COMP7703 Machine Learning
Assignment 1

Thanat Chokwijitkul 44522328

Question 1.2

Use Wekas Visualize and the histogram/statistical summary information on the Preprocess tab, do some "Exploratory Data Analysis" of the dataset "mystery.csv".

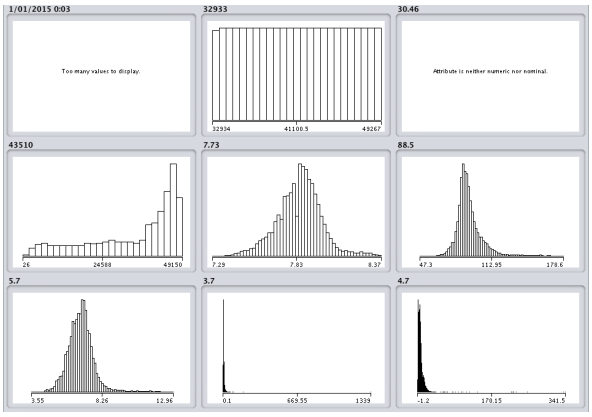


Figure 1: A histogram visualising each feature of the mystery data

The mystery data has 16,315 instances with 9 attributes (1/01/2015 0:03, 32933, 30.46, 43510, 7.73, 88.5, 5.7, 3.7 and 4.7). Among these attributes, 7 attributes are numeric (32933, 43510, 7.73, 88.5, 5.7, 3.7 and 4.7) and the rest are nominal (1/01/2015 0:03) and string (30.46).

1/01/2015 0:03 is of the nominal type. Implied by its name, this is the timestamp attribute. The values of all instances are 100% distinct and unique. 30.46 is of type string, meaning that all of its instances contain information expressed as a sequence of characters.

32933 contains the values in the range of 32934 to 49267, with a mean of 41109.774 and low standard deviation of 4710.266. These values are 100% distinct. However, the visualisation is not accurate due to the limitation of the space to represent such small differences.

The histogram representing data of the 43510 attribute is negatively skewed, with a mean of 34670.803 and standard deviation of 13489.978. The data is in the range of 26 to 49150.

The diagrams visualising data of 3.7 and 4.7 are very positively skewed with noticeable

outliers. These outliers strongly impact the overall results since the standard deviation values of these two sets of data are 11.813 and 8.048 while their mean values are 9.569 and 6.349 for 3.7 and 4.7 respectively.

The mean and standard deviation values of the other three attributes, including 7.73, 88.5 and 5.7, are 7.846, 92.423 and 6.838, and 0.142, 12.206 and 0.89 respectively. These standard deviation values are very low, meaning that most values are tightly clustered around the mean values. It is obvious that these values are normally distributed since approximately 68% of the values of these attributes lie within one standard deviation of the means according to the Empirical Rule.

Nevertheless, 1/01/2015 0:03 and its associated scatterplot imply that this mystery data seems to be a time-series data. The values from other attributes show statistical data of some event at a particular time and a new instance is created every 30 minutes. According to the patterns of the numerical data observed from the data, machine learning can definitely be applied as a tool for statistical modelling and prediction in this case.

Question 1.6

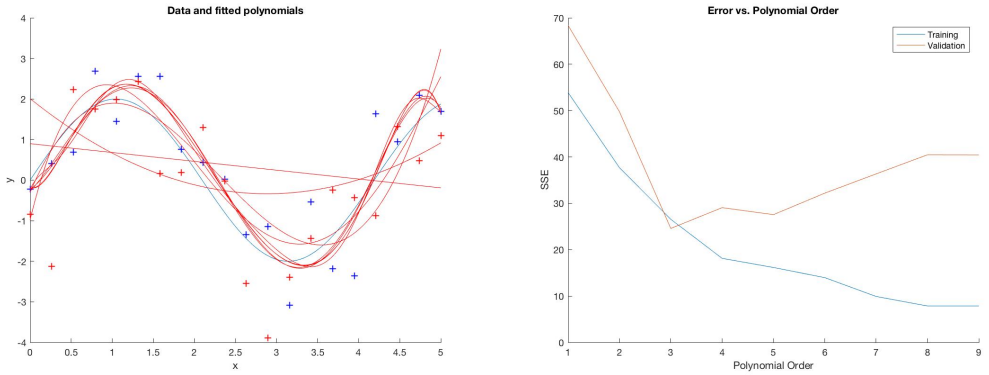
Create a function that has two inputs; 1) a vector (here we will call 'in') and 2) an integer (here we will call 'n'). The function must return a vector as output (here we will call 'out'). The purpose of the function is to reverse the input vector in chunks of size 'n' - the first 'n' entries of 'out' are the last 'n' entries in 'in' (and so forth). If the value of 'n' causes a chunk less than 'n' to be left in 'in', please append the remaining entries to out.

```
function out = reverse(in, n)
% Reverses the input vector in chunks of size 'n'.
    out = [];
    s = size(in);
    for i = 1:floor(s(2)/n)
        out = [out in(end-n+1:end)];
        in = in(1:end-n);
    end
    out = [out in];
end
```

This function works correctly according the example of appropriate inputs and outputs given in the practical sheet.

Question 2.1

Using the curve fitting tool to produce a plot of error as a function of model order up to 9, similar to that shown in Fig.4.7 of the Alpaydin book.



(a) Data and fitted polynomials

(b) Error vs polynomial order

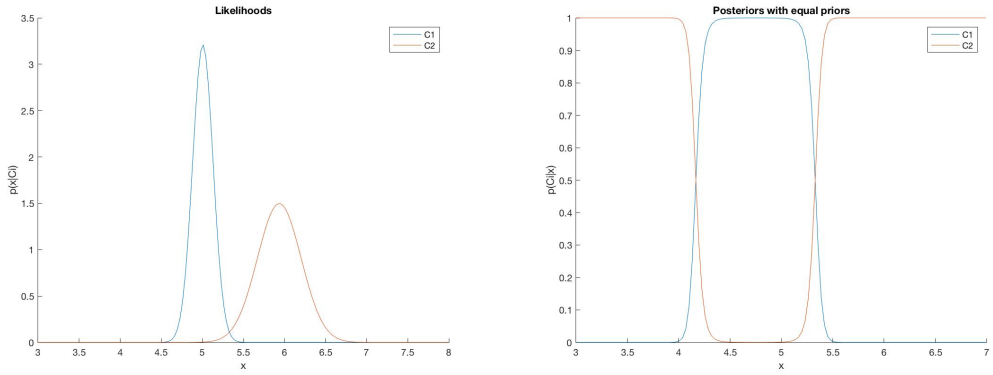
Figure 2: (a) Data and fitted polynomials of order from 1 - 9. (b) Training and validation errors as a function of the polynomial order.

Fig.2(a) illustrates data, including training (blue points) and validation (red points), and fitted polynomials of order from 1 - 9 (fitted on the training set). Fig.2(b) represents training and validation errors (using SSE) as a function of the polynomial order. According to [1], the *elbow* is 3, which indicates the optimal complexity level. The polynomial orders after 3 tends to overfit the data, resulting in inefficient generalisation.

Question 2.4

Using Matlab, write a function or script that implements a simple parametric classifier to produce plots similar to Figs 4.2 and 4.3 of Alpaydin. Use your code to produce plots of the likelihoods and class posteriors for the reduced Iris data.

The reduced Iris data was split into two smaller vectors, including data points that belong to class Iris Setosa and Iris Versicolor respectively. The equations 4.25 and 4.26 from [1] were used to compute the mean and variance estimators for each class. The likelihoods of both classes (Fig.3(a)) were plotted using normal probability density function (the normpdf function in MATLAB). Class posteriors (Fig.3(b)) were computed using the Bayes' rule presented in [1]. The classification results can be obtained based on the assumption that if the posterior of a data point is greater than 0.5, then it is predicted to be of the class Iris Setosa, Iris Versicolor otherwise. This simple probabilistic classification model achieved accuracy of 0.85, recall of 0.80, precision of 0.89 and F1-score of 0.8421 when performing classification on the training set.



(a) Likelihoods (b) Posteriors with equal priors

Figure 3: (a) Likelihoods and (b) posteriors with equal priors

Question 3.1

Apply and evaluate quadratic discriminant analysis on the Pima Diabetes dataset.

The Pima Diabetes dataset was split into two smaller datasets, including training (500 data points) and test (268 data points) sets. The classification results can be obtained based on the assumption that if the posterior of a data point is greater than 0.5, then it is predicted to be of the class positive (diabetic), negative (not diabetic) otherwise.

The training classification performance: The model achieved accuracy of 0.754 with **0.246 error rate**, recall of 0.8679, precision of 0.7731 and F1-score of 0.8178.

The test classification performance: The model achieved accuracy of 0.7799 with **0.2201 error rate**, recall of 0.8681, precision of 0.8187 and F1-score of 0.8427, which is slightly better than when it performs classification on the training data.

The model parameters

$$\mu_0 = \begin{bmatrix} 3.2516 & 110.5063 & 68.1981 & 19.9591 & 68.1321 & 30.0654 & 0.4397 & 31.2830 \end{bmatrix}$$

$$\mu_1 = \begin{bmatrix} 4.7802 & 140.4890 & 69.7253 & 21.7143 & 102.4286 & 35.3231 & 0.5672 & 36.2692 \end{bmatrix}$$

$$\Sigma_0 = 1.0e{+}04 \times \begin{bmatrix} 0.0009 & 0.0011 & 0.0006 & -0.0004 & -0.0033 & 0.0001 & -0.0000 & 0.0019 \\ 0.0011 & 0.0776 & 0.0072 & -0.0002 & 0.1061 & 0.0038 & 0.0001 & 0.0099 \\ 0.0006 & 0.0072 & 0.0312 & 0.0050 & 0.0142 & 0.0057 & 0.0000 & 0.0038 \\ -0.0004 & -0.0002 & 0.0050 & 0.0218 & 0.0613 & 0.0055 & 0.0000 & -0.0025 \\ -0.0033 & 0.1061 & 0.0142 & 0.0613 & 1.0863 & 0.0242 & 0.0009 & -0.0166 \\ 0.0001 & 0.0038 & 0.0057 & 0.0055 & 0.0242 & 0.0064 & 0.0000 & 0.0007 \\ -0.0000 & 0.0001 & 0.0000 & 0.0000 & 0.0009 & 0.0000 & 0.0000 & 0.0000 \\ 0.0019 & 0.0099 & 0.0038 & -0.0025 & -0.0166 & 0.0007 & 0.0000 & 0.0138 \end{bmatrix}$$

$$\Sigma_1 = 1.0e{+}04\times \begin{bmatrix} 0.0014 & 0.0008 & 0.0011 & -0.0005 & -0.0029 & -0.0004 & -0.0000 & 0.0018 \\ 0.0008 & 0.0968 & 0.0040 & 0.0003 & 0.1135 & 0.0006 & 0.0001 & 0.0060 \\ 0.0011 & 0.0040 & 0.0494 & 0.0078 & 0.0270 & 0.0007 & -0.0000 & 0.0060 \\ -0.0005 & 0.0003 & 0.0078 & 0.0297 & 0.1239 & 0.0035 & 0.0002 & -0.0031 \\ -0.0029 & 0.1135 & 0.0270 & 0.1239 & 1.9763 & 0.0031 & 0.0005 & 0.0162 \\ -0.0004 & 0.0006 & 0.0007 & 0.0035 & 0.0031 & 0.0056 & 0.0000 & -0.0017 \\ -0.0000 & 0.0001 & -0.0000 & 0.0002 & 0.0005 & 0.0000 & 0.0000 & -0.0000 \\ 0.0018 & 0.0060 & 0.0060 & -0.0031 & 0.0162 & -0.0017 & -0.0000 & 0.0115 \end{bmatrix}$$

Question 3.2

Apply and evaluate linear discriminant analysis on the diabetes dataset. Use a diagonal common/shared covariance matrix with equal variances for all dimensions, for each class.

Solution 1

This solution uses a diagonal common/shared covariance matrix with equal variances for all dimensions. However, replacing all the diagonal elements by 1 will cause an error as an input covariance matrix for the multivariate density estimation function (mvnpdf) needs to be a square, symmetric, positive definite matrix. Therefore, a potential approach is to reconstruct a new covariance matrix using the data from the existing matrix, namely estimating a specific shared value for all variances (diagonal elements) so that all the common/share covariances remain unchanged and the new covariance matrix still satisfies all the aforementioned properties. The following code illustrates how the new covariance matrix can be constructed in MATLAB:

```
pima = load('pima.txt');
training = pima(1:500,:);
sigma = cov(training(:,1:8));
s = zeros(size(sigma,1),size(sigma,2));

for i = 1:size(sigma,1)
    temp = sigma;
    temp(eye(size(temp))≠0) = sigma(i,i);
    s = s + temp;
end

s = s/size(sigma,1);
sigma = s;
```

The training classification performance: The model achieved accuracy of 0.658 with **0.342 error rate**, recall of 0.8176, precision of 0.6971 and F1-score of 0.7526.

The test classification performance: The model achieved accuracy of 0.6418 with **0.3582 error rate**, recall of 0.7637, precision of 0.7240 and F1-score of 0.7433.

The model parameters

$$\mu_0 = \begin{bmatrix} 3.2516 & 110.5063 & 68.1981 & 19.9591 & 68.1321 & 30.0654 & 0.4397 & 31.2830 \end{bmatrix}$$

$$\mu_1 = \begin{bmatrix} 4.7802 & 140.4890 & 69.7253 & 21.7143 & 102.4286 & 35.3231 & 0.5672 & 36.2692 \end{bmatrix}$$

$$\Sigma = 1.0e{+}03 \times \begin{bmatrix} 2.0291 & 0.0204 & 0.0080 & -0.0035 & -0.0196 & 0.0012 & -0.0000 & 0.0206 \\ 0.0204 & 2.0291 & 0.0707 & 0.0119 & 1.3245 & 0.0630 & 0.0018 & 0.1191 \\ 0.0080 & 0.0707 & 2.0291 & 0.0611 & 0.2003 & 0.0405 & 0.0000 & 0.0473 \\ -0.0035 & 0.0119 & 0.0611 & 2.0291 & 0.8530 & 0.0500 & 0.0009 & -0.0248 \\ -0.0196 & 1.3245 & 0.2003 & 0.8530 & 2.0291 & 0.2069 & 0.0086 & -0.0070 \\ 0.0012 & 0.0630 & 0.0405 & 0.0500 & 0.2069 & 2.0291 & 0.0004 & 0.0043 \\ -0.0000 & 0.0018 & 0.0000 & 0.0009 & 0.0086 & 0.0004 & 2.0291 & 0.0002 \\ 0.0206 & 0.1191 & 0.0473 & -0.0248 & -0.0070 & 0.0043 & 0.0002 & 2.0291 \end{bmatrix}$$

Solution 2

This solution assumes that the Gaussians for both classes share the same covariance matrix, namely $\Sigma_C = \Sigma$, where Σ_C is the covariance matrix of class C.

The training classification performance: The model achieved accuracy of 0.762 with **0.238 error rate**, recall of 0.9025, precision of 0.7653 and F1-score of 0.8283.

The test classification performance: The model achieved accuracy of 0.7985 with **0.2015 error rate**, recall of 0.9396, precision of 0.7991 and F1-score of 0.8637, which is slightly better than the previous results.

The model parameters

$$\mu_0 = \begin{bmatrix} 3.2516 & 110.5063 & 68.1981 & 19.9591 & 68.1321 & 30.0654 & 0.4397 & 31.2830 \end{bmatrix}$$

$$\mu_1 = \begin{bmatrix} 4.7802 & 140.4890 & 69.7253 & 21.7143 & 102.4286 & 35.3231 & 0.5672 & 36.2692 \end{bmatrix}$$

$$\Sigma = 1.0e{+}04 \times \begin{bmatrix} 0.0011 & 0.0020 & 0.0008 & -0.0004 & -0.0020 & 0.0001 & -0.0000 & 0.0021 \\ 0.0020 & 0.1052 & 0.0071 & 0.0012 & 0.1324 & 0.0063 & 0.0002 & 0.0119 \\ 0.0008 & 0.0071 & 0.0378 & 0.0061 & 0.0200 & 0.0040 & 0.0000 & 0.0047 \\ -0.0004 & 0.0012 & 0.0061 & 0.0247 & 0.0853 & 0.0050 & 0.0001 & -0.0025 \\ -0.0020 & 0.1324 & 0.0200 & 0.0853 & 1.4343 & 0.0207 & 0.0009 & -0.0007 \\ 0.0001 & 0.0063 & 0.0040 & 0.0050 & 0.0207 & 0.0067 & 0.0000 & 0.0004 \\ -0.0000 & 0.0002 & 0.0000 & 0.0001 & 0.0009 & 0.0000 & 0.0000 & 0.0000 \\ 0.0021 & 0.0119 & 0.0047 & -0.0025 & -0.0007 & 0.0004 & 0.0000 & 0.0135 \end{bmatrix}$$

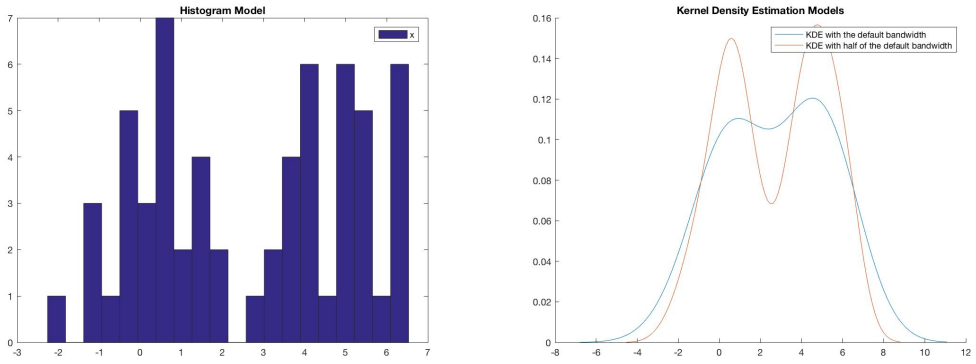
Question 3.5

Calculate the KL divergence (based on 100 equally spaced test points that cover the range of the data in x), between:

- M and H_1
- M and K_1
- M and K_2

where M is the distribution given by the mixture of 2 Gaussians. In addition to the 3 KL divergence values, provide the code used to calculate them. Include and discuss any assumptions or "work-arounds" you used.

Three nonparametric density estimation models were constructed in MATLAB, including one histogram model with 20 bins (H_1) and two kernel density estimators (K_1 and K_2). In this case, K_1 and K_2 are constructed using the default optimal bandwidth value and half of the default bandwidth respectively.



(a) Histogram model (b) Kernel density estimation models

Figure 4: (a) Histogram model and (b) kernel density estimation models

Fig.4 shows the visualisation of the histogram model (H_1) and kernel density estimation models (K_1 and K_2). The histogram model (Fig.4(a)) is simple, yet risky for inaccurate interpretation since it is highly dependent on both end points of bins and width of bins. However, the kernel density estimators (KDE) (Fig.4(b)) present smoother distributions of data. It is crucial to select an appropriate value for bandwidth as a value that is too large or too small may cause underfitting or overfitting. As illustrated in Fig.4(b), the blue curve is a KDE model with a default bandwidth and another is computed using only half of the default bandwidth value, resulting in a less smooth model that may either fit the data better or lead to overfitting.

KL Divergence Values (With Normalisation)

- $KL_1 = 0.4791$

- $KL_2 = 0$
- $KL_3 = 0.0804$

The true distribution $p(x)$ and the model distribution $q(x)$ are normalised to ensure that all the data points into the range [0,1] for the ease of interpretation (ranking the *best fit* models). In this case, K_1 appears to be the best model according to the resulting KL divergence values. These values were computed manually using to the following equation:

$$D_{KL}(p||q) = \sum_i p(x_i) log \frac{p(x_i)}{q(x_i)}$$

The following code is the list of MATLAB statements used to construct H_1, K_1 and K_2 along with computing the resulting KL divergence between M and H_1 (KL_1), M and K_1 (KL_1) and M and K_2 (KL_1).

```
function [kl_1 kl_2 kl_3] = p3q5()
    rng default
    x = [randn(30,1); 5+randn(30,1)];
    h = hist(x,20);
    h_1 = normpdf(h/2,mean(h/2),cov(h/2));
    [f_1,x_1,bw_1] = ksdensity(x);
    [f_2,x_2,bw_2] = ksdensity(x,'Bandwidth',bw_1/2);
    [m_1,xm_1] = ksdensity(x,'NumPoints',20);
    [m_2,xm_2] = ksdensity(x,'NumPoints',100);
    m_1 = m_1 ./repmat(sum(m_1,2),[1 size(m_1,2)]);
    h_1 = h_1 ./repmat(sum(h_1,2),[1 size(h_1,2)]);
    temp =  m_1.*log(m_1./h_1);
    temp(isnan(temp)) = 0;
    kl_1 = sum(temp,2);
    m_2 = m_2 ./repmat(sum(m_2,2),[1 size(m_2,2)]);
    f_1 = f_1 ./repmat(sum(f_1,2),[1 size(f_1,2)]);
    temp =  m_2.*log(m_2./f_1);
    temp(isnan(temp)) = 0;
    kl_2 = sum(temp,2);
    f_2 = f_2 ./repmat(sum(f_2,2),[1 size(f_2,2)]);
    temp =  m_2.*log(m_2./f_2);
    temp(isnan(temp)) = 0;
    kl_3 = sum(temp,2);

    % KL divergence without normalisation
    %kl_1 = sum(m_1.*(log(m_1./h_1)));
    %kl_2 = sum(m_2.*(log(m_2./f_1)));
    %kl_3 = sum(m_2.*(log(m_2./f_2)));
end
```

References

[1] E. Alpaydin, *Introduction to machine learning.* MIT press, 2014.