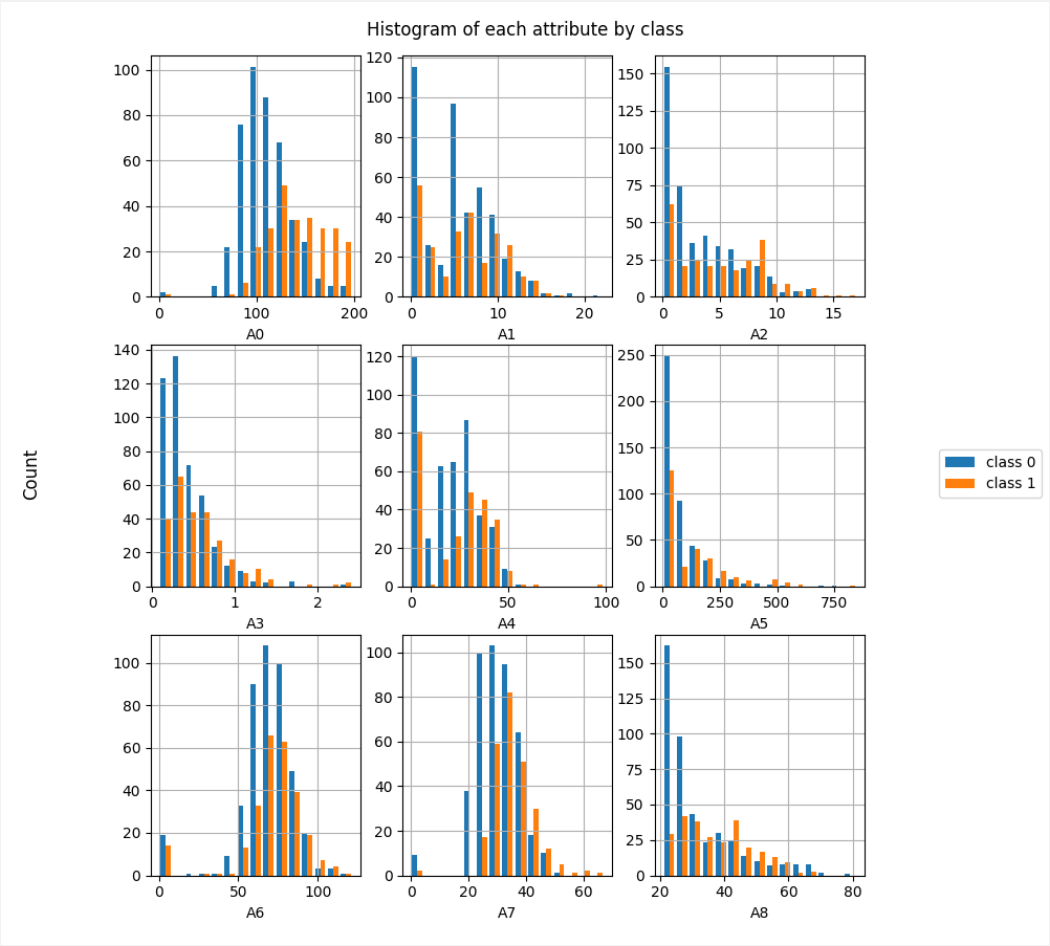
Question 1 : (70 total points) Experiments on a binary-classification data set

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.



Considering that we are dealing with medical data set, people will have different severity and conditions towards certain disease due to different genetics. Now we proceed to describe the observations, Attributes 0, 6 and 7 seems to follow the Gaussian distribution. In Attribute 0, those patients diagnosed with a particular disease are more likely to show the symptoms in Attribute 0. However, Attribute 6 and 7 seems to contain outliers from the distribution which can cause the variance to increase. Attributes 1, 2, 3, 5 and 8 seems to be skewed towards left (zero), where most of the observations were reported near zero. Indicating that some patients do not have certain symptom towards the disease.

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)

Attribute	A0	A1	A2	A3	A4	A5	A6	A7	A8
Correlation Coeff	0.491	0.0874	0.227	0.207	0.108	0.186	0.0763	0.304	0.240

(b) Yes it is a good idea, as the attributes with large correlations will represent and explain the model well.

(c) No, we should not ignore all the attributes with small correlations. Removing some attributes may increase the accuracy of the training set but our aim is to build a model that classify patient's disease and fit the model. Also, the small linear correlation may indicate that the non-linear model can be suitable for 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 $b > 0, d > 0$.

We let $x_i = a + bu_i$ and $y_i = c + dv_i$, then we can write the mean of them as $\bar{x} = a + b\bar{u}$ and $\bar{y} = c + d\bar{v}$ respectively. Thus subtracting each values $(x_i - \bar{x}) = b(u_i - \bar{u})$ and $(y_i - \bar{y}) = d(v_i - \bar{v})$ Now we express the correlation coefficient of x and y

$$r_{xy} = \frac{cov(x, y)}{\sigma_x \cdot \sigma_y} = \frac{\mathbf{E}((x_i - \bar{x})(y_i - \bar{y}))}{\sqrt{\mathbf{E}(x_i - \bar{x})^2}\sqrt{\mathbf{E}(y_i - \bar{y})^2}} = \frac{\mathbf{E}(b(u_i - \bar{u}) \cdot (d(v_i - \bar{v})))}{\sqrt{\mathbf{E}(b^2(u_i - \bar{u})^2)}\sqrt{\mathbf{E}(d^2(v_i - \bar{v})^2)}}$$

$$= \frac{\mathbf{E}((u_i - \bar{u}) \cdot (v_i - \bar{v}))}{\sqrt{\mathbf{E}(u_i - \bar{u})^2}\sqrt{\mathbf{E}(v_i - \bar{v})^2}} = r_{uv}$$

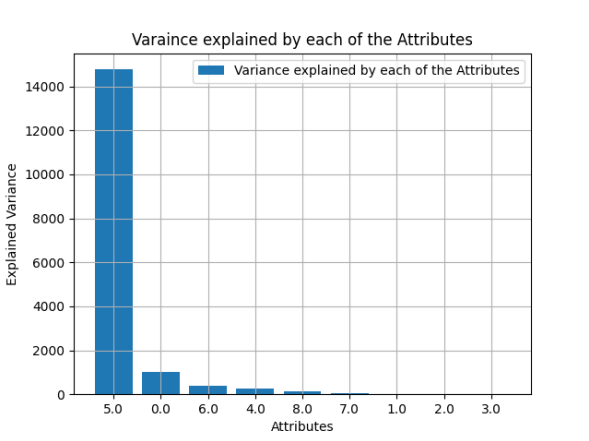
Thus we prove that r_{uv} is translation and scale invariant.

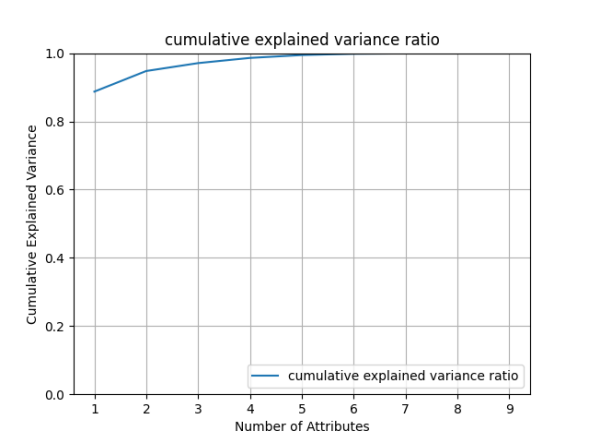
1.4 (5 points) Calculate the unbiased sample variance of each attribute of `Xtrn`, and sort the variances in decreasing order. Answer the following questions.

(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 1.665×10^4

(b) The two graphs are shown below





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.

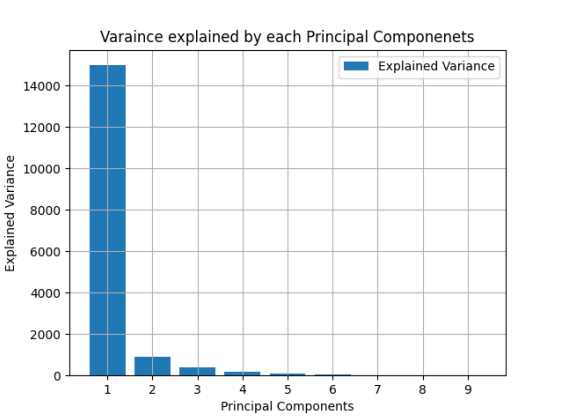
- A graph of the amount of variance explained by each of the principal components.
- A graph of the cumulative variance ratio, where the range of y-axis should be [0, 1].

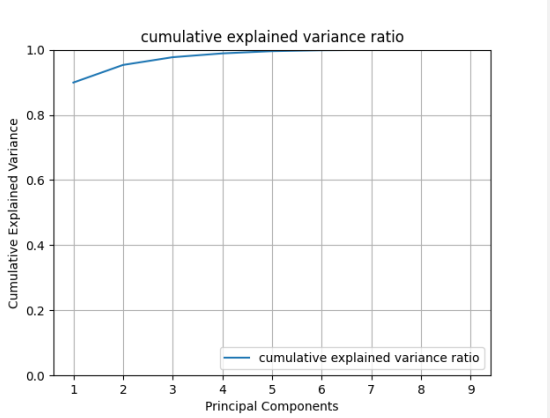
(c) Mapping all the instances in `Xtrn` on to the 2D space spanned with the first two principal components, and plot a scatter graph of the instances on the space, where instances of class 0 are displayed in blue and those of class 1 in red. Use grid lines. Note that the mapping should be done directly 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 the unbiased sample variance explained by the whole set of principal components is 1.665×10^4

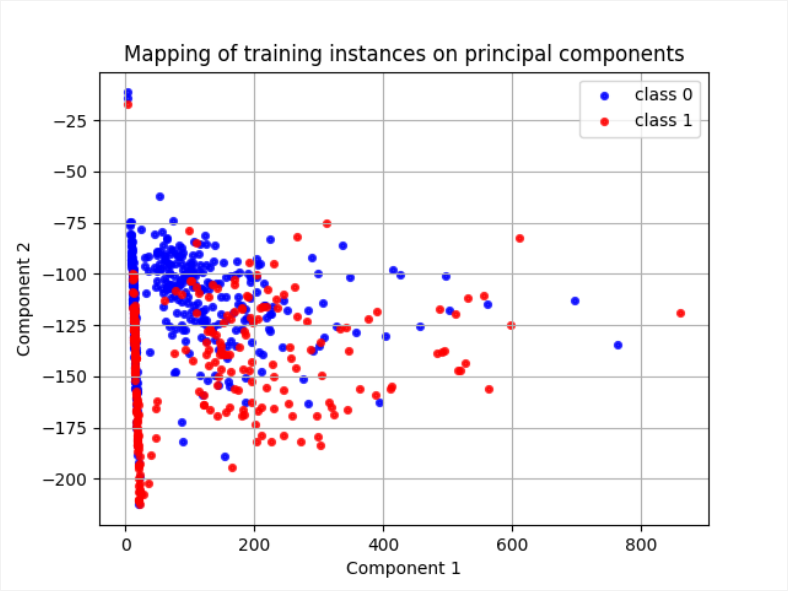
(b) The plots are shown below,





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

(c) The plot is shown below,



(d) The table is shown below

Attribute	1st Component	2nd Component
A0	0.386	-0.914
A1	-0.0458	-0.0908
A2	-0.0571	-0.225
A3	0.186	-0.0799
A4	0.459	0.0972
A5	0.999	0.0241
A6	0.101	-0.255
A7	0.232	-0.173
A8	-0.00157	-0.373

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.

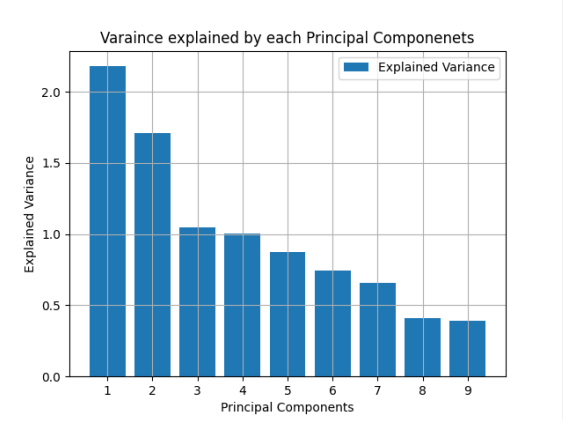
Create the standardised training data `Xtrn_s` and test data `Xtst_s` in your code in the following manner.

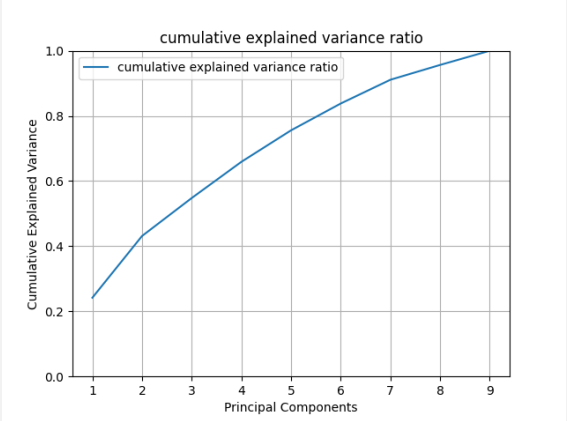
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler().fit(Xtrn)
Xtrn_s = scaler.transform(Xtrn)      # standardised training data
Xtst_s = scaler.transform(Xtst)      # standardised test data
```

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

(a) The total sum of the unbiased sample variance is 9.01

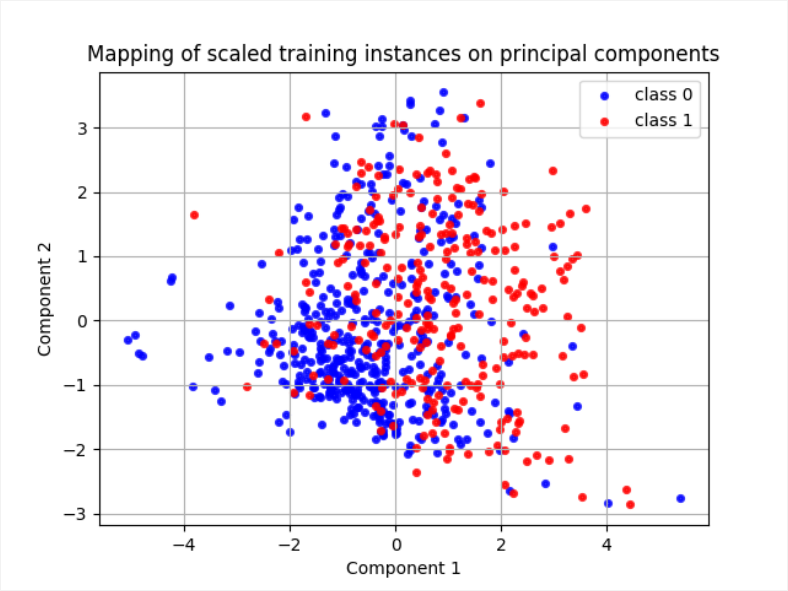
(b) The plots are shown below,





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

(c) The plot is shown below,



(d) The table is shown below

Attribute	1st Component	2nd Component
A0	0.601	0.177
A1	0.0573	0.100
A2	0.268	0.760
A3	0.366	-0.208
A4	0.623	-0.466
A5	0.630	-0.370
A6	0.523	0.224
A7	0.651	-0.168
A8	0.353	0.781

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) The values and plots are similar. Looking at the variance plot, we can see that most variance was explained by `Attribute 5` in 1.4 and 1st component in 1.5. Looking at the cumulative explained variance ratio, both plots have steep increase and contain approximately 95%. This allows that we can use a single attribute or single component to project our data set.

(b) The values and plots are different. In 1.6, we can see there is gradual decrease in the number of explained variance coverage by each principal component. Furthermore, the curve increases gradually in 1.6 whereas the plot in 1.5 increases steeply and quickly plateaus. Thus the first component represents 90% of the variance but since we are using a medical data, reducing to single dimension will not be able to represent the entire data set well. Comparing the pca plot, we can see that the points in 1.5 are concentrated on left and there is no clear dividend between the class. On the other hand, the points in 1.6 are well scattered and we can clearly differentiate the classes. Lastly comparing the coefficients, in 1.6 the absolute value of the correlation coefficient increases and it represents stronger relationship. We are strongly advised to standardise the data set before PCA. If we do not introduce standardisation, this will result in different variance and thus a high standard deviation will lead to higher weight for the calculation of the PCA components. By standardising it, we can keep the same weight for all variables for the calculation.

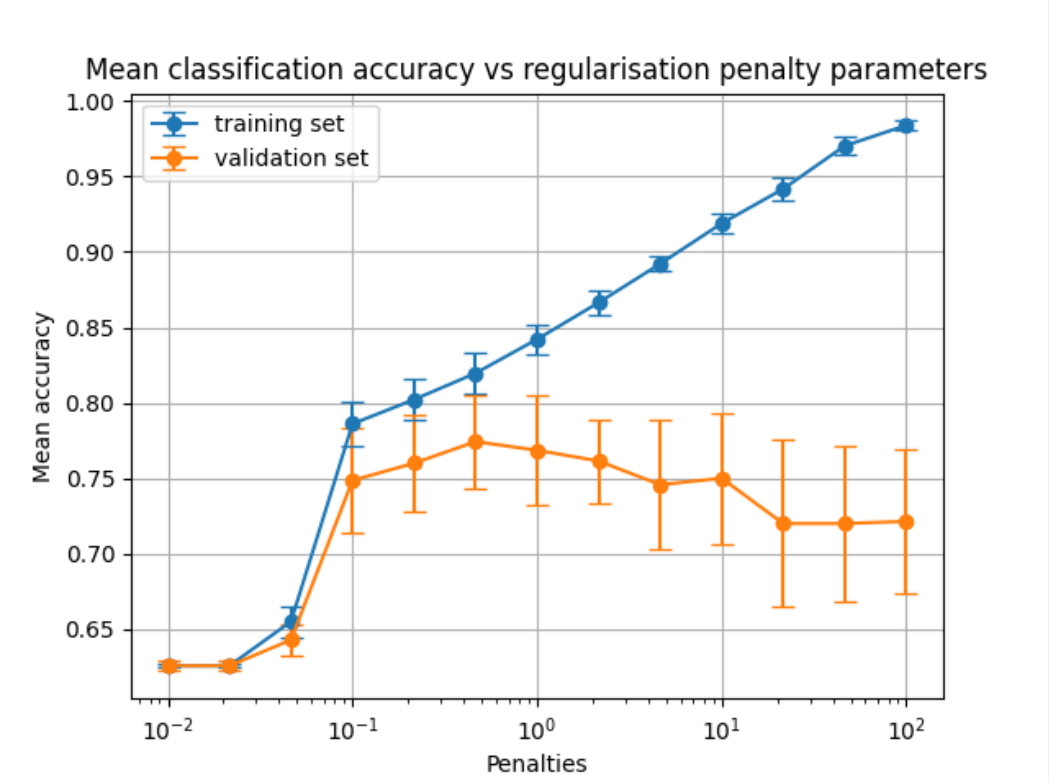
Question 1 continued on next page...

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 plot is shown below,



(b) As the value of C increases, the accuracy of training set increase with a trend and with $C = 100$ we see that the accuracy is near 1.00. On the other hand, the accuracy of the validation set changes along the value of penalties. Higher value of C refers to higher weight on the training data, and a lower weight to the penalty.

(c) The highest mean cross-validation accuracy is 0.774 with the value of $C = 0.464$

(d) The classification accuracy is 0.75 with 75 instances correctly classified for the test set.

Question 1 continued on next page...

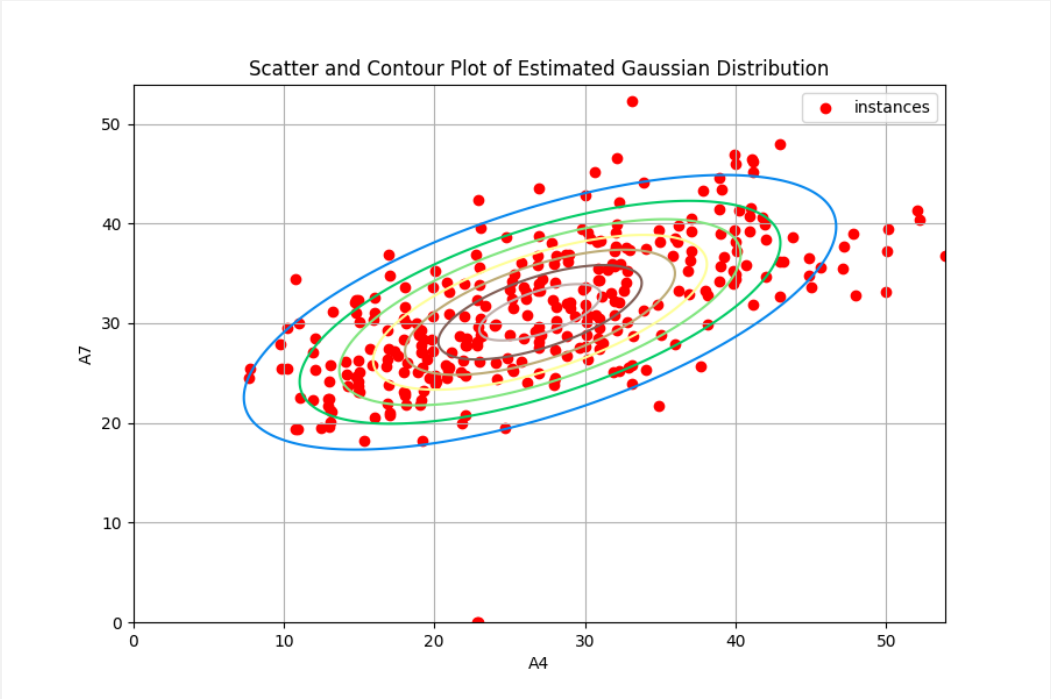
1.9 (5 points) We here consider a two-dimensional (2D) Gaussian distribution for a set of two-dimensional vectors, which we form by selecting a pair of attributes, A4 and A7, in `Xtrn` (NB: not `Xtrn_s`) whose label is 0. To make the distribution of data simpler, we ignore the instances whose A4 value 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 is $\begin{bmatrix}27.0 & 31.1\end{bmatrix}$ and the covariance matrix is $\begin{bmatrix}95.1 & 41.2\\ 41.2 & 46.7\end{bmatrix}$

(b) The scatter plot is shown below,

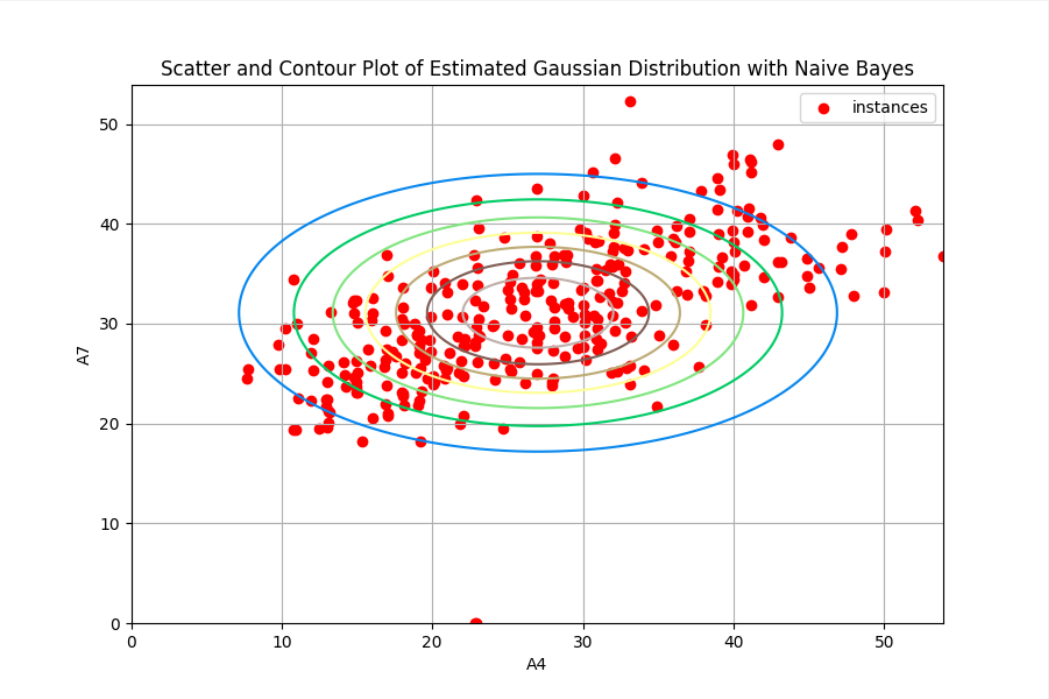


1.10 (7 points) Assuming naive-Bayes, estimate the model parameters of a 2D Gaussian distribution for the data `Ztrn` you created in 1.9, and answer the following questions.

(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 is $\begin{bmatrix}27.0 & 31.1\end{bmatrix}$ and the covariance matrix is $\begin{bmatrix}94.8 & 0\\ 0 & 46.5\end{bmatrix}$

(b) The scatter plot is shown below,



(c) It is a not a good idea to employ the naive Bayes assumption for this data `Ztrn`. Comparing against the contour plot in 1.9, the Gaussian distribution with Naive Bayes assumption does not represent the data well. This is because it assumes independence between the `Attribute 4` and `Attribute 7`. Since we are employing medical data set, clearly it is hard to assume the independence of certain symptom or observation and there exist a higher possibility of correlation between symptions towards a certain disease.

Question 2 : (75 total points) Experiments on an image data set of handwritten letters

2.1 (5 points)

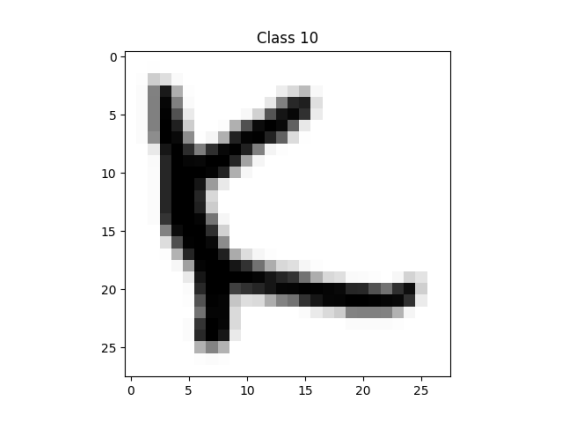
(a) Report (using a table) the minimum, maximum, mean, and standard deviation of pixel values for each `Xtrn` and `Xtst`. (Note that we mean a single value of each of min, max, etc. for each `Xtrn` and `Xtst`.)

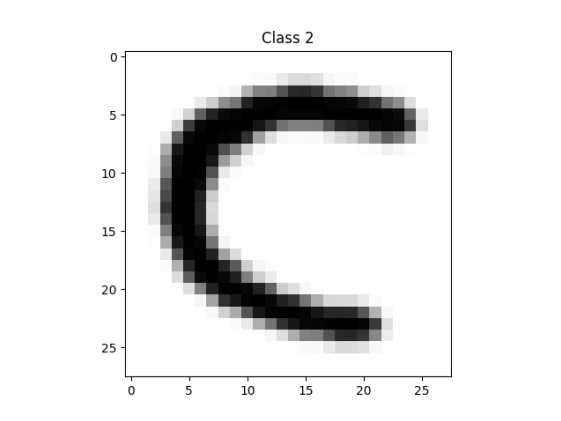
(b) Display the gray-scale images of the first two instances in `Xtrn` properly, clarifying the class number for each image. The background colour should be white and the foreground colour black.

(a) The table is shown below,

	Min	Max	Mean	Std Dev
Training	0.0	1.0	0.177	0.335
Testing	0.0	1.0	0.176	0.333

(b) The images are shown below,





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 `Xtrn` and `Xtrn_m` are exactly the same. Normalising with mean on instances are known as mean vector subtracted. Thus subtracting two instances will cancel the mean values and eventually computing the same distance. Hence, the computation of the euclidean distance will not depend on the choice of `Xtrn` and `Xtrn_m`.

(b) We can clearly see the difference in the mean vectors of `Xtrn` and `Xtst` as the values are not identical. However, we only want to use the mean vector of the training set. This is because we want to keep the testing data to be a new set of data this means that we are testing the test set with prior knowledge of the data set leading inaccuracy sampling errors may negatively bias the predictions our aim is to test and evaluate whether our model can fit the testing data.

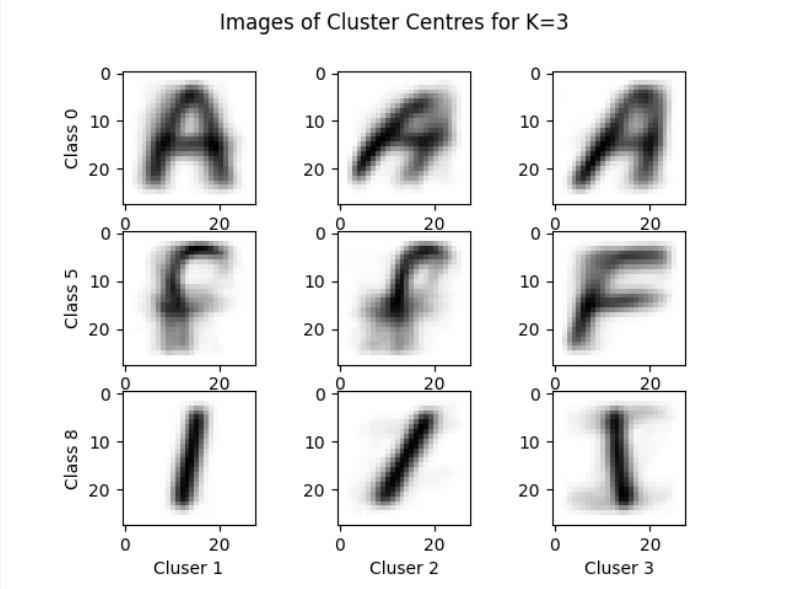
Question 2 continued on next page...

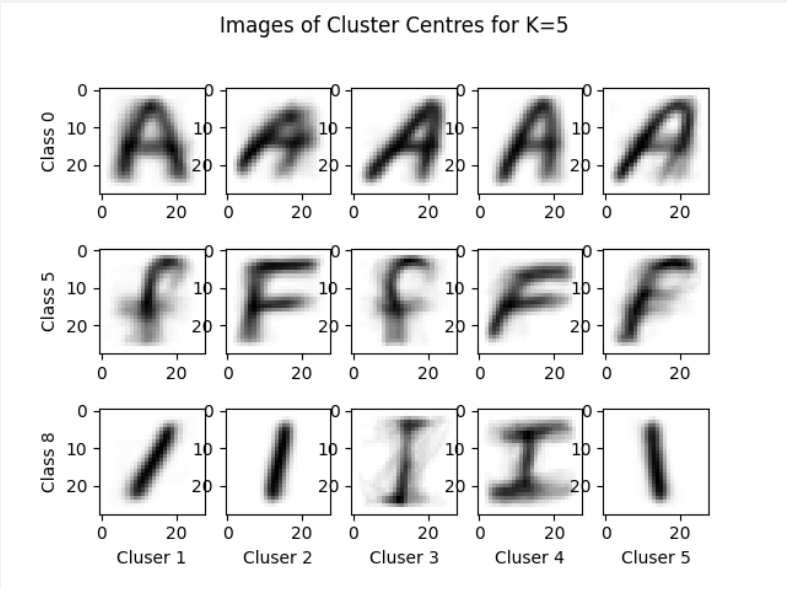
2.3 (7 points) Apply k-means clustering to the instances of each of class $0, 5, 8$ (i.e. 'A', 'F', 'I') in `Xtrn` with $k = 3, 5$, for which use Sklearn's `KMeans` with `n_clusters=`k and `random_state=0` while using default values for the other parameters. Note that you should apply the clustering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





(b) We can find similarities in the images using the same cluster centres where the hand writings in cluster 2 tends to be tilted compared to the other clusters when $k = 3$. This was also reflected in $k - 5$ where the hand writings in cluster 4 are more tilted towards the right.

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) The sum of square error represents the total sum of distances between the observation and clusters. k-means algorithm start with a random initial k clusters and assign data points to each clusters and evaluating the scree plot, we can see that as the value of k increases, the aggregate distance decreases. Therefore, the sum of square error does not increase but decrease as the value of k increases

(b) Every iteration, there is exist a point that was represented better than the previous iteration. In other words, the point is closer to the cluster centroid. Therefore, the sum of squared error will not increase but decrease. Eventually, we will stop the k-means algorithm when the decrease the in SSE is little and reach its local minimum.

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 of the training set and testing sets are 0.9162 and 0.7223 respectively.

(b) The table is shown below,

class number	alphabet	numbers of misclassification
11	L	53
17	R	48
8	I	42
10	K	38
13	N	36

(c) Hand writings differ greatly by individuals. We believe that the misclassifications occur greatly between the alphabets that have similar shape or stroke. For instance, `L` and `I` has a single vertical stroke and horizontal stroke at the bottom as common strokes. Thus, the logistic regression classification will have high chance to misclassify between these alphabets. Similar argument applies between`R` and `K` where they have similar stroke on 3 quarter of the letter. Lastly `N` can be confused with `H` as they have two vertical strokes on each end and a diagonal for `N` and horizontal for `H` respectively.

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) The possible approaches are

- standardising the training set
- optimising the hyperparameters (penalty type, regularisation penalty, solver) by performing on the validation set.
- performing cross validation.

As the data set we are using is huge, implementation of cross validation will take longer running time thus not efficient with the given computing power. Hence, we will perform hyperparameter tuning on the validation set to increase the accuracy of the classification model.

(b)
- Using `GridSearchCV` from the `sklearn` pacakge, we will perform parameter tuning on penalty type ($l1$-norm, $l2$-norm, none), and regularisation penalty with `[1e-3, 0.005, 1e-2, 0.05, 1e-1,0.5, 1, 5, 10, 50, 100]`.
- In order to compare the accuracy result with the previous values in 2.5, we used the same setup where `max_iter=1` and `random_state=0`.
- As we want to use $l1$-norm penalty and have faster convergence, we employed `[lbfgs, Newton-cg, liblinear]` as solvers.
- `GridSearchCV` it iterates all possible combination of the parameters and calculates the accuracy on the validation set and optimise the outcome.
- The process of optimisation ends by comparing the accuracy on the validation set of each choice of parameters and pick the parameters that result in the highest mean accuracy.

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

(c) The obtained results are shown in the table below,

C	Penalty	Solver	Accuracy	C	Penalty	Solver	Accuracy
0.001	l1	liblinear	0.03846	0.5	l1	liblinear	0.7377
	l2	newton-cg	0.6541		l2	newton-cg	0.7346
		lbfgs	0.6541			lbfgs	0.7346
		liblinear	0.6446			liblinear	0.7219
0.005	l1	liblinear	0.03846	1.0	l1	liblinear	0.7274
	l2	newton-cg	0.7091		l2	newton-cg	0.7217
		lbfgs	0.7091			lbfgs	0.7217
		liblinear	0.6776			liblinear	0.7121
0.01	l1	liblinear	0.150	5.0	l1	liblinear	0.6612
	l2	newton-cg	0.7279		l2	newton-cg	0.6869
		lbfgs	0.7279			lbfgs	0.6869
		liblinear	0.6931			liblinear	0.6773
0.05	l1	liblinear	0.645	10.0	l1	liblinear	0.6277
	l2	newton-cg	0.7519		l2	newton-cg	0.6769
		lbfgs	0.7519			lbfgs	0.6769
		liblinear	0.7222			liblinear	0.6567
0.1	l1	liblinear	0.6977	50	l1	liblinear	0.6635
	l2	newton-cg	0.7536		l2	newton-cg	0.6635
		lbfgs	0.7536			lbfgs	0.6633
		liblinear	0.7247			liblinear	0.6119
none	none	newton-cg	0.6201	100	l1	liblinear	0.5762
none	none	lbfgs	0.6379		l2	newton-cg	0.659
						lbfgs	0.6588
						liblinear	0.599

The best choice of parameters is {'C': 0.1, 'penalty': 'l2', 'solver': 'newton-cg'} with,

	Validation Set	Training Set	Testing Set
Accuracy	0.7536	0.8449	0.7473

Comparing to the classification accuracy in 2.5, we observe that the training accuracy decreased whereas the testing accuracy increased. Since value of C determines the weight on the training set in the model, we believe that choosing $C = 0.1$ decreased the training accuracy but increased the testing accuracy and preventing overfitting of the model on the training set.

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.

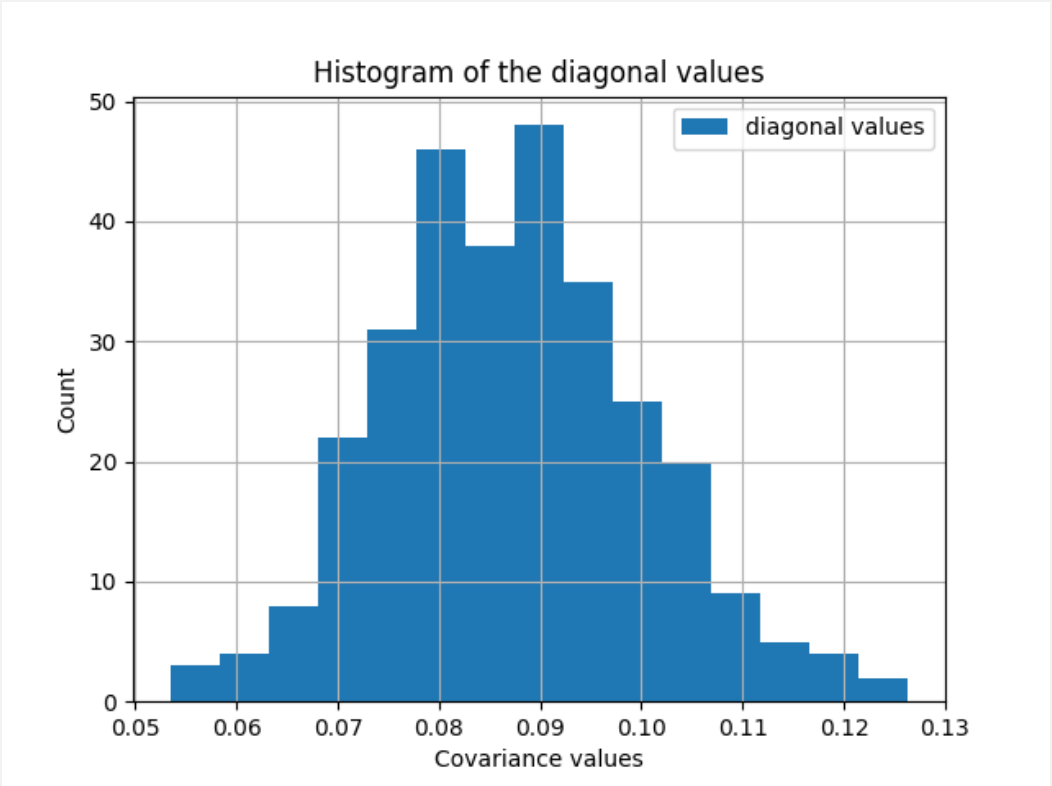
(a) The table is shown below,

Minimum	Maximum	Mean
-0.0341	0.126	0.0171

(b) The table is shown below,

Minimum	Maximum	Mean
0.0534	0.126	0.0875

(c) The plot is shown below,



(*continued from the previous page for Q*)

(d) The error message is shown below,

```
ValueError: Dimension mismatch: array 'cov' is of shape (300, 300),
but 'mean' is a vector of length 784
```

The error occurs due to dimension mismatch where the number of training samples is not as big as D and this leads Σ to at least have $D = 784$.

(e)
- We can consider reshaping the dataset so that we can allow the covariance matrix to be invertible and remain symmetric.
- We can consider using the EM algorithm technique to iteratively solve.
- we could choose different distribution by measuring the goodness. For instance how likely the distribution is to have the generated samples. Thus, we can calculate the maximum likelihood estimator. Thus, we could choose other than normal distribution.

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 value is -1.713×10^6. This was possible as we did not use the covariance matrix directly.

(b)

Class	Training Accuracy	No. Correct	Testing Accuracy	No. Correct
class 0	0.03846	300	0.03846	100
class 1	0.03846	300	0.03846	100
class 2	0.03846	300	0.03846	100
class 3	0.03846	300	0.03846	100
class 4	0.03846	300	0.03846	100
class 5	0.03846	300	0.03846	100
class 6	0.03846	300	0.03846	100
class 7	0.03846	300	0.03846	100
class 8	0.03846	300	0.03846	100
class 9	0.03846	300	0.03846	100
class 10	0.03846	300	0.03846	100
class 11	0.03846	300	0.03846	100
class 12	0.03846	300	0.03846	100
class 13	0.03846	300	0.03846	100
class 14	0.03846	300	0.03846	100
class 15	0.03846	300	0.03846	100
class 16	0.03846	300	0.03846	100
class 17	0.03846	300	0.03846	100
class 18	0.03846	300	0.03846	100
class 19	0.03846	300	0.03846	100
class 20	0.03846	300	0.03846	100
class 21	0.03846	300	0.03846	100
class 22	0.03846	300	0.03846	100
class 23	0.03846	300	0.03846	100
class 24	0.03846	300	0.03846	100
class 25	0.03846	300	0.03846	100

(c) The training accuracy is the same for all classes and also for the testing accuracy as we trained each model using a single class.

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) In order to apply the maximum likelihood estimation, we need to employ it on the joint distribution of the Gaussian mixture model. However, the computation cost of calculating the maximum likelihood estimators is expensive thus taking the EM algorithm or k-means cluster are suggested before applying MlE.

(b) First, we randomly select mixtures and their likelihoods. The Expectation step computes the expected class of the data points for each class. In the maximisation step, we compute the maximum out of the function. Thus estimating latent variables and then estimating the parameters given the latent variable estimates. EM algorithm taks soft assignment which computes the probability of a point belonging to a certain cluster. In k-means algorithm, it assumes each data point to belong to only one cluster using the euclidean distance computed between a point and a centroid of a cluster. Since we are employing hard assignment, we can apply the maximum likelihood estimation easily as the likelihood is either value of 0 or 1 and thus we can estimate the parameters of GMMs.