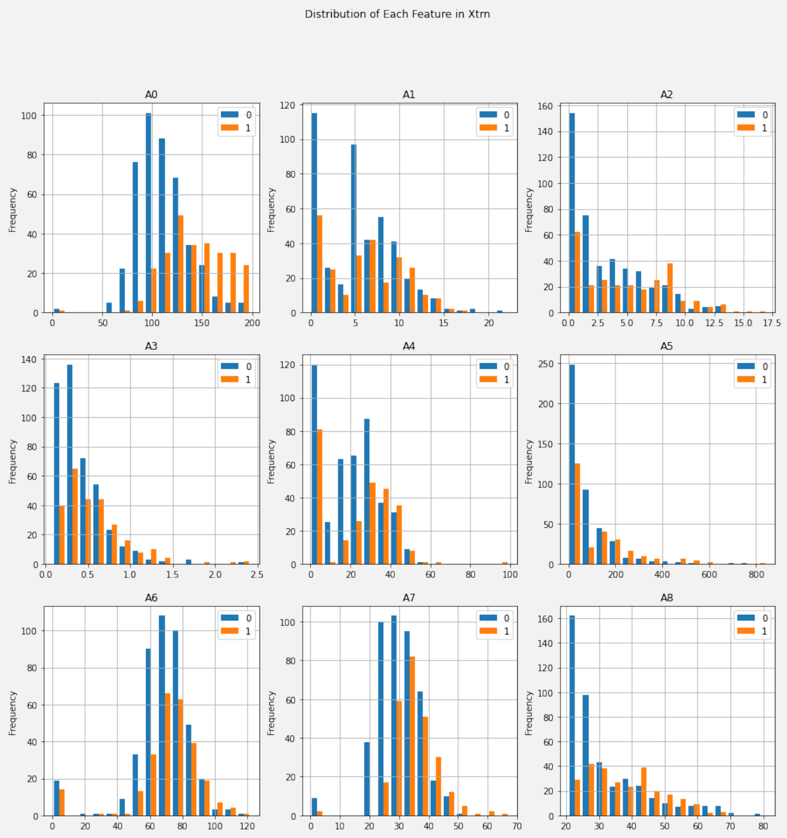
Question 1 : (70 total points) Experiments on a binary-classification data set

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.

Figure below showing the frequency of each attribute in the data set:



From the plots above it is clear that most attributes have more instances with a value of 0 than 1. Most attributes are skewed left with the exception of A0, A7 and A8. For many of the attributes the instances of 0 and 1 follow the same correlation of frequencies apart from the attribute A0 which shows a trend of the higher the value the higher the frequency of type 1 instances.

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	Correlation Coefficient
A0	0.4912
A1	0.0874
A2	0.2273
A3	0.20734
A4	0.1077
A5	0.1857
A6	0.0763
A7	0.3044
A8	0.2403

(b) It is a good idea to used the attributes with large correlations as the values it holds can be used to discern between a Type 0 case or a Type 1 case easier.

(c) Ignoring attributes is only a good idea when reducing the amount of computations is necessary. None of the attributes correlate to approximately zero, therefore all attributes should have an influence on the learning process. Moreover if attributes are indeed irrelevant the model should be able to learn to ignore any irrelevant attributes.

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 is invariant for positive linear transformations because the variables $u_i v_i$ are standardised in the process of computing the coefficient. That is, in the calculation the mean of each column is subtracted from each of the variables and they are divided by the standard deviation for each column. Therefore any positive linear transformation on the variables arithmetic or product, will have no effect on the correlation coefficient as it will essentially 'cancel out'.

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.

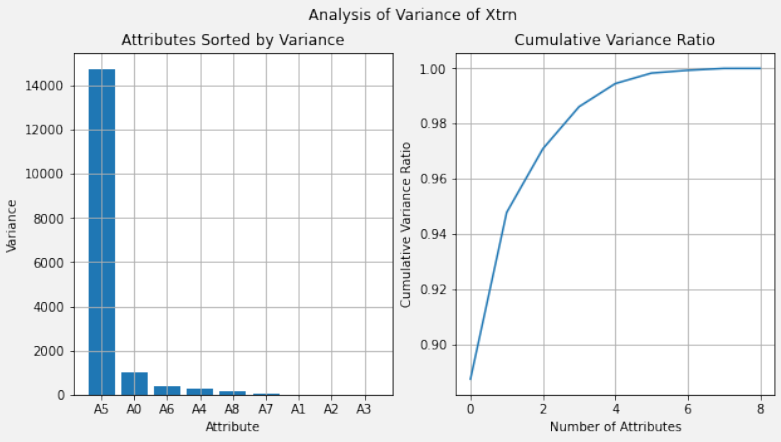
(a) Report the sum of all the variances.

(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 - 16621.8571

(b)



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.

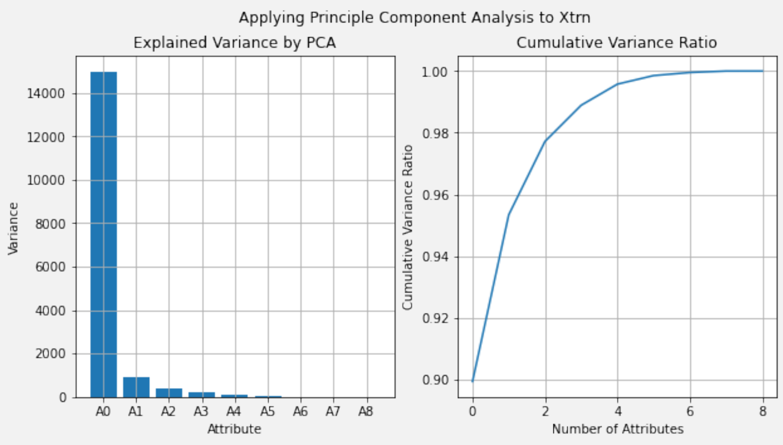
- A graph of the amount of variance explained by each of the principal components.
- A graph of the cumulative variance ratio, where the range of y-axis should be [0, 1].

(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.

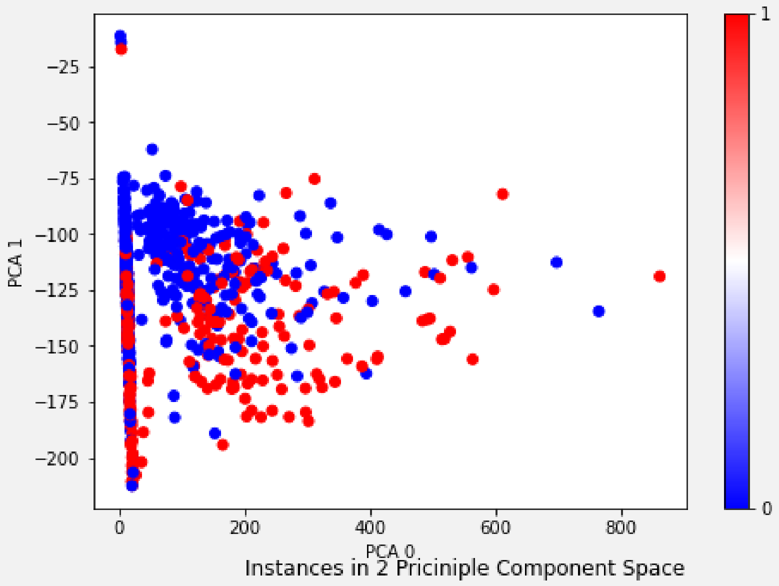
(a) Total amount of unbiased sample variance explained by the whole set of principal components = 16645.6366

(b)



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

(c)



(d)

Attribute	Correlation Coefficient
PCA 0	0.1972
PCA 1	-0.4529

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.

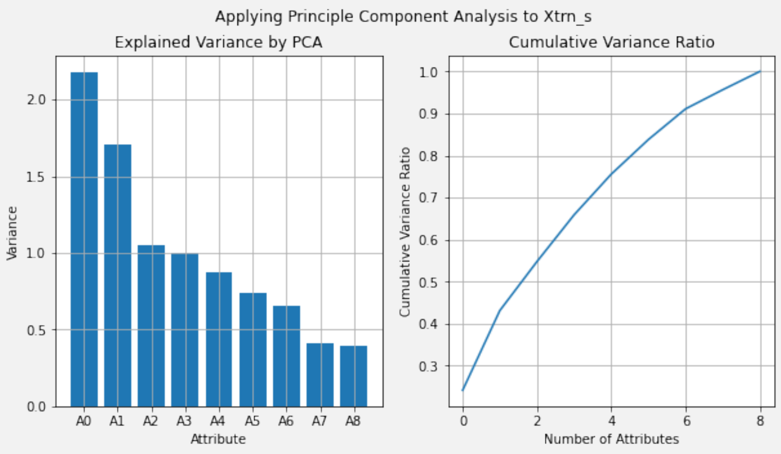
Create the standardised training data `Xtrn_s` and test data `Xtst_s` in your code in the following manner.

```
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.

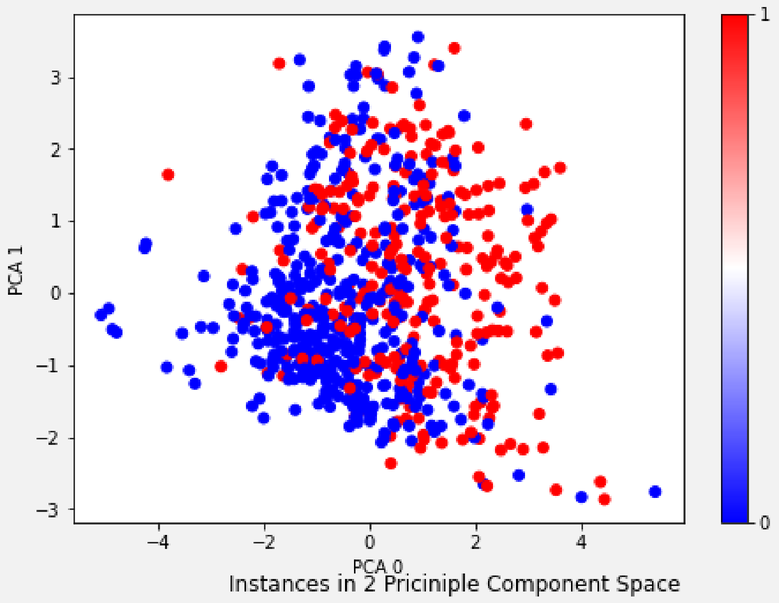
(a) Total amount of unbiased sample variance explained by the whole set of principal components = 9.0129

(b)



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

(c)



(d)

Attribute	Correlation Coefficient
PCA 0	0.4336
PCA 1	0.1527

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 question 1.4 and 1.5 we calculate the variance for each attribute in Xtrn using different methods. First was using Numpy methods and the second was using sklearn decomposition PCA. The sum of all the variance was higher for 1.5 at approx 16645.64 and lower at approx 16621.86. This can be explained by the way each package calculates the variance as NP standardises the variables and sklearn does not. From the analyses we found that A5 is responsible for more than 90% of the explained variance of the data-set for both methods. In terms of dimension reduction ideally we keep at least the first 2 components as they account for more than approximately more than 95% of all the variance. Ideally we would keep 3 components such that we only loose 2% of the variance whilst reducing the amount of data by more than half.

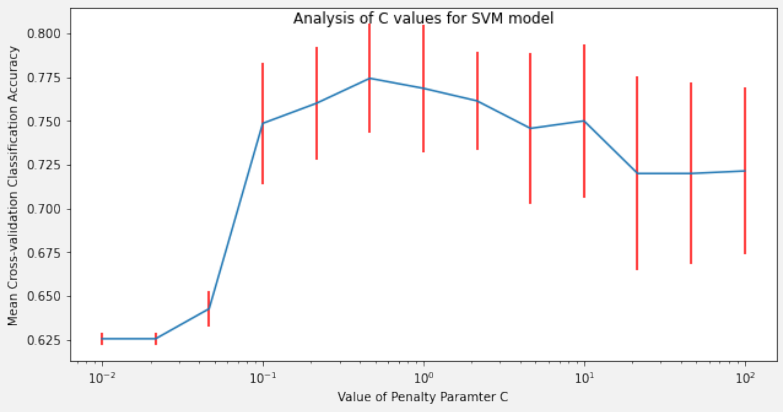
(b) From the results of 1.5 and 1.6 we can see that due to the standardisation of the attributes in Xtrn, that A5 no longer accounts for the same percentage of the total variance and that the total number of variance between Xtrn and Xtrn_s is greatly reduced. When looking at the scatter plots for 1.5 and 1.6 we see less separation between PCA 0 and PCA 1. We see the correlation coefficients show more positive correlation with projected instances. Because PCA maximises variance, PCA after standardisation will perform better than PCA without.

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)

(b) From the results produced we see that as the Value for parameter C increased the mean accuracy increased to a point and then began to decrease. Moreover from the results we see that in general as the value of C increases there is an increase in the standard deviation in the accuracy of the model.

(c) Optimal value of C is 0.4642 with a mean accuracy of 0.7743

(d) Number of correct predictions on test data 75

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`.

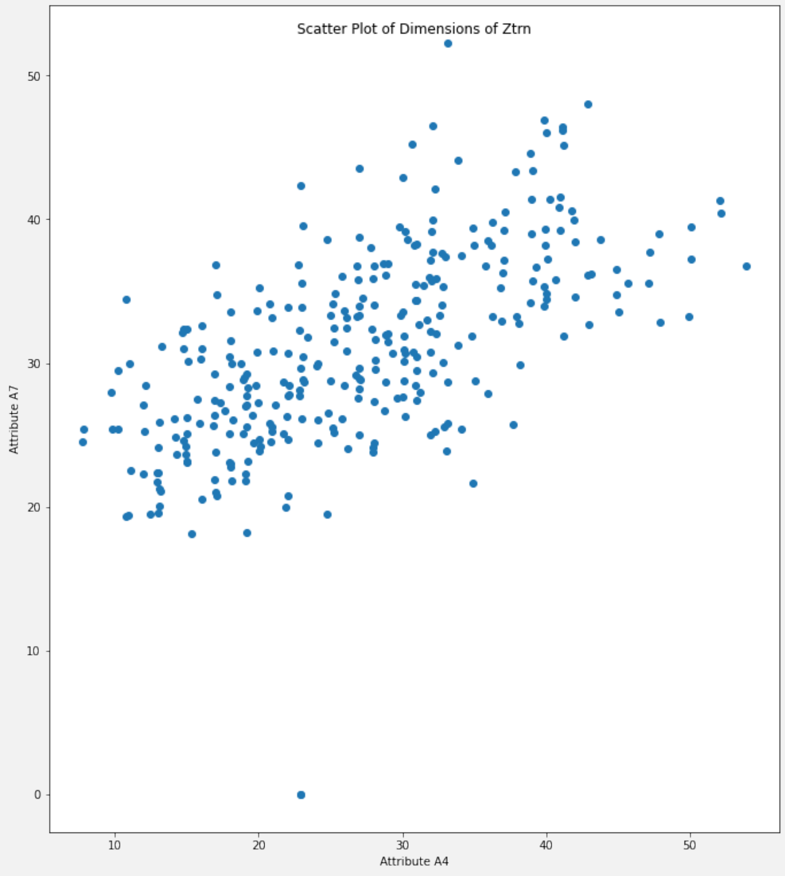
Using Numpy's libraries, estimate the sample mean vector and unbiased sample covariance matrix of a 2D Gaussian distribution for `Ztrn`. Answer the following questions.

(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) Sample mean vector $= \begin{pmatrix} 27.0209 & 31.0932 \end{pmatrix}$

Covariance Matrix $= \begin{pmatrix} 95.1411 & 41.4700 \\ 41.4670 & 46.6934 \end{pmatrix}$

(b)



(c)

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.

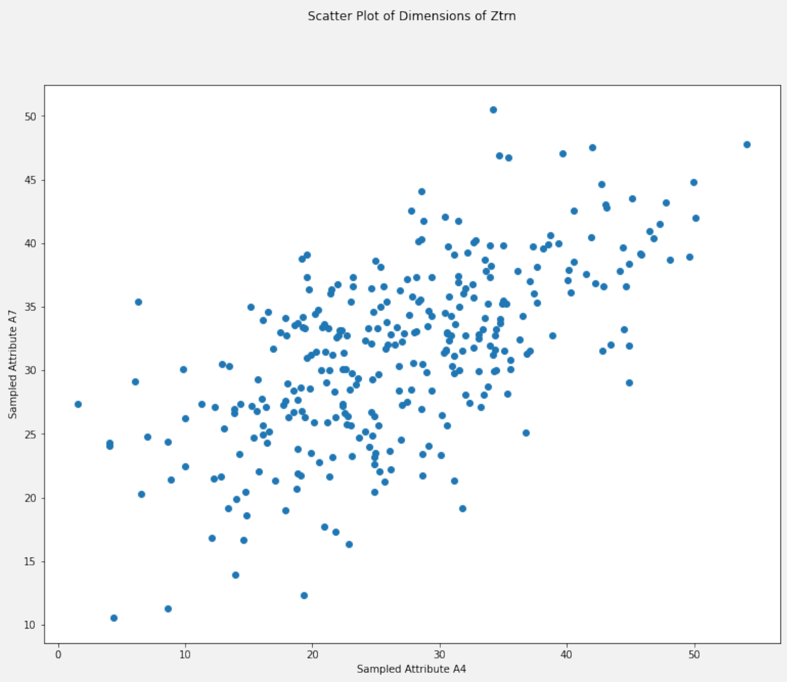
(b) Make a new scatter plot of the instances in `Ztrn` and display the contours of the estimated distribution on it. Note that you should always correspond the first dimension of `Ztrn` to x-axis and the second dimension to y-axis. Use the same scaling (i.e. equal aspect) for x-axis and y-axis, and show 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) Sample mean vector $= \begin{pmatrix} 27.3489 & 31.3275 \end{pmatrix}$

Covariance Matrix $= \begin{pmatrix} 90.0948 & 47.0641 \\ 47.0641 & 52.2508 \end{pmatrix}$

(b)



(c)

(d) The samples taken from the gaussian distribution in 1.10 follow a similar correlation to Ztrn in 1.9. However the Gaussian has more extreme values in comparison to Ztrn. Moreover the Gaussian has a lower covariance than Ztrn. In general it might be a good idea to employ naive-Bayes as it tends to help classifcation and makes it faster to predict on test sets of data. However, the naive bayes assumption assumes that all attributes contribute equally and this could be argued not to be the case as seen with previous questions and the larger values of A7 seen in 1.9.

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

2.1 (5 points)

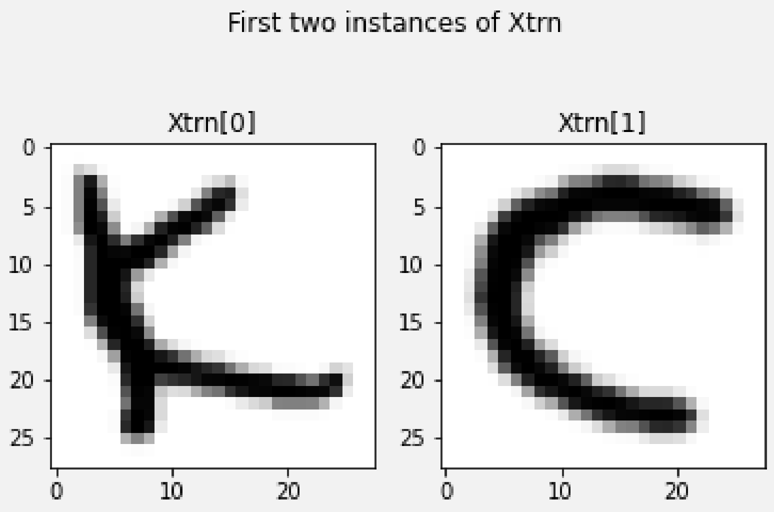
(a) Report (using a table) the minimum, maximum, mean, and standard deviation of pixel values for each `Xtrn` and `Xtst`. (Note that we mean a single value of each of min, max, etc. for each `Xtrn` and `Xtst`.)

(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)

Dataset	Minimum Value	Maxiumum Value	Mean Value	STD
Xtrn	0.0	1.0	0.1774	0.1697
Xtst	0.0	1.0	0.1756	0.3335

(b)



(c)

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) Xtrn_m should have the same euclidean distance between pairs on instances as in Xtrn. This is because all components of the instances have been adjusted by the same factor. Meaning that the straight line distance between them is consistent.

(b) The euclidean distance between pairs of instances should be the same in Xtst_m as in Xtst. This is because all components of the instances are adjusted by the same amount. Because Xtst and Xtrn are portions of a larger dataset the mean of either should be sufficient enough to represent the mean for the dataset aslong as they are sampled appropriately.

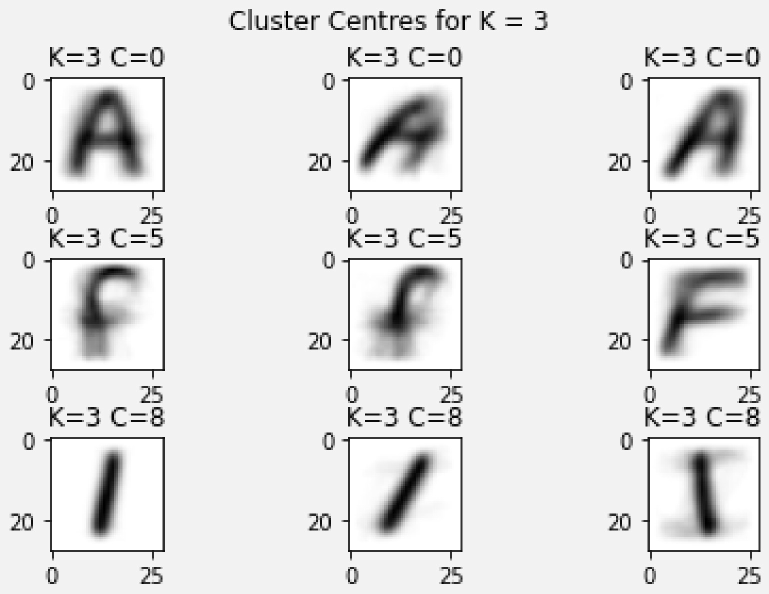
Question 2 continued on next page. . .

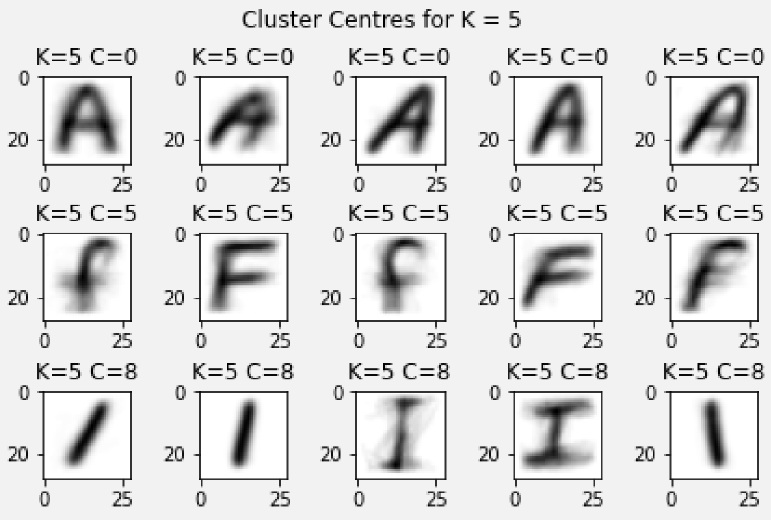
2.3 (7 points) Apply k-means clustering to the instances of each of class $0, 5, 8$ (i.e. 'A', 'F', 'I') in `Xtrn` with $k = 3, 5$, for which use Sklearn's `KMeans` with `n_clusters=`k and `random_state=0` while using default values for the other parameters. Note that you should apply the clustering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





(b) From the clustering used for the 3 classes there are some clear distinctions in the variations of letters in each class. This is noticeable for the I's or Class 8. Because of this there may be issues when it comes to classifying as some variations of classes can be very similar to other classes.

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) As k increases the sum of squared errors has to decrease because there are more clusters and therefore less room for error. This is true until k is equal to the number of data points, here each point has its own cluster and therefore the error is 0.

(b) Your Answer for (b) Here

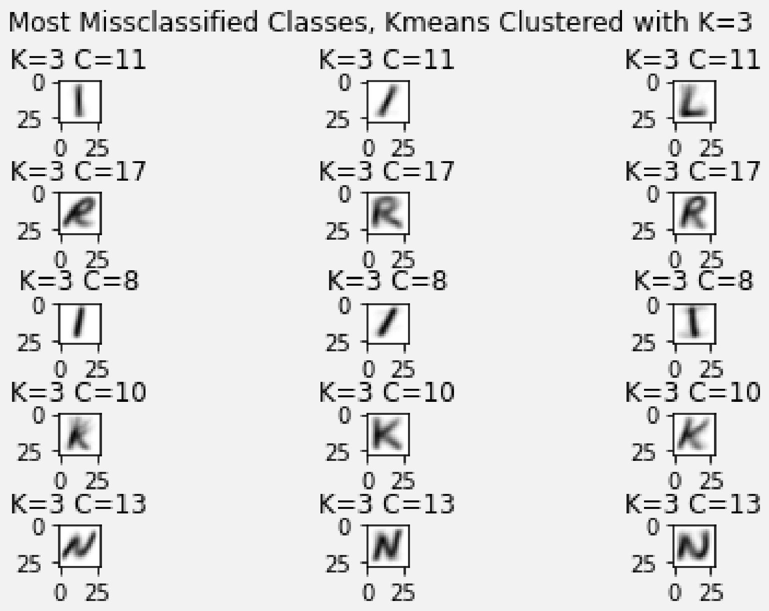
2.5 (11 points) Here we apply multi-class logistic regression classification to the data. You should use Sklearn's `LogisticRegression` with parameters '`max_iter=1000`' and '`random_state=0`' while use default values for the other parameters. Use `Xtrn_m` for training and `Xtst_m` for testing. We do 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) Classification accuracy for training set - 0.9161538461538462
Classificaiton accuracy for test set - 0.7223076923076923

(b) Top 5 missclassified classes - l,r,i,k,n
Corresponding number of missclassifications - 53, 48, 42, 38, 36

(c) For I and L it is quite obvious that these two can easily be confused with each other as they both have very similar physical features, ie, straight vertical line. The other 3 letters follow similar patterns and it seems feasible that they could easily be confused with each other.



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) Ways to improve the classification accuracy would be to tune the hyper-parameter c, normalise the data so that it has a mean of 0 and a standard deviation of 1 and we could test using different solvers to find the one with the optimum accuracy.
I would choose finding the optimum solver.

(b) For this I would use the model and compare accuracy for a range of selected solvers and choose the model that performs the best on the test data.
This is an easy implementation, changing the solver parameter for SKlearn logistic regression from the ones available to use from the package

Question 2 continued on next page. . .

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

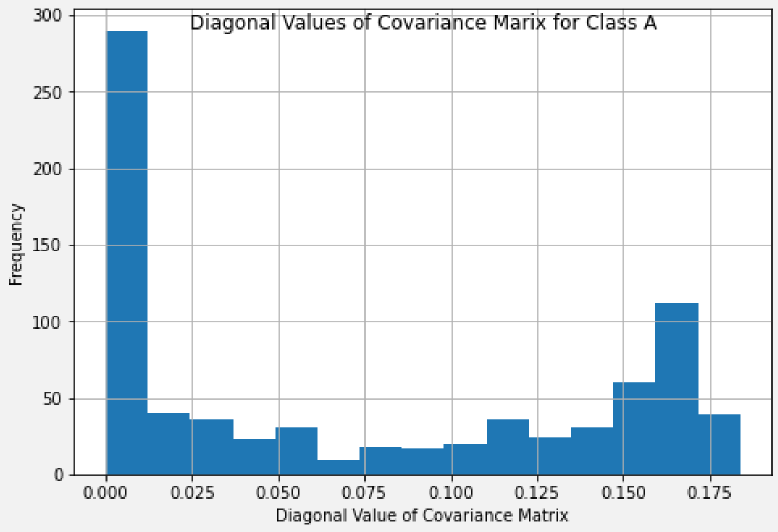
(c) Your Answer for (c) Here

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) Min - -0.0975
Max - 0.1838
Mean - 0.0017

(b) Min - 0.0000
Max - 0.1838
Mean - 0.0723

(c)


Question 2 continued on next page. . .

(*continued from the previous page for Q*)

(d) Your Answer for (d) Here

(e) Your Answer for (e) Here

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) Your Answer for (a) Here

(b) Your Answer for (b) Here

(c) Your Answer for (c) Here

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) Your Answer for (a) Here

(b) Your Answer for (b) Here