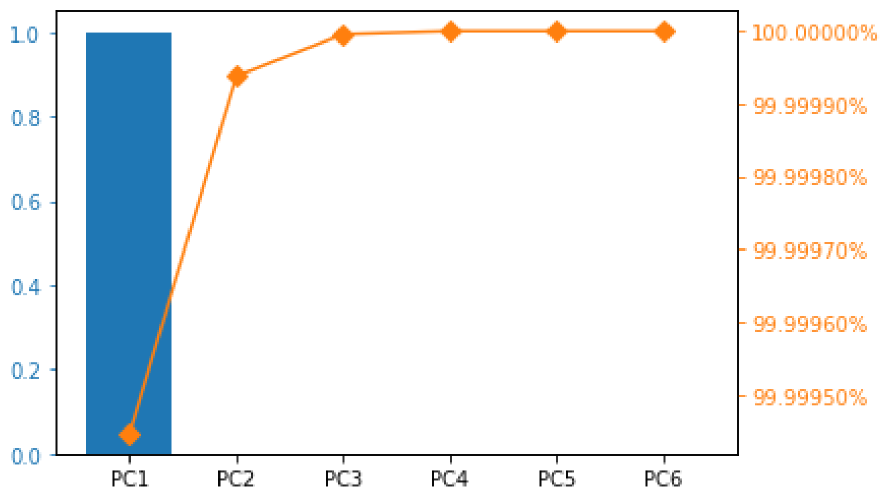

Figure 3. Perato chart for PCA.

6. Perform *Principal Component Analysis* on the training data matrix of independent variables. Show the variance explained by each principal component. (6 pts)

See Python code.

7. Visualize a *Pareto chart* for the variance explained by each principal component. (10 pts)

See Figure 3 and Python code.

8. Take the first 3 principal components from 7 and fit a multivariate regression model. Do prediction with the model for testing samples and calculate the mean squared error. (Hint: You need to transform the testing data to principal components as well.) (10 pts)

See Python code.

9. Give a practical scenario when you will use *Principal Component Analysis* to reduce the data dimensionality before fitting a regression model; give a practical scenario when you will NOT use *Principal Component Analysis* to reduce the data dimensionality before fitting a regression model. (Hint: you can name any data source and/or situations, which is not necessarily related to the Raison dataset.) (5 pts)

Will: multicollinearity arises in the independent variables, and these independent variables do not have strong, practical meanings. Using PCA to extract PCs from them will reduce the dimensionality and facilitate regression model fitting.

Will Not: independent variables may have some multicollinearity, but they each have practical meanings and the interpretation of regression model is important.