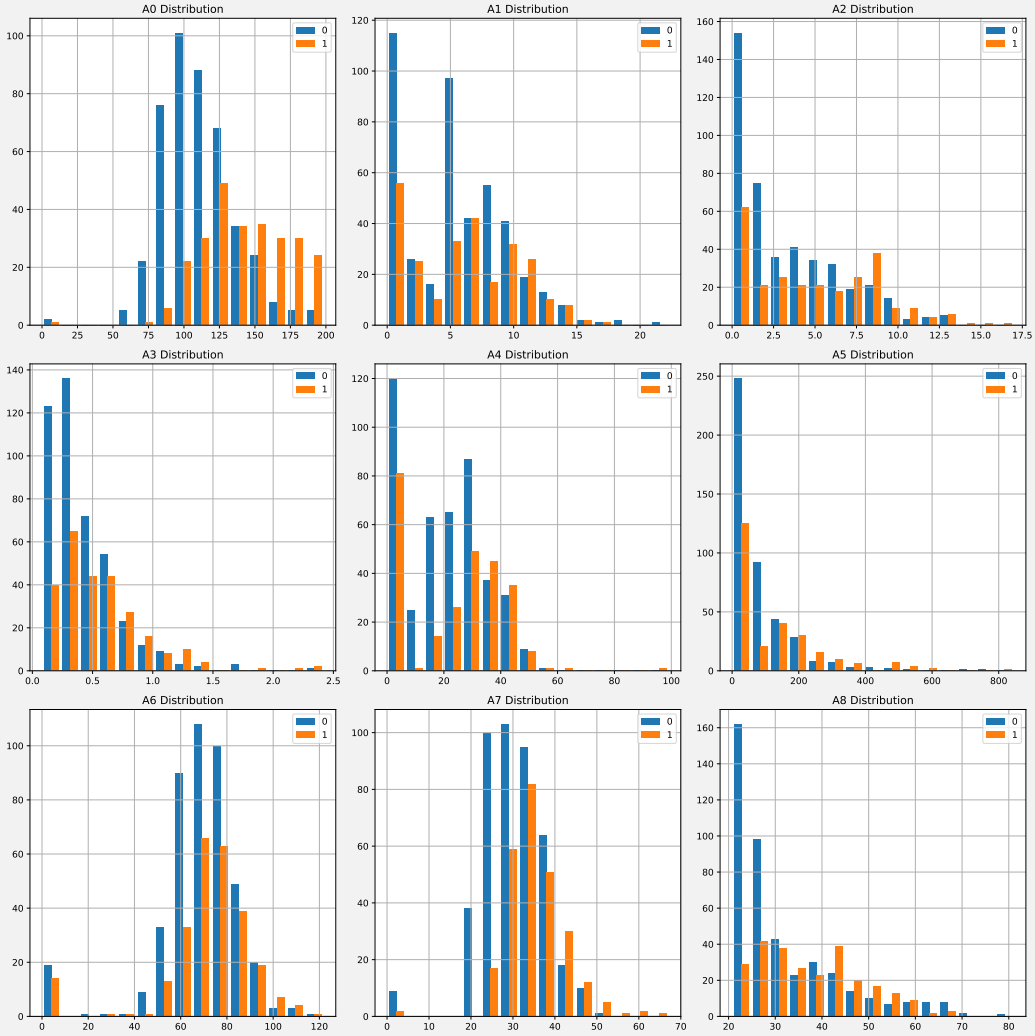
Question 1 : (80 total points) Experiments on a binary-classification data set

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.



In the above figure each plot represents a histogram of individual attribute of the dataset. We can see that all the attributes do not have a same scale (i.e) they have values in different ranges. We can also observe that the samples of class 0 are more in number compared to samples of class 1 for the whole dataset and also for individual attribute this shows that the dataset is not balanced.
The histograms for attributes A1,A2,A3,A4,A5,A8 are right skewed whereas for the attributes A0,A6,A7 are symmetrical. This information is useful to know the spread of the data for each attribute.
In all of the above histograms we observe some outliers for each attribute, this is very important to know before using some ML algorithms on the dataset.

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 table shows the correlation coefficient of each attribute in `Xtrn` with respect to class label `Ytrn`.

Attribute	A0	A1	A2	A3	A4	A5	A6	A7	A8
Corr_coef	0.4769	0.0808	0.2396	0.1825	0.1119	0.1775	0.0800	0.2969	0.2564

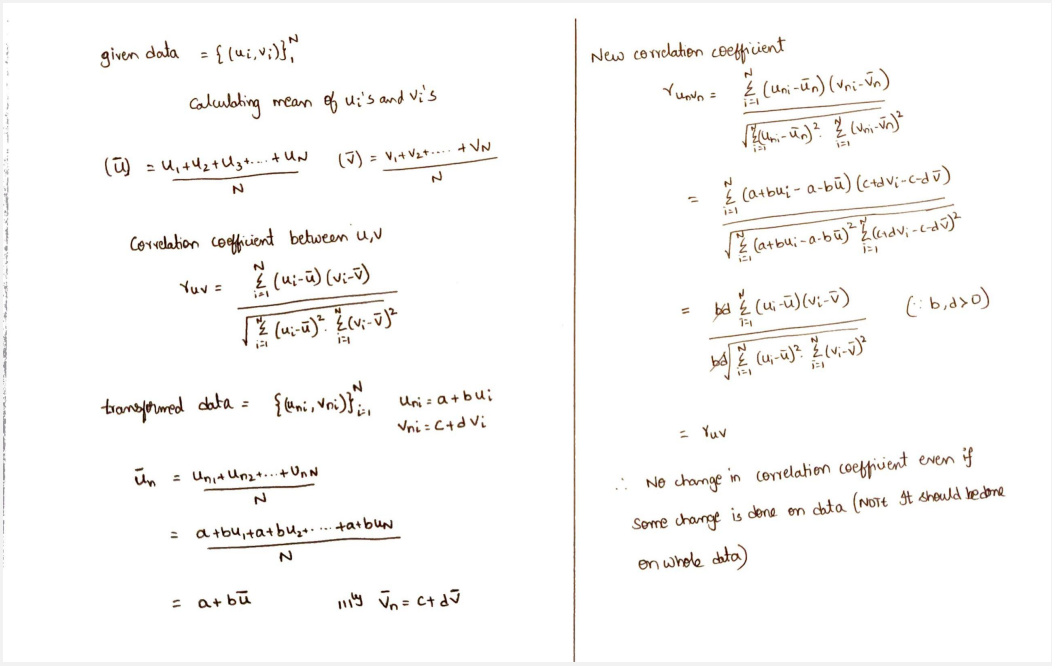
The formula used in calculating correlation coefficients was taken from the lecture slides.

(b) It is a good idea to use variables having high correlation with class label for classification tasks because they might help us to gain some common inference from the data which helps us in further analysis for creating models.

(c) It is generally not a good idea to ignore the attributes that has small correlation with the label because sometimes they might be indirectly related (i.e) they might have high correlation with other attribute which may in turn have high correlation with the class label. If we want to reduce the dimensions then it is better to use algorithms such as PCA.

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 $b > 0, d > 0$.



given data $= \{(u_i, v_i)\}_1^N$

Calculating mean of u_i's and v_i's

$(\bar{u}) = \frac{u_1+u_2+u_3+\cdots+u_N}{N}$ $(\bar{v}) = \frac{v_1+v_2+\cdots+v_N}{N}$

Correlation coefficient between u,v

$$r_{uv} = \frac{\sum_{i=1}^{N}(u_i-\bar{u})(v_i-\bar{v})}{\sqrt{\sum_{i=1}^{N}(u_i-\bar{u})^2 \cdot \sum_{i=1}^{N}(v_i-\bar{v})^2}}$$

transformed data $= \{(u_{ni}, v_{ni})\}_{i=1}^{N}$ $u_{ni} = a + bu_i$, $v_{ni} = c + dv_i$

$$\bar{u}_n = \frac{u_{n1}+u_{n2}+\cdots+u_{nN}}{N}$$

$$= \frac{a+bu_1+a+bu_2+\cdots+a+bu_N}{N}$$

$$= a + b\bar{u} \quad \text{similarly } \bar{v}_n = c + d\bar{v}$$

New correlation coefficient

$$r_{u_n v_n} = \frac{\sum_{i=1}^{N}(u_{ni}-\bar{u}_n)(v_{ni}-\bar{v}_n)}{\sqrt{\sum_{i=1}^{N}(u_{ni}-\bar{u}_n)^2 \cdot \sum_{i=1}^{N}(v_{ni}-\bar{v}_n)^2}}$$

$$= \frac{\sum_{i=1}^{N}(a+bu_i-a-b\bar{u})(c+dv_i-c-d\bar{v})}{\sqrt{\sum_{i=1}^{N}(a+bu_i-a-b\bar{u})^2 \sum_{i=1}^{N}(c+dv_i-c-d\bar{v})^2}}$$

$$= \frac{bd\sum_{i=1}^{N}(u_i-\bar{u})(v_i-\bar{v})}{bd\sqrt{\sum_{i=1}^{N}(u_i-\bar{u})^2 \cdot \sum_{i=1}^{N}(v_i-\bar{v})^2}} \quad (\because b, d > 0)$$

$$= r_{uv}$$

∴ No change in correlation coefficient even if some change is done on data (NOTE it should be done on whole data)

Hence we can prove that linear transformation of data does not affect the correlation coefficient.

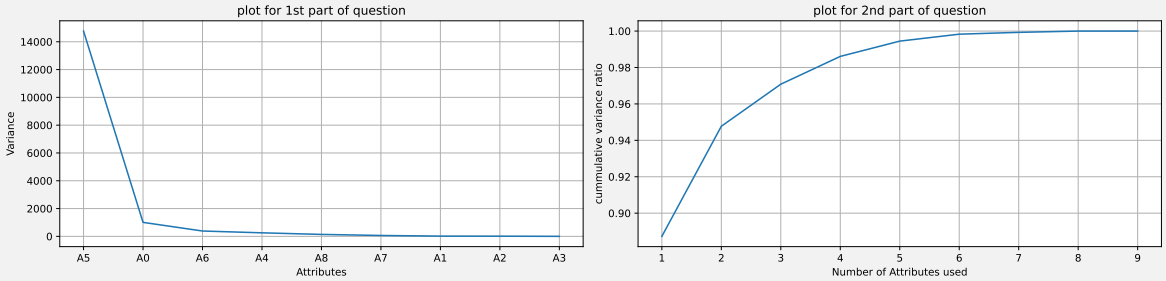
Question 1 continued on next page...

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) Sum of all variances is equal to 16645.6366

(b) The below figure represents plot for each part of question and it is assumed that in second part of question for calculating cummulative variance ratio we select the attributes in descending order of their variance.



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.

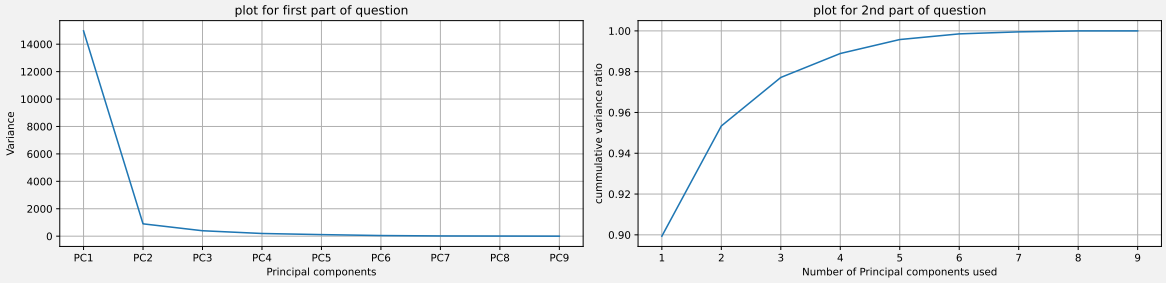
- A graph of the amount of variance explained by each of the principal components.
- A graph of the cumulative variance ratio, where the range of y-axis should be [0, 1].

(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 principal components is equal to 16645.6366

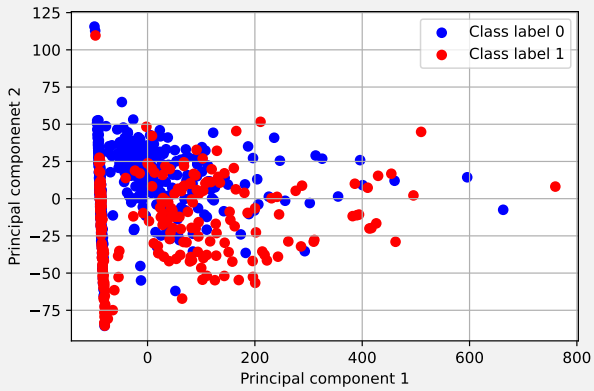
(b) The figure below represents the plots for two parts of the question.



For the first part of the question we have the principal components represented on x axis and the variance on y axis and according to the question it is in decreasing order of variance.

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

(c) The figure on the left represents the plot described according to the question. The image on the right shows the code to obtain the figure using the eigen values(NOTE-formula taken from lecture slides "Week 8 PCA - low dimensional projection of data").




```
In [17]:
eigen_vectors = pca.components_[0:2]
#print(eigen_vectors[0].shape)
means_Xtrn=[]
for i in range(0,9):
    mean = Xtrn[:,i].mean()
    means_Xtrn.append(Xtrn[:,i]-mean)
X_1 = np.array(means_Xtrn).transpose()
val = np.dot(X_1,eigen_vectors.transpose())

In [19]:
print(val.shape)
print(val)
Xa1,Xb1=[],[]
zero_y = np.where(Ytrn == 0)[0]
one_y = np.where(Ytrn ==1)[0]
for i in zero_y:
    Xa1.append(val[i])
for j in one_y:
    Xb1.append(val[j])
Xa1,Xb1= np.array(Xa1),np.array(Xb1)

(700, 2)
[[-88.48961299  -0.82651381]
 [196.34286052 -52.58081499]
 [ 93.99775534   3.95616421]
 ...
 [413.44775913 -20.03871517]
 [-87.67238222  -4.75176836]
 [ 17.66472726  19.25686184]]
```

(d) The table represents the correlation between the attributes(A0-A8) and projection of training samples on principal components PC1 , PC2.

Attribute	Principal component 1	Principal component 2
A0	0.3856	**-0.9142**
A1	-0.0458	-0.0908
A2	-0.0571	-0.2255
A3	0.1858	-0.0798
A4	0.4592	0.0972
A5	**0.9996**	0.0241
A6	0.1006	-0.2554
A7	0.2323	-0.1726
A8	-0.0016	-0.3734

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.

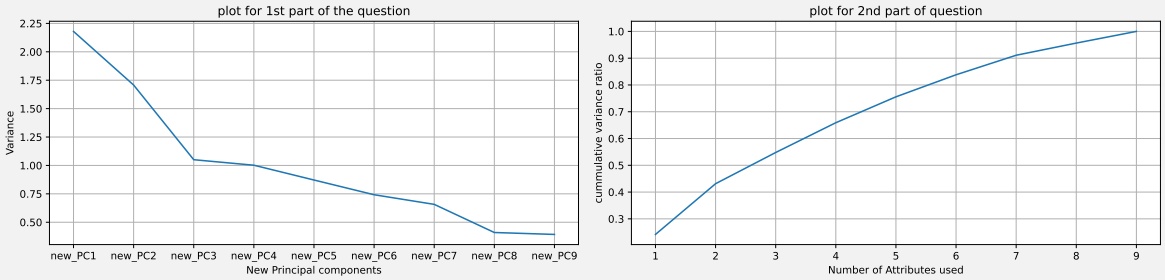
Create the standardised training data `Xtrn_s` and test data `Xtst_s` in your code in the following manner.

```
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 as explained by principal components after standardizing the data is equal to 9.01287553

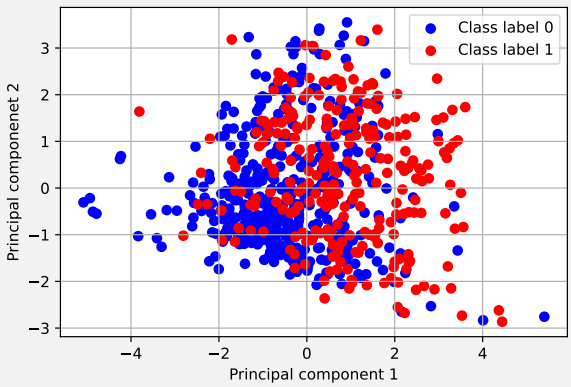
(b) The figure below represents the plots for two parts of the question.



In the first part of the question use newly created principal components after standardization of training/test data on x axis and the newly calculated variance on y axis.

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

(c) The figure on the left represents the plot described according to the question. The image on the right shows the code to obtain the figure using the eigen values(NOTE-formula taken from lecture slides "Week 8 PCA - low dimensional projection of data").




```
In [15]:
eigen_vectors1 = pcaq6.components_[0:2]
#print(eigen_vectors[0].shape)
means_Xtrn1=[]
for i in range(0,9):
    mean1 = Xtrn_s[:,i].mean()
    means_Xtrn1.append(Xtrn_s[:,i]-mean1)
X_11 = np.array(means_Xtrn1).transpose()
val1 = np.dot(X_11,eigen_vectors1.transpose())

In [16]:
print(val1.shape)
print(val1)
Xa11,Xb11=[],[]
zero_y = np.where(Ytrn == 0)[0]
one_y = np.where(Ytrn ==1)[0]
for i in zero_y:
    Xa11.append(val1[i])
for j in one_y:
    Xb11.append(val1[j])
Xa11,Xb11= np.array(Xa11),np.array(Xb11)

(700, 2)
[[-1.47111815e+00  1.12034934e+00]
 [ 2.10332737e+00 -1.51716117e+00]
 [-3.77942531e-01 -8.34907680e-01]
 ...
 [ 3.34667666e+00  9.64860656e-01]
 [-1.92796467e-04 -7.93335345e-02]
 [ 9.10685592e-01  1.64326590e+00]]
```

(d) The table represents the correlation between the attributes(A0-A8) and PC1 , PC2.

Attribute	Principal component 1	Principal component 2
A0	0.6007	0.1774
A1	0.0572	0.1000
A2	0.2679	0.7599
A3	0.3657	-0.2076
A4	0.6230	-0.4659
A5	0.6299	-0.3698
A6	0.5229	0.2242
A7	**0.6512**	-0.1684
A8	0.3529	**0.7812**

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) In questions 1.4 and 1.5 we have the same total amount of unbiased sample variance and this is because in Q1.5 we are not dropping any data while applying PCA and therefore the total amount of variance by principal components will be equal to that of total variance given by attributes. This is also the same reason why the graph looks identical for variance values and cumulative variance ratio in both the questions.

(b) In case of question 1.5 and 1.6 we have differenct values for total unbiased sample variance because in case of Q1.6 we standardize the data and hence all the attributes are made to same scale. We can also observe the difference when we compare the plots of part c of both questions we find that question 1.6 data points are centered on mean
We can also see the clear difference in correlations before and after standardization of data, this has a huge impact on the model we create for further analysis. The values in bold represent the attributes that have high correlation with one of the principal component we can observe that attributes A5 and A0 had high correlation with PC1,PC2 respectively whereas after standardization attributes A7,A8 had high correlation with new PC1,PC2.

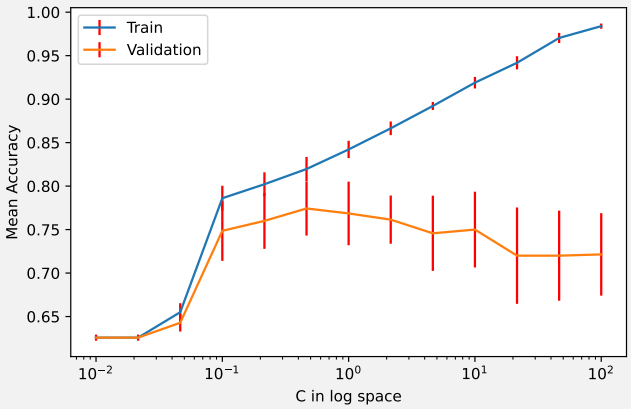
Question 1 continued on next page...

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 shows the mean accuracy of training set in cross validation , cross validation accuracy for values of C along with mean standard deviation as error bars in red.



(b) We can observe from the graph that when C value increases the mean cross validation accuracy increases and after reaching a maximum value it starts to decrease. Whereas in case of training set accuracy increases as the value of C increases showing that it is starting to overfit.

(c) The maximum mean cross validation accuracy of 0.7742857 is achieved when value of c = 0.46415888336127775

(d) Using the value of C calculated in part c of the question(best parameter) we test it on test set(`Xtst_s`, `Ytst`) and get a classification accuracy of 0.75 and the number of correctly classified instances as 75 (Since number of sample in `Xtst_s` is 100.

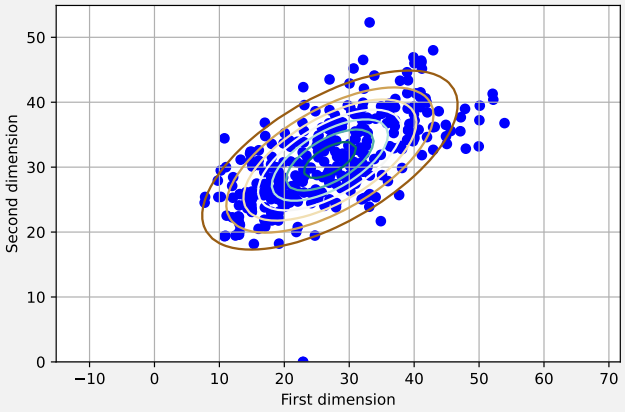
1.9 (5 points) We here consider a two-dimensional (2D) Gaussian distribution for a set of two-dimensional vectors, which we form by selecting a pair of attributes, A4 and A7, in `Xtrn` (NB: not `Xtrn_s`) whose label is 0. To make the distribution of data simpler, we ignore the instances whose A4 value 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 gaussian distribution = $[27.020943396226414, 31.093207547169815]$. The two values corresponds to mean of attributes of gaussian distribution (A4,A7). The unbiased sample covariance matrix = $[[95.14113475, 41.46999034][41.46999034, 46.69341618]]$

(b) The figure contains gaussian distribution between x and y axis shown as contours and the x axis corresponds to first dimension of `Ztrn` (A4 attribute value of all samples which are greater than 1 and class label =0) and y axis corresponds to second dimension of `Ztrn` (A7 attribute value of all samples with class label 0).

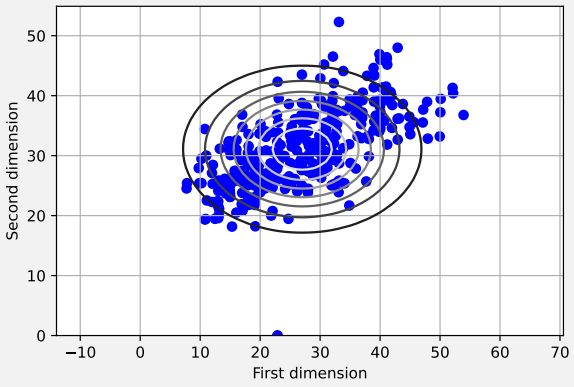


1.10 (7 points) Assuming naive-Bayes, estimate the model parameters of a 2D Gaussian distribution for the data `Ztrn` you created in 1.9, and answer the following questions.

(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 gaussian distribution = $[27.020943396226414, 31.093207547169815]$. The two values corresponds to mean of attributes of gaussian distribution (A4,A7). The unbiased sample covariance matrix =
$[[95.14113475, 0][0, 46.69341618]]$ Here since we assume Naive Bayes the attributes are independent of each other (i.e) covariance between them is zero.

(b) The figure contains gaussian distribution between x and y axis shown as contours and the x axis corresponds to first dimension of `Ztrn` (A4 attribute value of all samples which are greater than 1 and class label =0) and y axis corresponds to second dimension of `Ztrn` (A7 attribute value of all samples with class label 0).



(c) When we compare results of Q1.9 and Q1.10 we find that the position of the contour does not change (i.e) because the mean vector is same in both cases but the direction of spread of contour is different due to use of different covariance matrix.
No it is not a good idea to assume Naive baye's assumption in this question because there is a slight correlation between the attributes and we cannot misinterpret the information. This is true because if we compare figures in part b of the questions we find that Q1.9 has a better fit of gaussian model to the data rather than gaussian defined in Q1.10.

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 and standard deviation of cross validation accuracy are as follows: 0.7714285714285716 , 0.04356557337707687

(b) The table below is constructed based on instructions.

Dropped Attribute	Mean cross validation accuracy
A0	**0.6914**
A1	0.7657
A2	**0.7829**
A3	0.7557
A4	0.7743
A5	0.77
A6	0.7629
A7	0.7557
A8	0.7686

When attribute A2 was dropped we got the highest mean cross validation accuracy of 0.7829.

(c) From the table we can say that the least accuracy was obtained when attribute A0 was dropped showing that it was given more importance by the model. Similarly we can say that A2 was given the least importance.

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

2.1 (5 points)

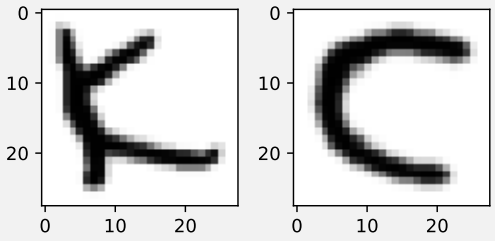
(a) Report (using a table) the minimum, maximum, mean, and standard deviation of pixel values for each `Xtrn` and `Xtst`. (Note that we mean a single value of each of min, max, etc. for each `Xtrn` and `Xtst`.)

(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) From the table we can note that the mean and standard deviation of train and test set are almost same. This is a good thing because it shows that the training and test set have the same distribution.

Train/test set	Minimum	Maximum	Mean	Standard deviation
Xtrn	0.0	1.0	0.177377	0.334982
Xtst	0.0	1.0	0.175634	0.333463

(b) The plot shows the first two instances of training data present in Xtrn. To check the class labels when we print the first two instances of Ytrn we get 10,2 representing the class label for letter k as 10 and for letter c as 2. We use pyplot.imshow() function to show the training samples as images by reshaping them into their original size(28,28) and then taking transpose of it to show the image in correct direction now we use the value of cmap = `'gray_r'` to invert the foreground and background color.



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) Yes, the euclidean distance would remain the same between same instances in `Xtrn` and `Xtrn_m` because we applied same transformation for all data in `Xtrn` to obtain `Xtrn_m` this can be proved as follows.

$x_1, x_2, x_3, \ldots, x_n$ represents training data with 'n' samples

Euclidean distance between any two samples

$$d = \sqrt{\sum_{i=1}^{k}(x_{ni} - x_{i})^2}$$ for k attributes in training data.

Now subtracting mean vector from samples we get new data as

$(x_1 - \bar{x}), (x_2 - \bar{x}), (x_3 - \bar{x}), \ldots (x_n - \bar{x})$

If we take distance (Euclidean) between two samples we get.

$$d_1 = \sqrt{\sum_{i=1}^{k}((x_n - \bar{x}) - (x_i - \bar{x}))^2} = \sqrt{\sum_{i=1}^{k}(x_n - x_i)^2} = d$$

$\therefore$ we can mathematically prove that euclidean distance does not change.

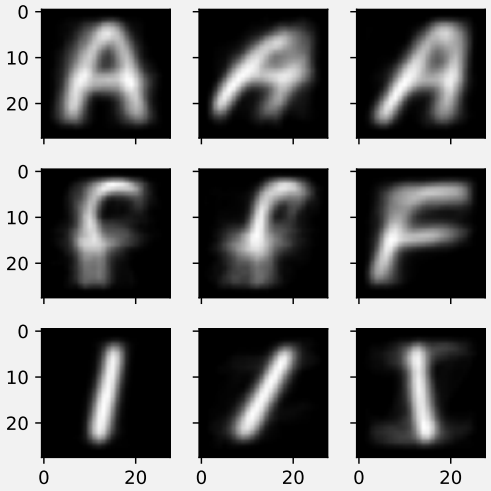
(b) No, we should not use a different mean vector for `Xtst` because we always apply same preprocessing on test data which is applied on training data this is because if we do not scale it same as training data the model might mistreat the input based on unscaled values.

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 figure on the left represents the centers of the clusters where `KMeans` algorithm is applied with $k = 3$ as parameter and figure on the right represents the centers of the clusters with $k = 5$.



(b) From the above figures we can note that there are some samples which has distorted images and they are represented as centers of the clusters. We also note that different cluster for same alphabet has different representation of the letter.
When the k value was increased from 3 to 5 we find there are similar variations and some different variations of the same letter.
There are some concerns in using this dataset for classification tasks because there are many variations of the same letter and based on the model we use for classification some of them may perform bad because of the high similarity between different letters.

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) In case of clustering with 'k' clusters let's say we have 'm' number of samples for each cluster so when calculating the mean squared error we get 'k' values now we take sum of these values according to the question. But in case of 'k+1' clusters we will have 'l' number of samples for each cluster where l<m so we get 'k+1' number of squared errors and when we take sum of these values we compare it with the first case and observe that the value does not increase.

$$\text{for 'k' clusters} = \sum_{K=1}^{n} D_k = \sum_{K=1}^{n} \sum_{j=1}^{m} (x_j - x_m)^2$$

$$\text{for 'k+1' clusters} = \sum_{K=1}^{n+1} D_k = \sum_{K=1}^{n+1} \sum_{j=1}^{l} (x_j - x_m)^2 \quad [l < m]$$

(b) If we consider during update times for 'k' number of clusters we find that the number of samples of each clusters might change based on the value of centroid but the total number of clusters remains same. In each update we try to reduce the intra class distance therefore we use this as objective function at each update so we can say that the sum of squared errors of all 'k' clusters decrease for each update step and we stop updating when we reach minimum value(i.e) centroids do not shift anymore.

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.

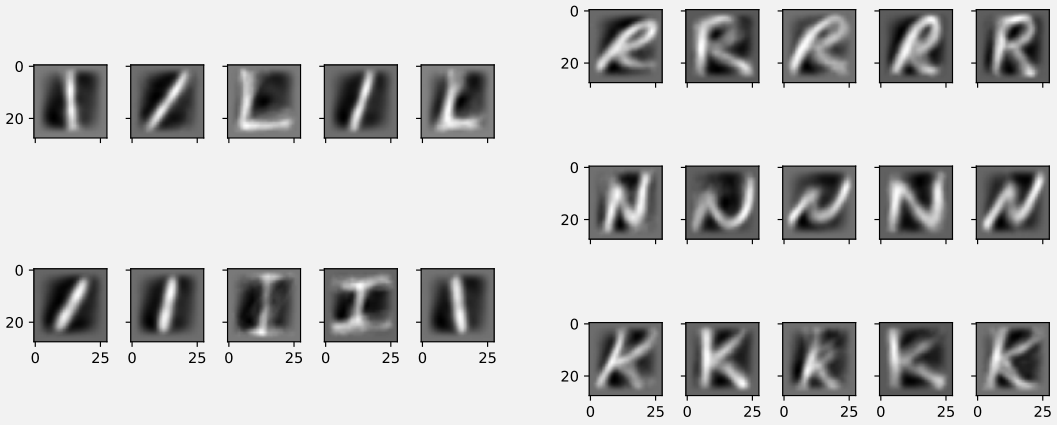
(a) Report the classification accuracy for each of the training set and test set.
(b) Find the top five classes that were misclassified most in the test set. You should provide the class numbers, corresponding alphabet letters (e.g. A,B,...), and the numbers of misclassifications.
(c) For each class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

(a) The classification accuracy of training and test set were 0.9162 , 0.72231 respectively.

(b) The table represents the letters that are mostly misclassified arranged in descending order.

Class number	Letter	number of times misclassified
11	L	53
17	R	48
8	I	42
10	K	38
13	N	36

(c) After some experimentation we find that when we apply k-means(used so that we can visualize the results better in comparision with logistic regression) with $k = 5$ on `Xtrn_m` for samples labelled as letters L,I,R,K,N and find the following figure showing their centers . We can observe that there are many samples in data that look alike after normalization is applied.



For example there are many samples which look like I but was labelled as L and in case of R and K the samples might have more similarity and since the test accuracy is not so high there is a high chance that the model misclassified these letters the most.

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 three possible solutions to increase the classification accuracy are as follows: The first possible solution is to find an optimum value of C value for the logistic regression model and check for test accuracy.
The second possible solution is to experiment with different error functions and check which one of them gave good test accuracy.
The third possible solution is to use PCA and reduce the number of dimensions and then apply logistic regression to find the test accuracy.

(b) In the experiment which was performed initially some values of C were chosen between -2 and 2 and then 8 models were trained using different C values out of which one value of C was chosen based on the test accuracy.(i.e the value of C that gave the highest test accuracy). Now we experiment using different solver functions and different errors(i.e L1,L2) it is observed that we get the highest test accuracy when we use solver as 'saga' and error as 'L1'. In the final part of the experiment we create a model with the optimum C value obtained along with the solver as 'saga' and error as 'L1'.

Question 2 continued on next page...

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

(c) First part of experimentation: Finding optimum C value.(NOTE baseline accuracy for test set = 0.7223)
Here we have chosen 8 different values for C betweeen -2 and 2 and found the following test accuracies.

Value of C	1e-2	2.2e-2	4.6e-2	1e-1	2.2e-1	4.6e-1	1	2.2
Test accuracy	0.7430	**0.7530**	0.7473	0.7442	0.7369	0.7223	0.7061	0.6865

Now for the second part of the experiment different solvers were used and experimented with L1,L2 regularization and out of them when we used 'saga' solver and 'L1' regularization we get a test accuracy of **0.74038**. For the final part of the question when experimented with both the optimal values calculated we get test accuracy of 0.64038 which is less than baseline model so we can choose one of the above two solutions which gave better results than their combinations.
Therefore in conclusion of the experiments we can choose either of the two discussed solutions and since the combination of the two optimal solutions did not result in a good solution we can ignore it. To add some final thoughts we could also use dimensional reduction on the input data and then apply logistic regression to find the test accuracy but according to the question the data should not be changed so this expeiment was not performed.

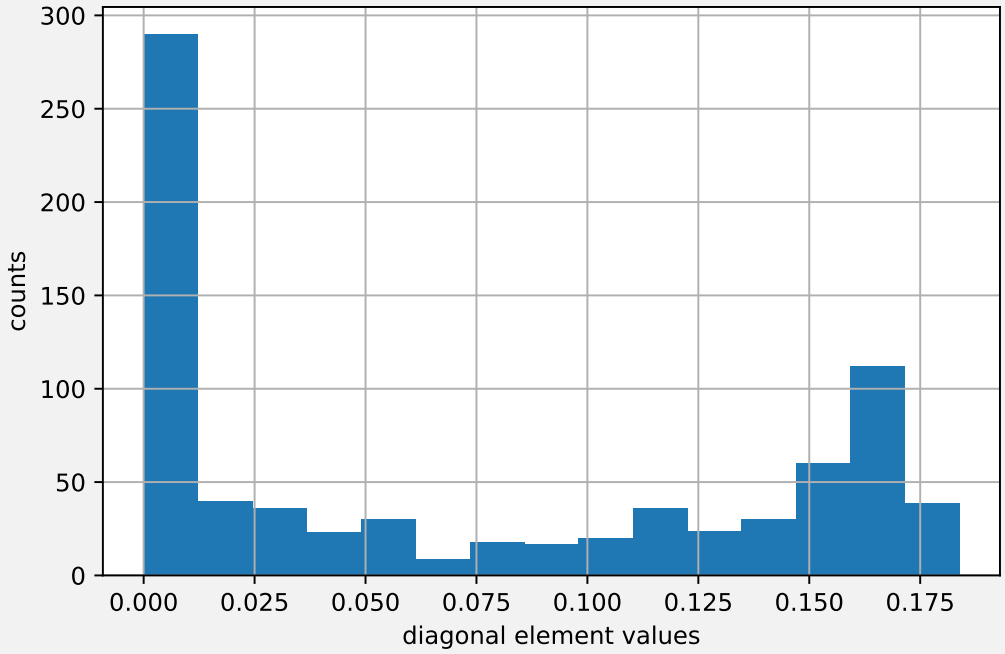
2.7 (9 points) Using the training data of class 0 ('A') from the training set `Xtrn_m`, calculate the sample mean vector, and unbiased sample covariance matrix using Numpy's functions, and answer the following.

(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 , maximum , mean values of the covariance matrix are: −0.09747401775408393 , 0.18378613679585562 , 0.001708791632836062

(b) The minimum , maximum , mean values of the diagonal elements are as follows: 0.0 , 0.18378613679585562 , 0.07231314807821394

(c) Histogram showing the diagonal values on x axis and counts on y axis.



(*continued from the previous page for Q*)

(d) When we apply the calculated mean and covariance to the scipy's `multivariate_normal` we get the following error-
`LinAlgError: singular matrix`
This error occured because during calculations we find a matrix with determinant zero which should not be the case. Scipy's multivariate normal function assigns a default threshold for all the eigen values in the covariance matrix if they are less than the threshold then it rises the above error.

(e) The first possibility is to use the arguments "allow_singular = True" to the multivariate_normal function. The second possibility would be to use a SVD(Singular Value Decomposition) algorithm to solve such cases where the matrix cannot be inverted. The third possibility would be to scale the inputs in different way.

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) According to the question model was trained and when tested on first instance of test set `Xtst_m` we get a log likelihood value of -838252.1823930496. In this case we were able to calculate the value because in GMM we use EM algorithm to iteratively search for the parameters and during the M step suitable choices are made such that the singularity problem(getting a singular matrix) does not occur.

(b) When we train a gaussian model for each letter we get 26 models which are stored in a list. Now to run the classification experiment for each sample in test data log likelihood is calculated across all 26 models and the index of the gaussian model was selected from list of gaussians which gave the highest log likelihood. It is compared with the class label `Ytst` and if they are not equal it means that the sample was misclassified. We count the number of misclassified samples and we get a value of 797.
Therefore the number of correctly classified samples for test set = number of samples in `Xtst_m` - misclassified samples = $2600 - 797 = 1803$.
The classification accuracy for test set was calculated as 0.6934615384615385.
Similarly the number of correctly classified samples for train data was calculated as 7800
Train data classfication accuracy = 1.00

(c) We can observe that we get a accuracy of `100%` on training data which is to be expected because we have trained all the models with the training data `Xtrn_m`. But when we test it on `Xtst_m` we get some misclassifications. For example if a test set sample corresponding to letter L(class label 11) was taken and applied our experiment we find out that the gaussian model for letter I(class label 8) gave more loglikelihood compared to gaussian model created for letter L because of their similarity. Therefore the sample was misclassified as I rather than L.
These misclassifications happen because there are many letters that look similar to each other after transformation and therefore might give more log likelihood to other gaussian model that the one it belongs to.

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 case of Gaussian Mixture Models we cannot use MLE directly because it is not a good idea to use it as a prior probability because there may be cases where we get zero values as probabilities. If we use MLE for estimation of probability we can find some examples where the MLE is maximum and the variance is zero leading to singularity problem which we encountered in Q1.7.

(b) In the case where we use EM algorithm we iteratively try to search for the model parameters until we converge at a point and declare those as the model parameters. Here, we do soft clustering in the sense we do not assign a data point to a gaussian but take the probability that it belongs to that gaussian. Whereas when we use the second method which is suggested we do hard clustering using Kmeans to cluster the data and then use MLE to estimate the probabilities , in such cases we do not choose the parameters randomly for the GMM but take the parameters of clusters found using Kmeans as initial parameters and estimate the probabilities using the MLE.

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) The same experiment was performed as from question 2.8 but with different number of gaussian mixture components.

(i) Table showing values of k and train/test accuracies.

K	Training set accuracy	Test set accuracy
1	1.0	0.69346
2	1.0	0.70346
4	1.0	0.82038
8	1.0	0.83269

(ii) We can see from the above table that the train accuracies remained same i.e (100%) because all the models created were trained on same data. Whereas we can see that the test set accuracy increases as the number of mixture components increase this is because now for each letter we create k number of gaussians and the likelihood of the new sample is calculated for k number of gaussians for each letter(NOTE we have 26 gaussian mixture models and each mixture model contains k number of gaussians).

(b) Now we experiment with k =2 and various values of reg_covar.

(i) The following results were obtained and the value in bold indicate the value of reg_covar value which yielded the highest test set accuracy..

reg_covar	Test set accuracy
1e-6	0.70346
1e-4	0.72923
1e-2	0.85730
1e-1	**0.87153**
1	0.7961

(ii) The reg_covar parameter is a regularization term added to the diagnol matrix of the covariance matrix so that the values stay positive on diagonal matrix. For this question by adding regularization we prevent overfitting and have improved test results.