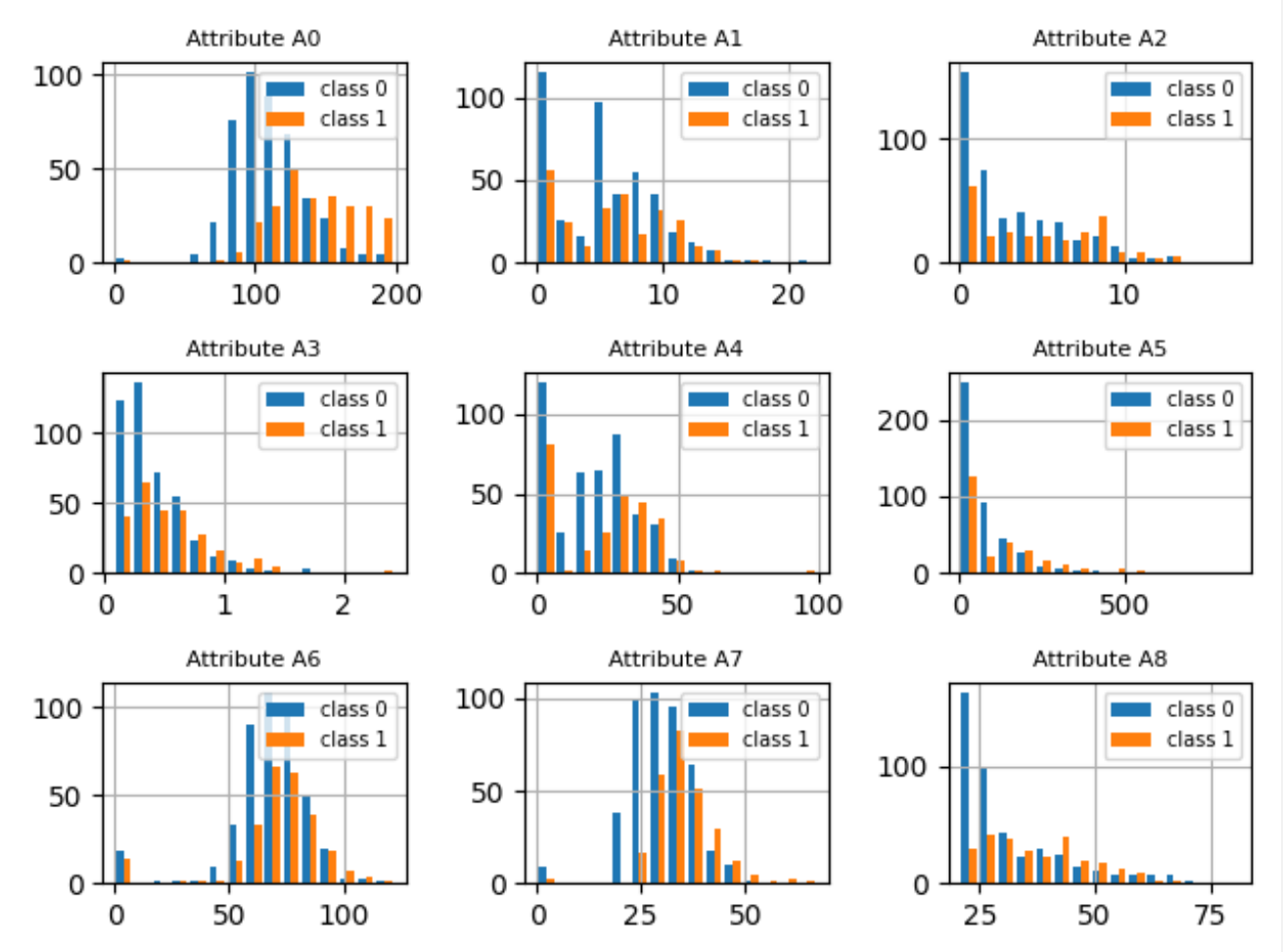
Question 1 : (80 total points) Experiments on a binary-classification data set

1.1 (9 points) We want to see how each feature in `Xtrn` is distributed for each class. Since there are nine attributes, we plot a total of nine figures in a 3-by-3 grid, where the top-left figure shows the histograms for attribute 'A0' and the bottom-right 'A8'. In each figure, you show histograms of instances of class 0 and those of class 1 using `pyplot.hist([Xa, Xb], bins=15)`, where `Xa` corresponds to instances of class 0 and `Xb` to those of class 1, and you set the number of bins to 15. Use grid lines. Based on the results you obtain, discuss and explain your findings.

As shown in Figure 1.1, through the obtained bar graph, we can clear the observed distribution of the respective values of 9 variables.
My findings:
1. The distribution of some variables is very concentrated (i.e. the variance is very small), such as variables A1, A2 and A3, and some variables are scattered (i.e. the variance is large), such as variables A0 and A5;
2. Some variables are concentrated in the middle of their distribution range, while some variables are at the boundary of their distribution range;
3. Finally, it can be seen from the histogram color of Class 0 and class 1 data that the value of Class 0 sample is generally larger than that of class 1 sample, which may be an important basis for the model to classify these two types of samples.



Question 1 continued on next page...

1.2 (9 points) Calculate the correlation coefficient between each attribute of `Xtrn` and the label `Ytrn`, so that you calculate nine correlation coefficients. Answer the following questions.

(a) Report the correlation coefficients in a table.
(b) Discuss if it is a good idea to use the attributes that have large correlations with the label for classification tasks.
(c) Discuss if it is a good idea to ignore the attributes that have small correlations with the label for classification tasks.

(a) The correlation coefficients between each attribute of Xtrn and the label Ytrn is shown as the following table.

Correlation coefficients	A0	A1	A2	A3	A4	A5	A6	A7	A8
Y	0.4912	0.0874	0.2273	0.2074	0.1077	0.1857	0.0762	0.3045	0.2403

(b) It is not a good idea to use only variables with large correlation coefficient to establish a classification model. If the correlation coefficient between a variable and the output value y is large, it indicates that the variable can explain the change of y value to a great extent. Therefore, building a model with this variable is indeed helpful to improve the classification accuracy, but we need to manually determine: how large the correlation coefficient is? This completely depends on human experience and is not scientific; In addition, when dealing with high-dimensional classification problems, the correlation coefficient of most variables may be very small, so it is impossible to choose.

(c) It is not a good idea to ignore the variables with small correlation coefficient when building a classification model. Because the correlation coefficient can only describe the linear correlation between two variables, but can not reflect the nonlinear relationship between them. Therefore, if they are excluded from the model just because their correlation coefficient is small, the model can not learn the nonlinear relationship between variables, resulting in the poor classification effect of the model.

Question 1 continued on next page...

1.3 (4 points) We consider a set of instances of two variables, $\{(u_i, v_i)\}_{i=1}^N$, where N denotes the number of instances. Show (using your own words and mathematical expressions) that the correlation coefficient between the two variables, r_{uv}, is translation invariant and scale invariant, i.e. r_{uv} does not change under linear transformation, $a + bu_i$ and $c + dv_i$ for $i = 1, \ldots, N$, where a, b, c, d are constants and $b > 0, d > 0$.

The correlation coefficient between u and v is

$$r_{uv} = \frac{cov(u, v)}{\sqrt{D(u)D(v)}}.$$

With linear transformation $u'_i = a + bu_i, v'_i = c + dv_i$, it can be seen that

$$cov(u', v') = cov(a + bu, c + dv) = bdcov(u, v),$$

$$D(u') = D(a + bu) = b^2 D(u),$$
$$D(v') = D(c + du) = d^2 D(v).$$

Thus

$$r_{u'v'} = \frac{cov(u', v')}{\sqrt{D(u')D(v')}} = \frac{bdcov(u, v)}{\sqrt{b^2 D(u) d^2 D(v)}} = \frac{cov(u, v)}{\sqrt{D(u)D(v)}} = r_{uv}.$$

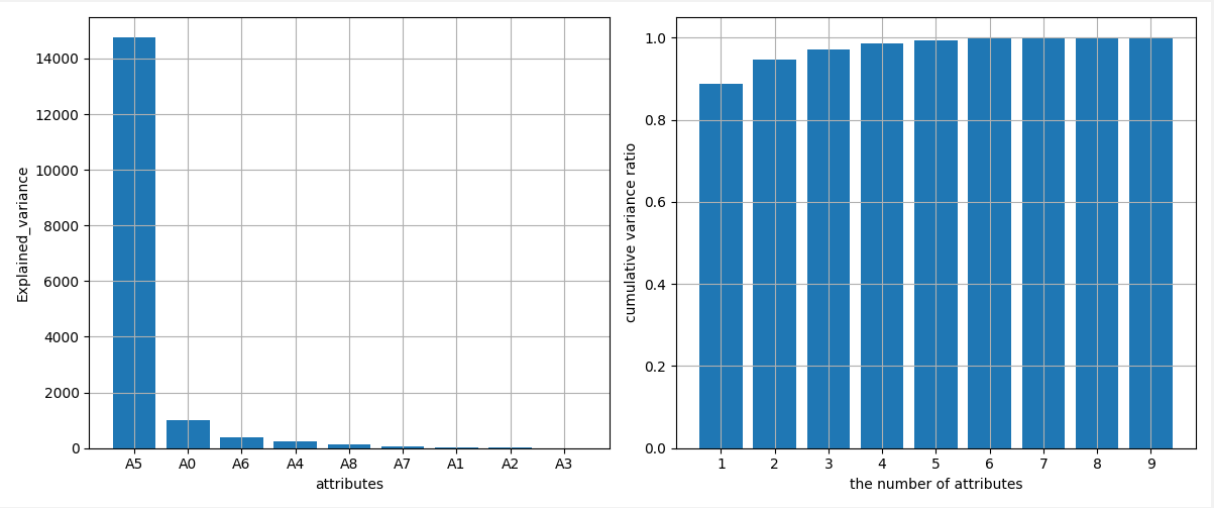
Question 1 continued on next page...

1.4 (5 points) Calculate the unbiased sample variance of each attribute of `Xtrn`, and sort the variances in decreasing order. Answer the following questions.

(a) Report the sum of all the variances.
(b) Plot the following two graphs side-by-side. Use grid lines in each plot.
- A graph of the amount of variance explained by each of the (sorted) attributes, where you indicate attribute numbers on the x-axis.
- A graph of the cumulative variance ratio against the number of attributes, where the range of y-axis should be [0, 1].

(a) The sum of all the variances is 16621.8571.

(b) The resulting figures are shown as follows. Note that the attributes in x-axis is already sorted in decreasing order according to its variance, which is A5, A0, A6, A4, A8, A7, A1, A2, A3.



Question 1 continued on next page...

1.5 (8 points) Apply Principal Component Analysis (PCA) to `Xtrn`, where you should not rescale `Xtrn`. Use Sklearn's PCA with default parameters, i.e. specifying no parameters.

(a) Report the total amount of unbiased sample variance explained by the whole set of principal components.

(b) Plot the following two graphs side-by-side. Use grid lines in each plot.

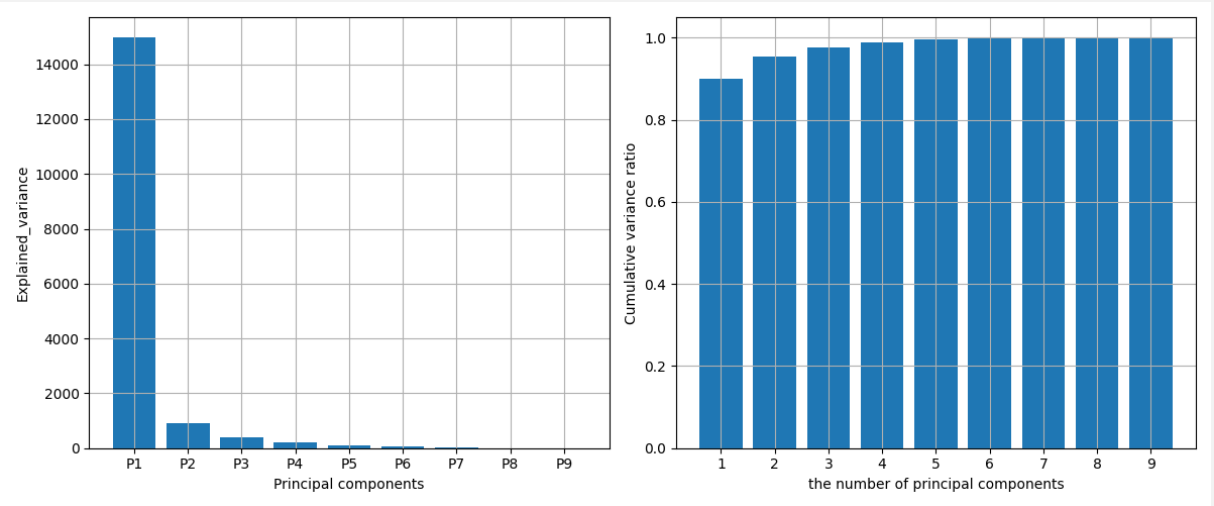
- A graph of the amount of variance explained by each of the principal components.
- A graph of the cumulative variance ratio, where the range of y-axis should be [0, 1].

(c) Mapping all the instances in `Xtrn` on to the 2D space spanned with the first two principal components, and plot a scatter graph of the instances on the space, where instances of class 0 are displayed in blue and those of class 1 in red. Use grid lines. Note that the mapping should be done directly using the eigen vectors obtained in PCA - you should not use Sklearn's functions, e.g. `transform()`.

(d) Calculate the correlation coefficient between each attribute and each of the first and second principal components, report the result in a table.

(a) The total amount of unbiased sample variance explained by the whole set of principal components is 16645.6366.

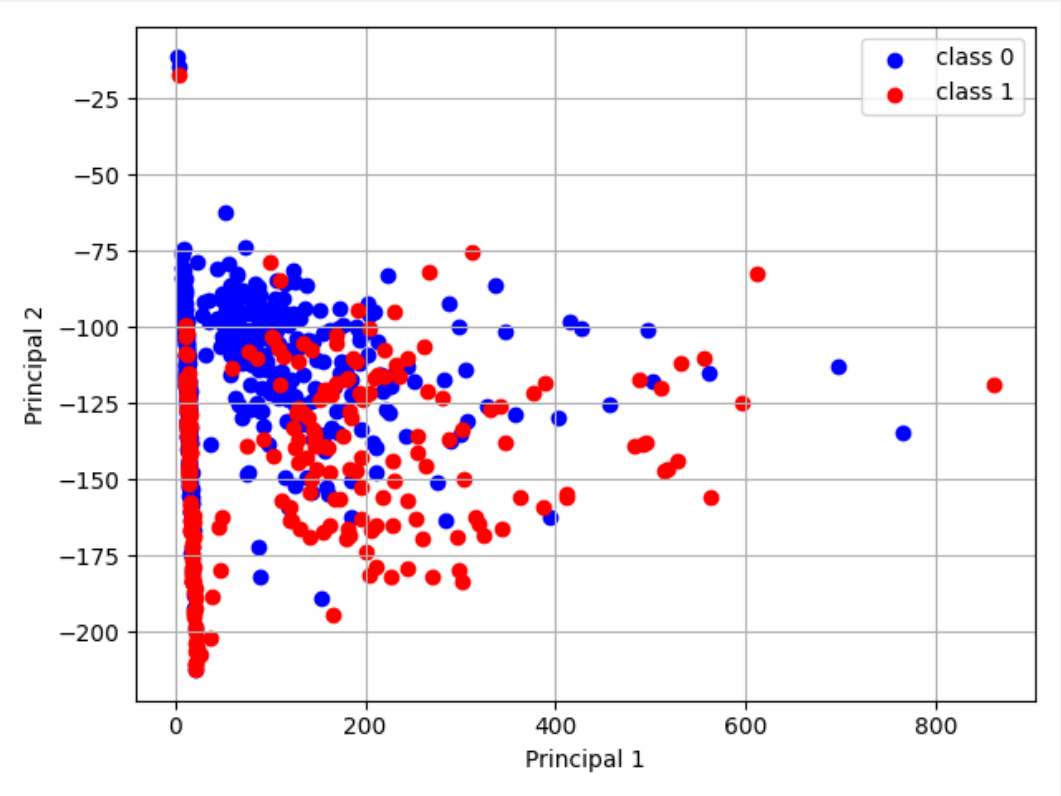
(b) The resulting figures are shown as follows.



Question 1 continued on next page...

(*continued from the previous page for Q1.5*)

(c) The resulting figures are shown as follows.



(d) The correlation coefficients between each attribute and each of the first and second principal components is shown as the following table.

Correlation coefficients	A0	A1	A2	A3	A4	A5	A6	A7	A8
First principal component	0.38559	-0.04583	-0.05705	0.18579	0.45924	0.99968	0.10057	0.23230	-0.00157
Second principal component	-0.91429	-0.09077	-0.22546	-0.07987	0.09717	0.02409	-0.25539	-0.17261	-0.37344

Question 1 continued on next page...

1.6 (4 points) We now standardise the data by mean and standard deviation using the method described below, and look into how the standardisation has impacts on PCA.

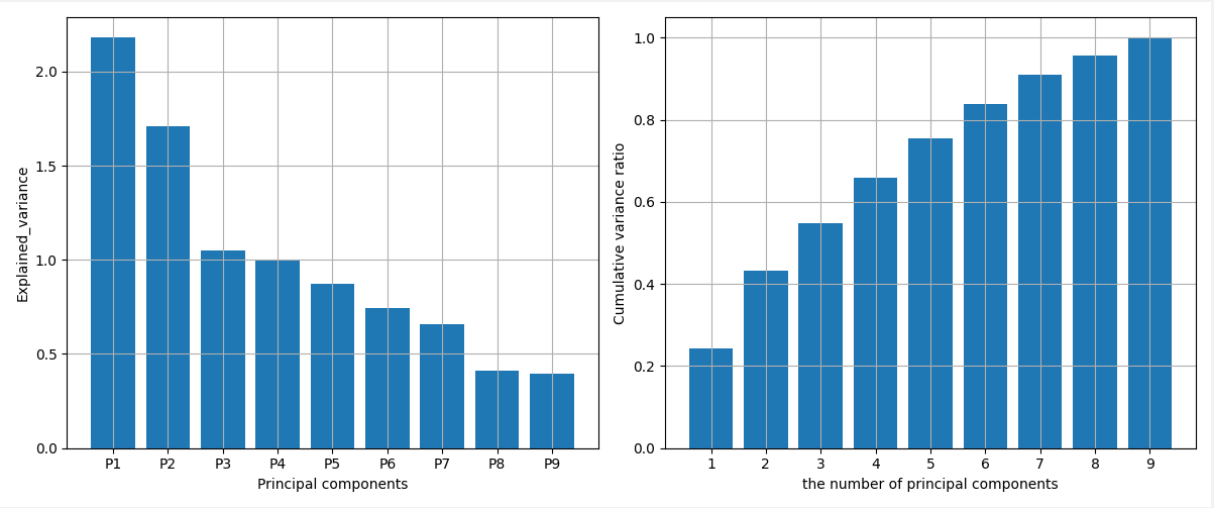
Create the standardised training data `Xtrn_s` and test data `Xtst_s` in your code in the following manner.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler().fit(Xtrn)
Xtrn_s = scaler.transform(Xtrn)      # standardised training data
Xtst_s = scaler.transform(Xtst)      # standardised test data
```

Using the standardised data `Xtrn_s` instead of `Xtrn`, answer the questions (a), (b), (c), and (d) in 1.5.

(a) The total amount of unbiased sample variance explained by the whole set of principal components is 9.0129.

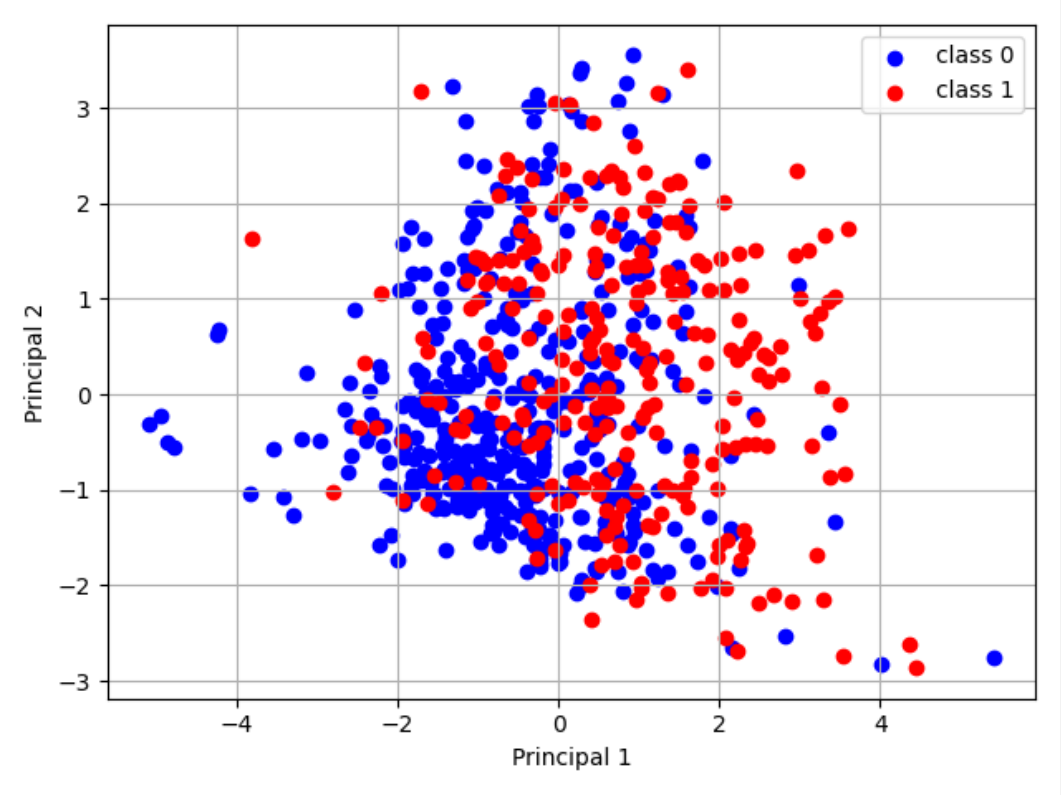
(b) The resulting figures are shown as follows.



Question 1 continued on next page...

(*continued from the previous page for Q1.6*)

(c) The resulting figures are shown as follows.



(d) The correlation coefficients between each scaled attribute and each of the first and second principal components is shown as the following table.

Correlation coefficients	A0	A1	A2	A3	A4	A5	A6	A7	A8
First principal component	0.60 067	0.05 730	0.26 795	0.36 566	0.62 304	0.62 986	0.52 285	0.65 116	0.35 286
Second principal component	0.17 743	0.10 004	0.75 998	-0.20 764	-0.46 599	-0.3 -698	0.22 423	-0.16 845	0.78 125

Question 1 continued on next page...

1.7 (7 points) Based on the results you obtained in 1.4, 1.5, and 1.6, answer the following questions.

(a) Comparing the results of 1.4 and 1.5, discuss and explain your findings.
(b) Comparing the results of 1.5 and 1.6, discuss and explain your findings and discuss (*using your own words*) whether you are strongly advised to standardise this particular data set before PCA.

(a) My findings: 1. The unbiased sample variance of each attribute of Xtrn which is 16621.8571, is very closed to the total amount of unbiased sample variance explained by the whole set of principal components which is 16645.6366; 2. The two figures obtained by 1.4(b) and 1.5(b) are as well very similar to each other, not only in the value range of y-axis but also in the trend that displayed in both figures; 3. The first principal component and the attribute A5 in the two corresponding figures both have a very large explained variance ratio which is almost near to 90%, in other words, the total variance in both method are mainly depended on the first part.
My explanation: Both 1 and 2. The results obtained by PCA method are very close to those obtained by directly using variance as the contribution, because PCA method follows the maximum variance theory and the principle of minimum dimensionality reduction loss when establishing the model. When selecting the principal component each time, PCA will select the one with the largest variance as the principal component under the condition of meeting the minimum dimensionality reduction loss, Because the largest variance means that the more information the variable contains, the more it can reflect the information of the original data. Therefore, the results obtained by directly using the variance of the original variables are very close to those obtained by PCA method. 3. This is caused by the inconsistent dimensions of various variables in the original data. When the equivalent dimensions are inconsistent, the variance of variables with a wide range of values tends to become large, although the data distribution of the variable is concentrated; The variance of variables with small range is often relatively small, although the data distribution of this variable is scattered.

(b) My findings: 1. The total amount of unbiased sample variance explained by the whole set of principal components before and after standardization are very different, the result value before standardization is 16645.6366 while the result value after standardization is 9.0129, which is a huge difference;2. Different from 1.5(b), the resulting figure obtained in 1.6(b) showed that the first principal component did not contribute a large part for total explained variance, instead it only contributes less than 30%. 3. According to the scatter plot, I find that in 1.5(c), the two principals are both ranged from a relatively large value range which caused the uniformly distribution of the dots, while in 1.6(c), the center of the value range in both x-axis and y-axis are zero, which makes the dots' distribution relatively uniformly.
Explanation for findings: Both 1 and 2. This is because in 1.6 (a), we standardized the training data, which makes the data after standardization distributed around 0 and the variance of each variable smaller. 3. This is because standardization makes the data distributed around 0 and makes the x-axis and y-axis have the same scale. Therefore, visually, the distribution of samples in 1.6 (c) will be more uniform.
At last, I strongly advised to standardize this particular data set before PCA, because this dataset has some unscaled attribute such as attribute A0 and A3, and it can not only make the obtained variance more explainable, but also make it easier for the classification model to fit the particular dataset.

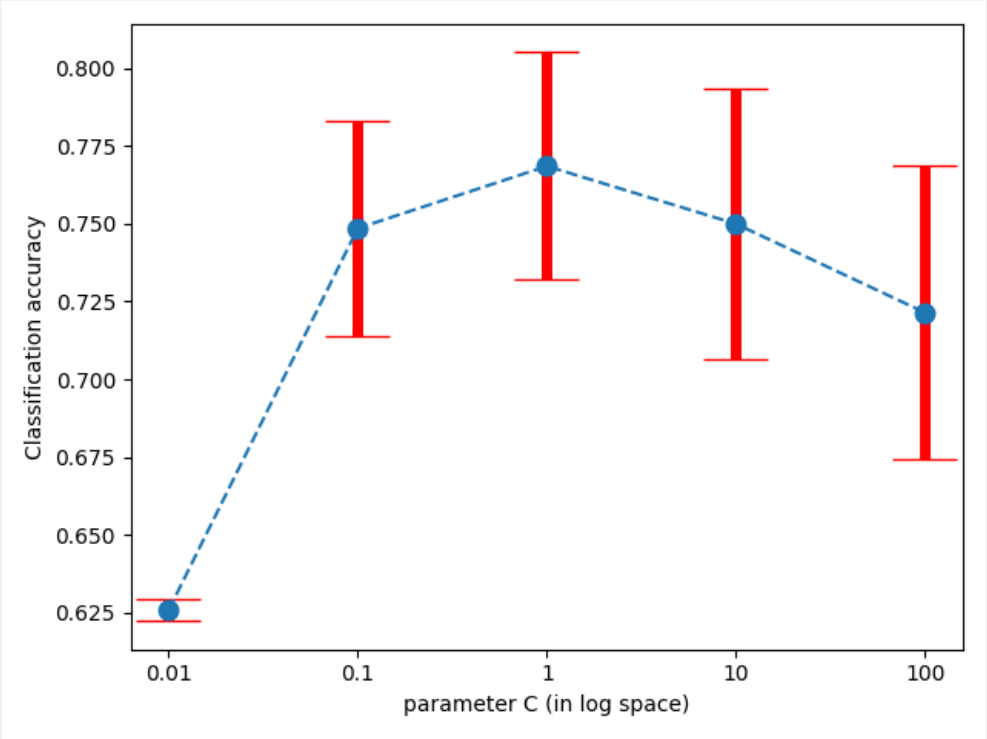
Question 1 continued on next page...

1.8 (12 points) We now want to run experiments on Support Vector Machines (SVMs) with a RBF kernel, where we try to optimise the penalty parameter C. By using 5-fold CV on the standardised training data `Xtrn_s` described above, estimate the classification accuracy, while you vary the penalty parameter C in the range 0.01 to 100 - use 13 values spaced equally in log space, where the logarithm base is 10. Use Sklearn's `SVC` and `StratifiedKFold` with default parameters unless specified. Do not shuffle the data.

Answer the following questions.

(a) Calculate the mean and standard deviation of cross-validation classification accuracy for each C, and plot them against C by using a log-scale for the x-axis, where standard deviations are shown with error bars. On the same figure, plot the same information (i.e. the mean and standard deviation of classification accuracy) for the training set in the cross validation.
(b) Comment (in brief) on any observations.
(c) Report the highest mean cross-validation accuracy and the value of C which yielded it.
(d) Using the best parameter value you found, evaluate the corresponding best classifier on the test set { `Xtst_s`, `Ytst` }. Report the number of instances correctly classified and classification accuracy.

(a) The resulting figure is shown as follows. Note that the blue dots represent the mean value of the cross-validation accuracy for each C, and the red bar with caps represent the standard deviation of the cross-validation accuracy for each C.



(b) I find that when parameter C is relatively small, the SVC model gives a low classification accuracy, but it also has a very small standard deviation which means that when C equals 0.01, the model is relatively robust than other values of C. As C increased, the classification accuracy increased and it reached the best classification accuracy when C equals to 1.0, and then, the classification accuracy went down as the parameter C continued to go up. The above observations indicate that the best parameter C is probably lies near 1.0.

(c) The highest mean cross-validation accuracy is 76.86% which is yield by C=1.0.

(d) The number of instances correctly classified by the corresponding best classifier is 75, and the classification accuracy is 75%.

Question 1 continued on next page...

1.9 (5 points) We here consider a two-dimensional (2D) Gaussian distribution for a set of two-dimensional vectors, which we form by selecting a pair of attributes, A4 and A7, in `Xtrn` (NB: not `Xtrn_s`) whose label is 0. To make the distribution of data simpler, we ignore the instances whose A4 value is less than 1. Save the resultant set of 2D vectors to a Numpy array, `Ztrn`, where the first dimension corresponds to A4 and the second to A7. You will find 318 instances in `Ztrn`.

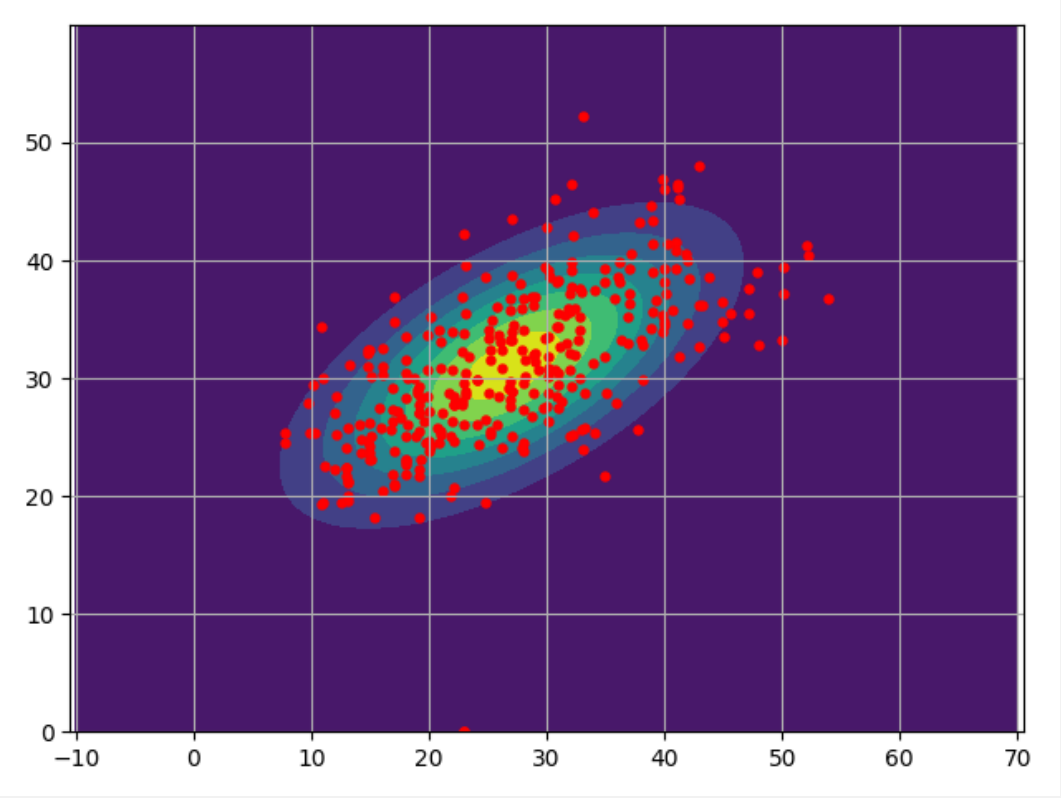
Using Numpy's libraries, estimate the sample mean vector and unbiased sample covariance matrix of a 2D Gaussian distribution for `Ztrn`. Answer the following questions.

(a) Report the mean vector and covariance matrix of the Gaussian distribution.
(b) Make a scatter plot of the instances and display the contours of the estimated distribution on it using Matplotlib's contour. Note that the first dimension of `Ztrn` should correspnd to the x-axis and the second to y-axis. Use the same scaling (i.e. equal aspect) for the x-axis and y-axis, and show grid lines.

(a) The mean vector of the Gaussian distribution is :[27.02,31.09], the covariance matrix of the Gaussian distribution is shown as the following table.

covariance matrix	A4	A7
A4	95.14113475	41.46999034
A7	41.46999034	46.69341618

(b) The scatter plot of the instances and the contours of the estimated distribution on it are shown as the following figure. Note that the red dots represent the 318 instances in Ztrn and the contour plot is obtained by using the mean vector and covariance matrix of the Gaussian distribution.



Question 1 continued on next page...

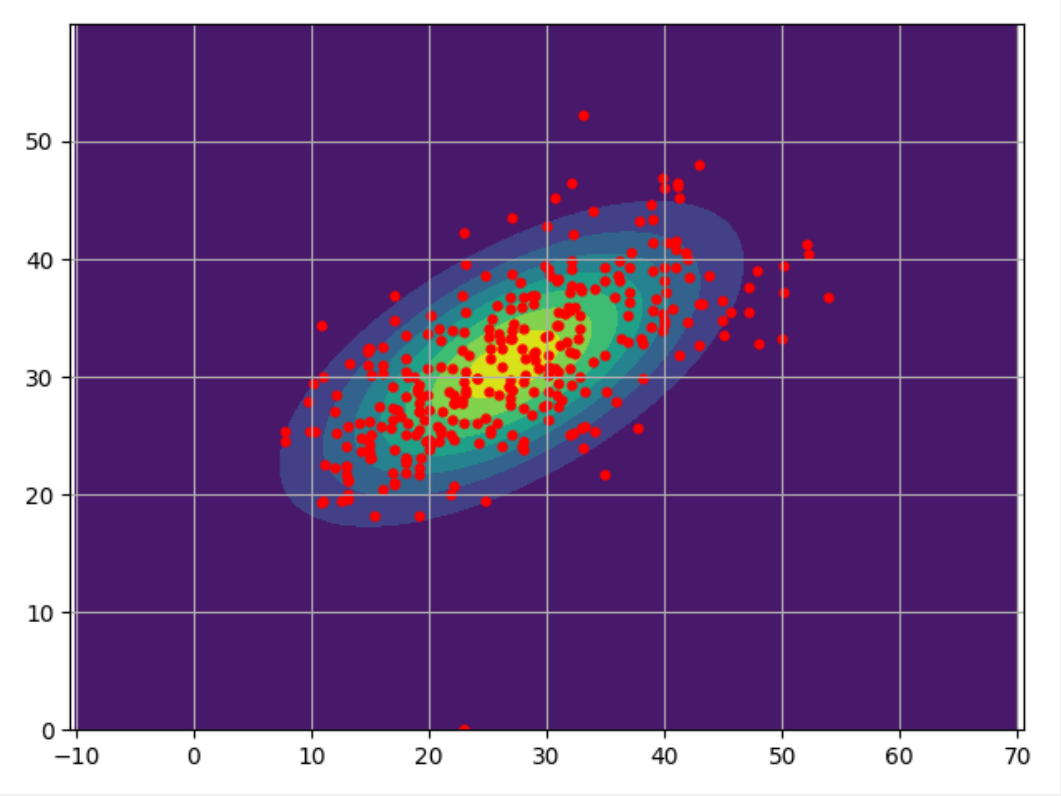
1.10 (7 points) Assuming naive-Bayes, estimate the model parameters of a 2D Gaussian distribution for the data `Ztrn` you created in 1.9, and answer the following questions.

(a) Report the sample mean vector and unbiased sample covariance matrix of the Gaussian distribution.
(b) Make a new scatter plot of the instances in `Ztrn` and display the contours of the estimated distribution on it. Note that you should always correspond the first dimension of `Ztrn` to x-axis and the second dimension to y-axis. Use the same scaling (i.e. equal aspect) for x-axis and y-axis, and show grid lines.
(c) Comparing the result with the one you obtained in 1.9, discuss and explain your findings, and discuss if it is a good idea to employ the naive Bayes assumption for this data `Ztrn`.

(a) The mean vector of the Gaussian distribution obtain by naive-Bayes is :[27.02,31.09], the covariance matrix of the Gaussian distribution is shown as the following table.

covariance matrix	A4	A7
A4	95.14113475	41.46999034
A7	41.46999034	46.69341618

(b) The scatter plot of the instances and the contours of the estimated distribution on it are shown as the following figure. Note that the red dots represent the 318 instances in Ztrn and the contour plot is obtained by using the mean vector and covariance matrix of the Gaussian distribution.



(c) It is a good idea to employ the naive Bayes assumption for this data Ztrn.

Question 1 continued on next page...

1.11 (10 points) We now consider classification with logistic regression, for which we use the standardised training data `Xtrn_s` created in 1.6. Use Sklearn's `LogisticRegression` with default parameters except for specifying '`max_iter=1000`' and '`random_state=0`'. Use Sklearn's `StratifiedKFold` with default parameters. Do not shuffle the data.

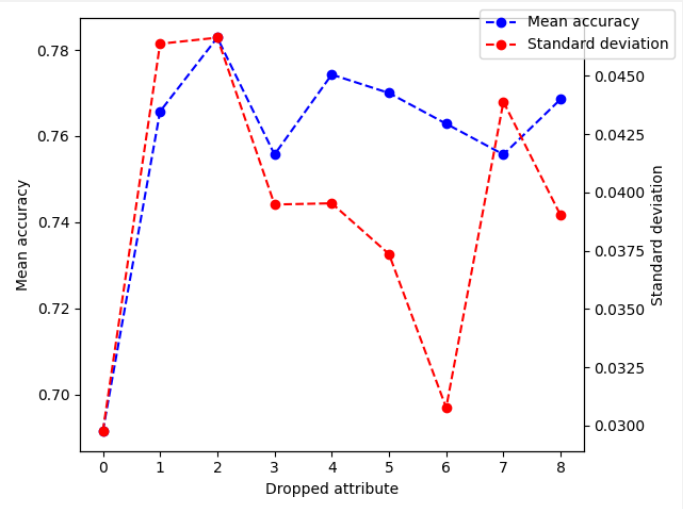
(a) Using 5-fold CV on the training set, report the mean and standard deviation of cross-validation accuracy.
(b) We consider a simple feature selection that chooses eight attributes out of the nine, i.e. dropping a single attribute. Using 5-fold CV on the training set, for each choice of attribute to drop, report the mean and standard deviation of cross-validation accuracy in a table, and report the attribute which gave the highest mean cross-validation accuracy when it was omitted.
(c) Discuss and explain your findings.

(a) The mean of cross-validation accuracy is 0.7714. The standard deviation of cross-validation accuracy: 0.0436.

(b) The resulting table is shown as follows:

Dropped attribute name	A0	A1	A2	A3	A4	A5	A6	A7	A8
Mean	0.69143	0.76571	0.78286	0.75571	0.77429	0.77000	0.76286	0.75571	0.76857
Standard deviation	0.02976	0.04638	0.04664	0.03949	0.03954	0.03736	0.03077	0.04389	0.03902

(c) In order to analysis the results above, I plot an extra figure shown as below. Note that the blue dots represent the mean of cross-validation accuracy while the red dots represent the standard deviation. As we can see through both the figure and the numeral results in 1.11(b), when drop the attribute A2 and use the left dataset to train the logistic regression model, we can obtain the best classification accuracy. However, the corresponding standardization deviation is also the largest, which means the model is relatively not steady or in other words, robust. This is probably because we didn't perform enough experiments to get a more convincing result. Another important finding is that when drop attribute A0 or A6, the standardization deviation will be very low. I think the reason is that these two attributes has an bad influence for the model, in other word, they will add some turbulence into the model if they are kept in the training dataset. Different from attribute A0, when drop attribute A6, the mean accuracy did not change too much while the standardization deviation became very low. This indicated that we can not drop A0 but we can drop A6 to train the model, in this way, we can not only decrease the standardization deviation but also keep the mean accuracy in a relatively high value.



Question 2 : (90 total points) Experiments on an image data set of handwritten letters

2.1 (5 points)

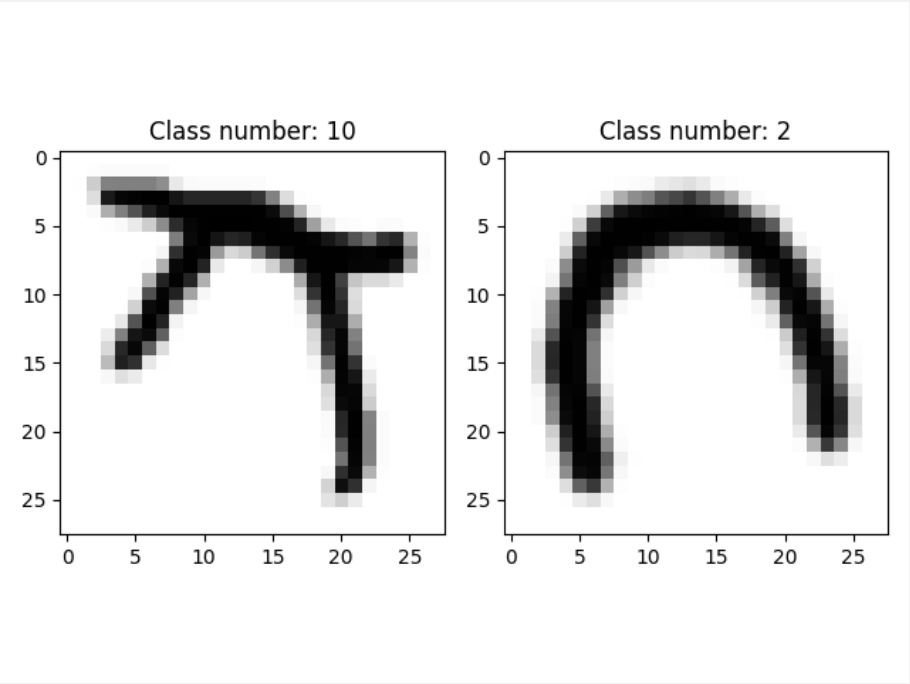
(a) Report (using a table) the minimum, maximum, mean, and standard deviation of pixel values for each `Xtrn` and `Xtst`. (Note that we mean a single value of each of min, max, etc. for each `Xtrn` and `Xtst`.)

(b) Display the gray-scale images of the first two instances in `Xtrn` properly, clarifying the class number for each image. The background colour should be white and the foreground colour black.

(a) The minimum, maximum, mean, and standard deviation of pixel valuesfor each Xtrn and Xtst are shown as follows.

	Minimum	Maximum	Mean	Standard deviation
Train set (Xtrn)	0.0	1.0	0.1774	0.3350
Test set (Xtst)	0.0	1.0	0.1756	0.3335

(b) The following figure shows the first two instances in Xtrn. The subfigure on the left corresponding to the first image in Xtrn while the right subfigure corresponding to the second image. Their class number are shown in their titles.



Question 2 continued on next page...

2.2 (4 points)

(a) `Xtrn_m` is a mean-vector subtracted version of `Xtrn`. Discuss if the Euclidean distance between a pair of instances in `Xtrn_m` is the same as that in `Xtrn`.

(b) `Xtst_m` is a mean-vector subtracted version of `Xtst`, where the mean vector of `Xtrn` was employed in the subtraction instead of the one of `Xtst`. Discuss whether we should instead use the mean vector of `Xtst` in the subtraction.

(a) The Euclidean distance between a pair of instances in Xtrn_m is the same as that in Xtrn, the demonstration is as follows.
Assuming that we have two vectors a and b randomly selected from Xtrn_m, the Euclidean distance between vector a and b are denoted as $d(a, b)$, according to the definition of Euclidean distance, we can compute $d(a, b)$ as follows:

$$d(a, b) = ((a_1 - b_1)^2 + (a_2 - b_2)^2 + ... + (a_n - b_n)^2)$$

Then, we perform mean-vector subtraction to vector a and b, and obtained two new vectors which are denoted as a^*,b^*:

$$a^* = a - mean$$
$$b^* = b - mean$$

Where mean is the corresponding mean vector.
Now, we compute the Euclidean distance between a^* and b^*:

$$d(a^*, b^*)$$
$$= \sqrt{[(a_1 - mean_1) - (b_1 - mean_1)]^2 + ... + [(a_n - mean_n) - (b_n - mean_n)]^2}$$
$$= \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + +(a_n - b_n)^2)}$$
$$= d(a, b)$$

As we can see, $d(a^*, b^*)$ is equal to $d(a, b)$, which means that The Euclidean distance between a pair of instances in Xtrn_m is the same as that in Xtrn.

(b) We should not use the mean vector of Xtst in the subtraction. Because, first of all, Xtst is a test dataset which shouldn't be used during the model construction procedure, it can only be used for testing the model's performance after all training process is done, otherwise there will be bias in the model evaluating result using this Xtst. Secondly, this processing method is similar to the one in data standardization, where you should use the mean and standard deviation of the trainset to standardize the test set instead of the mean and standard deviation of the testset. At last, the trainset and test set should follow the same distribution function, otherwise there will be unforeseeable problems when evaluating the model, in this case, we should use the mean and standard deviation of the trainset to standardize the test set.

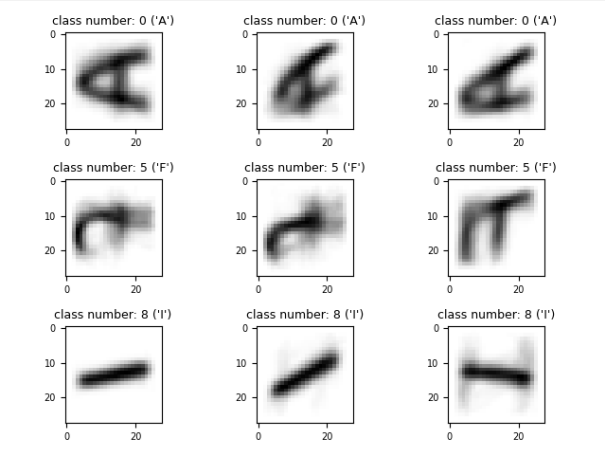
Question 2 continued on next page...

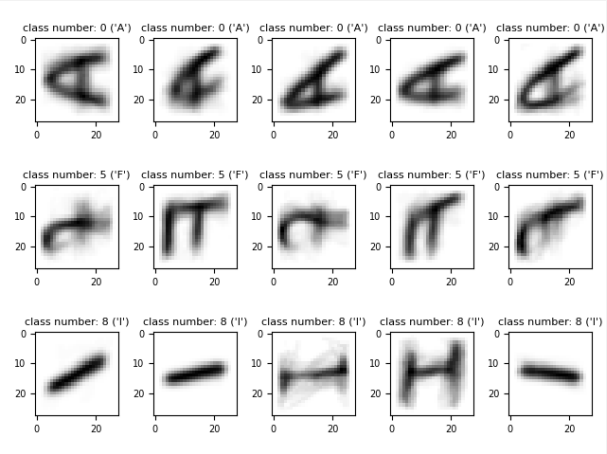
2.3 (7 points) Apply k-means clustering to the instances of each of class $0, 5, 8$ (i.e. 'A', 'F', 'I') in `Xtrn` with $k = 3, 5$, for which use Sklearn's `KMeans` with `n_clusters=`k and `random_state=0` while using default values for the other parameters. Note that you should apply the clustering to each class separately. Make sure you use `Xtrn` rather than `Xtrn_m`. Answer the following questions.

(a) Display the images of cluster centres for each k, so that you show two plots, one for $k = 3$ and the other for $k = 5$. Each plot displays the grayscale images of cluster centres in a 3-by-k grid, where each row corresponds to a class and each column to cluster number, so that the top-left grid item corresponds to class 0 and the first cluster, and the bottom-right one to class 8 and the last cluster.

(b) Discuss and explain your findings, including discussions if there are any concerns of using this data set for classification tasks.

(a) The two figures are shown as follows. Note that the first figure is corresponding to k=3, and the second figure is corresponding to k=5.





(b) My findings: 1. the images of cluster centers are all very blurry, there is no precise images edges compared to images shown in 2.1(b); 2. Some images are not shown in the upright position, instead it has been rotated 90°. 3. The cluster results clearly showed the difference between different clusters. For example, in the third row of the first figure, we can see that the main reason to separate three clusters is that they have different rotate degrees.
My explains: For the first finding, I think the blurry edge is exactly the reason why these particular images are selected as the cluster centers. Note that KMeans is a Euclidean distance-based clustering method, therefore the cluster center images' pixel values should have middle-sized values between 0 and 1 in font areas of an image, which result in relatively grey color instead of black (pixel value closed to 1) or white (pixel value closed to 0), which eventually make the figure looks blurry.
My concerns: If a image is upside down or just rotate 90°, it will totally looks like another image with difference class label, such as images with class label 'N' and 'Z'. This may cause some error classification when training and testing the classifier.

Question 2 continued on next page...

2.4 (5 points) Explain (using your own words) why the sum of square error (SSE) in k-means clustering does not increase for each of the following cases.

(a) Clustering with $k+1$ clusters compared with clustering with k clusters.
(b) The update step at time $t+1$ compared with the update step at time t when clustering with k clusters.

(a) When increasing the cluster number from k to $k+1$, the SSE will not increase. To demonstrate this theory, let's consider two special cases. In the first case, the cluster number $k = N$, where N is the total number of training samples. In this case we can easily know that SSE equals to zero, because after enough training process, every sample in the training set will be a cluster center and there will be no distance between samples and cluster centers (Because clusters centers is samples themselves). In the second case, the cluster number $k = 1$, which means there is only one cluster center and every other sample will have a large distance contribution to the SSE, clearly, in this case the SSE will be the largest. In conclusion, if we increase k to $k+1$, the distance in some particular samples may update their belonging cluster center, which means the new distance is smaller than its original one, which means SSE will decrease. In the worst situation, number of cluster centers increased from k to $k+1$, but no samples change its belonging center, so the SSE keeps same. All in one word, SSE will not increase when increasing the cluster number from k to $k+1$.

(b) Whether update at time $t+1$ or at time t, the terminate condition is reach the maximum epochs or the cluster centers have no significant change. In any case, the final SSE is the same.

Question 2 continued on next page...

2.5 (11 points) Here we apply multi-class logistic regression classification to the data. You should use Sklearn's `LogisticRegression` with parameters '`max_iter=1000`' and '`random_state=0`' while use default values for the other parameters. Use `Xtrn_m` for training and `Xtst_m` for testing. We do not employ cross validation here. Carry out a classification experiment.

(a) Report the classification accuracy for each of the training set and test set.
(b) Find the top five classes that were misclassified most in the test set. You should provide the class numbers, corresponding alphabet letters (e.g. A,B,...), and the numbers of misclassifications.
(c) For each class that you identified in the above, make a quick investigation and explain possible reasons for the misclassifications.

(a) The classification accuracy for the training set is 91.62%;
The classification accuracy for the test set is 72.23%.

(b) Top five classes that were misclassified most in the test set is shown as follows:
Top 5 misclassified class numbers: 11, 17, 8, 10, 13;
Top 5 misclassified alphabet letters: 'L', 'R', 'I', 'K', 'N';
Corresponding numbers of top 5 misclassifications: 53, 48, 42, 38, 36.

(c)) For class 11 (corresponding to the alphabet letter 'L'), it's image sometimes looks very similar to the letter 'I' and 'J', especially 'J'. More specifically, the classifier can not recognize the inversion information of the image which makes the 'L' classification subproblem the most difficult among all 26 alphabet letters.
For class 17 (corresponding to the alphabet letter 'R'), its image is relatively more complicated than the other letters.
For class 8 (corresponding to the alphabet letter 'I'), this is similar to letter 'L', and the other reason is that letter 'I' is too simple (compared with letter 'R'), and it only has one straight line of pixels, which indicate that the information is not enough.
For class 10 (corresponding to the alphabet letter 'K'), its image sometimes looks very similar to the letter 'k', which caused a lot of misclassified examples.
For class 13 (corresponding to the alphabet letter 'N'), its image also has a rotate problem. More specifically, if an image of letter 'N' rotate 90°, it becomes almost the same as letter 'Z'.
I think this is why this letter had a low classification performance.

Question 2 continued on next page...

2.6 (20 points) Without changing the learning algorithm (i.e. use logistic regression), your task here is to improve the classification performance of the model in 2.5. Any training and optimisation (e.g. hyper parameter tuning) should be done within the training set only. Answer the following questions.

(a) Discuss (using your own wards) three possible approaches to improve classification accuracy, decide which one(s) to implement, and report your choice.
(b) Briefly describe your implemented approach/algorithm so that other people can understand it without seeing your code. If any optimisation (e.g. parameter searching) is involved, clarify and describe how it was done.
(c) Carry out experiments using the new classification system, and report the results, including results of parameter optimisation (if any) and classification accuracy for the test set. Comments on the results.

(a) As we can see from the above results, the test set accuracy is much lower than the training set accuracy, which indicate that there is an overfitting problem. Therefore, we proposed the following three approaches to address this overfitting problem.
1. Train and test the model using k-fold cross validation. Low accuracy in test set is probably because this accuracy we obtained is not accurate. K-fold cross validation method gives the model a global observation and let the model learn classification skills from the whole dataset, which is very helpful to obtained a more accurate classification accuracy.
2. Increase the number of training samples. The overfitting problem is probably caused by the lack of sample diversity during training process, which means that some kind of samples is very unfamiliar to the model. So we can increase the diversity of training set by increase the number of training set.
3. Find the best parameters using Grid Search. The regularization parameter C and maximum iteration number N are two key parameters in Logistic regression. Therefore we use Grid Search method to search the best combination of these two parameters.

(b) We choose the third approach to improve the classification accuracy. The details are shown as follows:
1. Determine the range of the two parameters respectively. For maximum iteration we set 5 independent values which are: 100, 200, 500, 1000, 1500; For regularization parameter C we set 5 independent values which are: 0.01, 0.05, 0.1, 0.7, 1.0.
2. Initialize the accuracy cache array which has the shape $(n1, n2)$, where n1 is the total number of the first parameter and n2 is the total number of the second parameter.
3. Run the whole training and testing process for $n1 * n2$ times, each time using the corresponding parameters in the parameter grid, which is a 5-by-5 Numpy array in our case.
4. After all evaluations using all parameter combinations obtained from the parameter grid, we can get a result array which has the same shape as the grid. Find the highest value in the result array and extract the corresponding parameters to construct the Logistic Regression classifier.
5. After we obtained the classifier with the best parameter combination, we can then continue to train and test the classifier which will give a better classification performance due to the best parameters found by Grid Search.

Question 2 continued on next page...

(*continued from the previous page for Q2.6*)

(c) The Grid search result is shown as the following table:

	0.01	0.05	0.1	0.7	1.0
100	0.7208	0.7535	0.7473	0.7308	0.7215
200	0.7208	0.7535	0.7473	0.7308	0.7215
500	0.7208	0.7535	0.7473	0.7308	0.7215
1000	0.7208	0.7535	0.7473	0.7308	0.7215

As we can see, the maximum iteration number has no influence on the test accuracy , therefore we just select the minimal number as the best parameter because the computing time cost is the lowest. As for the parameter C, the best value is 0.05, in this case the model has the best test accuracy, and the best test accuracy is 75.35%.
The above results indicate that Grid search is very useful to find the promising parameter combinations, it can improve the classification performance of the classifier.

Question 2 continued on next page...

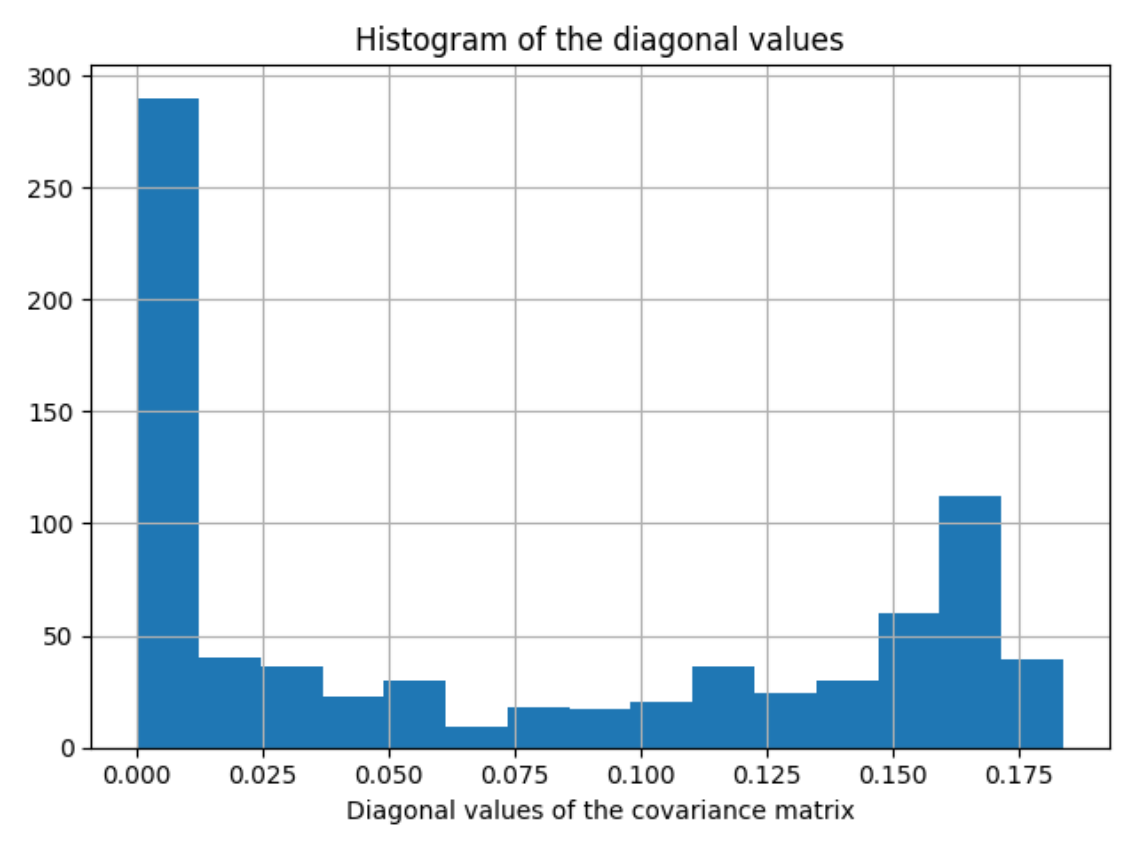
2.7 (9 points) Using the training data of class 0 ('A') from the training set `Xtrn_m`, calculate the sample mean vector, and unbiased sample covariance matrix using Numpy's functions, and answer the following.

(a) Report the minimum, maximum, and mean values of the elements of the covariance matrix.
(b) Report the minimum, maximum, and mean values of the diagonal elements of the covariance matrix.
(c) Show the histogram of the diagonal values of the covariance matrix. Set the number of bins to 15, and use grid lines in your plot.
(d) Using Scipy's `multivariate_normal` with the mean vector and covariance matrix you obtained, try calculating the likelihood of the first element of class 0 in the test set (`Xtst_m`). You will receive an error message. Report the main part of error message, i.e. the last line of the message, and explain why you received the error, clarifying the problem with the data you used.
(e) Discuss (using your own words) three possible options you would employ to avoid the error. Note that your answer should not include using a different data set.

(a) The minimum of the elements of the covariance matrix: -0.097474.
The maximum of the elements of the covariance matrix: 0.183786.
The mean values of the elements of the covariance matrix: 0.001709.

(b) The minimum of the diagonal elements of the covariance matrix: 0.0.
The maximum of the diagonal elements of the covariance matrix: 0.183786.
The mean values of the diagonal elements of the covariance matrix: 0.072313.

(c) The histogram of the diagonal values of the covariance matrix is shown as below.



Question 2 continued on next page...

(*continued from the previous page for Q*)

(d) The main part of error message is: 'numpy.linalg.LinAlgError: singular matrix'.The reason why I receive this error is that the 'numpy.linalg' method that I used in this case is not allowed to use a singular matrix, whose determinant is equals to zero. The deeper reason that caused this error comes from the dataset that we used, that is, there is too much zeros in the dataset so that when we use Scipy's multivariate_normal method and compute the determinant, the result will be zero, which is not allowed. The character that our dataset has a large number of zeros can not be changed, therefore we must find another way to fix the error, that will be discussed more specifically in 2.7(e).

(e) There are many methods that can avoid this error. For instance:
1. Add a very small number to every diagonal values of the covariance matrix, this will enforce the diagonal values not equal to zeros, so that when we compute determinant, there will be no error occurred. But note that, in 'numpy.linalg' method, there is a threshold named 'eps' that control the confidence of whether the value is greater than zeros, therefore the small number we add must be at least greater than this threshold 'eps'.
2. We can just simply change the passing parameter 'allow_singular' equals to 'True' when using the Scipy's multivariate_normal function. This simple change allows the Scipy to deal with the singular matrix with an particular way.
Finally, after the above methods we fixed this error, and the computation result of the likelihood is as follows: 'The likelihood of the first element of class 0 in the test set is: 4.3079893181065306e-176'.

Question 2 continued on next page...

2.8 (8 marks) Instead of Scipy's `multivariate_normal` we used in 2.7, we now use Sklearn's `GaussianMixture` with parameters, `n_components=1, covariance_type='full'`, so that there is a single Gaussian distribution fitted to the data. Use { `Xtrn_m`, `Ytrn` } as the training set and { `Xtst_m`, `Ytst` } as the test set.

(a) Train the model using the data of class 0 ('A') in the training set, and report the log-likelihood of the first instance in the test set with the model. Explain why you could calculate the value this time.

(b) We now carry out a classification experiment considering all the 26 classes, for which we assign a separate Gaussian distribution to each class. Train the model for each class on the training set, run a classification experiment using a multivariate Gaussian classifier, and report the number of correctly classified instances and classification accuracy for each training set and test set.

(c) Briefly comment on the result you obtained.

(a) The log-likelihood of the first instance in the test set is: 1.
This time the error that occurred in 2.7(d) didn't occur again, because this time we used the Gaussian Mixture Model (GMM) as our classifier. Specifically, in GMM we define the parameter covariance_type='full' which means each component has its own general covariance matrix where every item is not equal to zeros. Therefore, the determinant that computed later will not be zeros either, and this small change avoid the error from occurrence.

(b) The number of correctly classified instances and classification accuracy for each training set and test set are shown as the following table.

Class number	Trainset Accuracy	Correct number	Testset Accuracy	Correct number
0 (A)	1.0	300	1.0	300
1 (B)	1.0	300	1.0	300
2 (C)	1.0	300	1.0	300
...	...	...	...	...
25 (Z)	1.0	300	1.0	300

(c) As we can see from the above table, the results seem to be perfect because we performed a Gaussian Mixture model on every class. As a matter of fact, this problem is now a binary classification problem because we only determine whether the sample belongs to the given class or not, by using the Gaussian probability that model predicted.
Note that GMM is a clustering method instead of a classification method, which means that we can only get a cluster result and we can't get the predict label from the GMM. When it comes to multi-class problem such as 26 hand-written alphabet letters recognize problem, we can only get a clustering result from the GMM. However, we can make it achieve the classification mission through some numerical computation by programming in python.

Question 2 continued on next page...

2.9 (6 points) Answer the following question on Gaussian Mixture Models (GMMs).

(a) Explain (using your own words) why Maximum Likelihood Estimation (MLE) cannot be applied to the training of GMMs directly.

(b) The Expectation Maximisation (EM) algorithm is normally used for the training of GMMs, but another training algorithm is possible, in which you employ k-means clustering to split the training data into clusters and apply MLE to estimate model parameters of a Gaussian distribution for each cluster. Explain the difference between the two algorithms in terms of parameter estimation of GMMs.

(a) Maximum likelihood estimation involves treating the problem as an optimization or search problem in which we look for a set of parameters that result in the joint probability of the most suitable data sample. One limitation of maximum likelihood estimation is that it assumes that the data set is complete, or fully observed. This does not mean that the model has access to all data; Instead, it assumes the existence of all variables relevant to the problem. But this is not always the case. Some data sets can only observe a few relevant variables, while others cannot, and they remain hidden, even though they affect other random variables in the data set.

(b) In terms of parameter estimation of GMMs, the main difference between the two algorithms is that the first one (EM) can only obtained an approximate numerical solution, while the second one (k-means MLE) can obtained a precise analytic solution. The above difference is caused by the nature of EM and MLE, the former is an iterated method while MLE is an accurate statistical method.

Question 2 continued on next page...

2.10 (15 points) We now extend the classification with a separate multivariate Gaussian model for each class that we performed in 2.8 to one with Gaussian Mixture Model (GMM) per class. To achive this, change the number of mixture components in Sklearn's `GaussianMixture`. To simplify the experiment, do not use cross validation, but instead use the test set as a validation set. Use `random_state=0` when you call Sklearn's `GaussianMixture`.

(a) Run experiments with GMMs for $k = 1, 2, 4, 8$ (where k is the number of mixture components).
  (i) Report classification accuracy for each of the training set and test set in a single table.
  (ii) Describe and briefly explain your findings.

(b) Using GMMs with $k = 2$, optimise the parameter '`reg_covar`' for the test set.
  (i) Report the result, i.e. the highest test-set accuracy and the value of the parameter value of `reg_covar` that yields it.
  (ii) Briefly discuss the '`reg_covar`' parameter in the context of this data set.

(a)

(i) Classification accuracy for each of the training set and test set are shown as below.

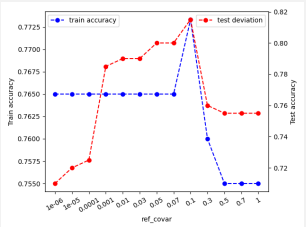
k	1	2	4	8
Train accuracy	1.000	0.765	0.122	0.202
Test accuracy	1.000	0.710	0.135	0.204

(ii) From the above table I find that when $k = 1$, both train and test accuracy are 100%, which means the model can classifier the samples without misclassification. This is because in this case the dataset is very simple and the model just need to approximate one gaussian distribution parameters. As k increased, both accuracy decreased, this is because the model get more and more complicated and it's more hard to approximate multiple gaussian distribution parameters. When $k = 8$, both accuracy becomes very low, which means the current model has too much parameters to approximate but failed.

(b)

(i) The optimization result of the parameter 'reg_covar' is shown as follows.
The highest test-set accuracy is 0.815, and the value of the parameter value of reg_covar that yields the best accuracy is: 0.1.

(ii) For optimize the the parameter 'reg_covar', I used the 1-dimentional grid search. I select 13 values of this parameter range from 1e-6 to 1.0. According to my initial experiment, the best parameter lies around 0.1, so I split the range more precisely around 0.1. Finally, as shown in the following figure, the best parameter 'reg_cpvar=0.1' is found.



The parameter 'reg_covar' represents the non-negative regularization added to the diagonal of covariance, which allows to assure that the covariance matrices are all positive. The default value of this parameter equals to 1e-6, however the best value found by experiment is 0.1, which is a tremendous difference! The reason is that this image dataset of handwritten letters is very sparse, it has a lot of zeros and the corresponding values of parameter 'reg_covar' should be relatively large than its default value, otherwise the classification accuracy will be influenced.