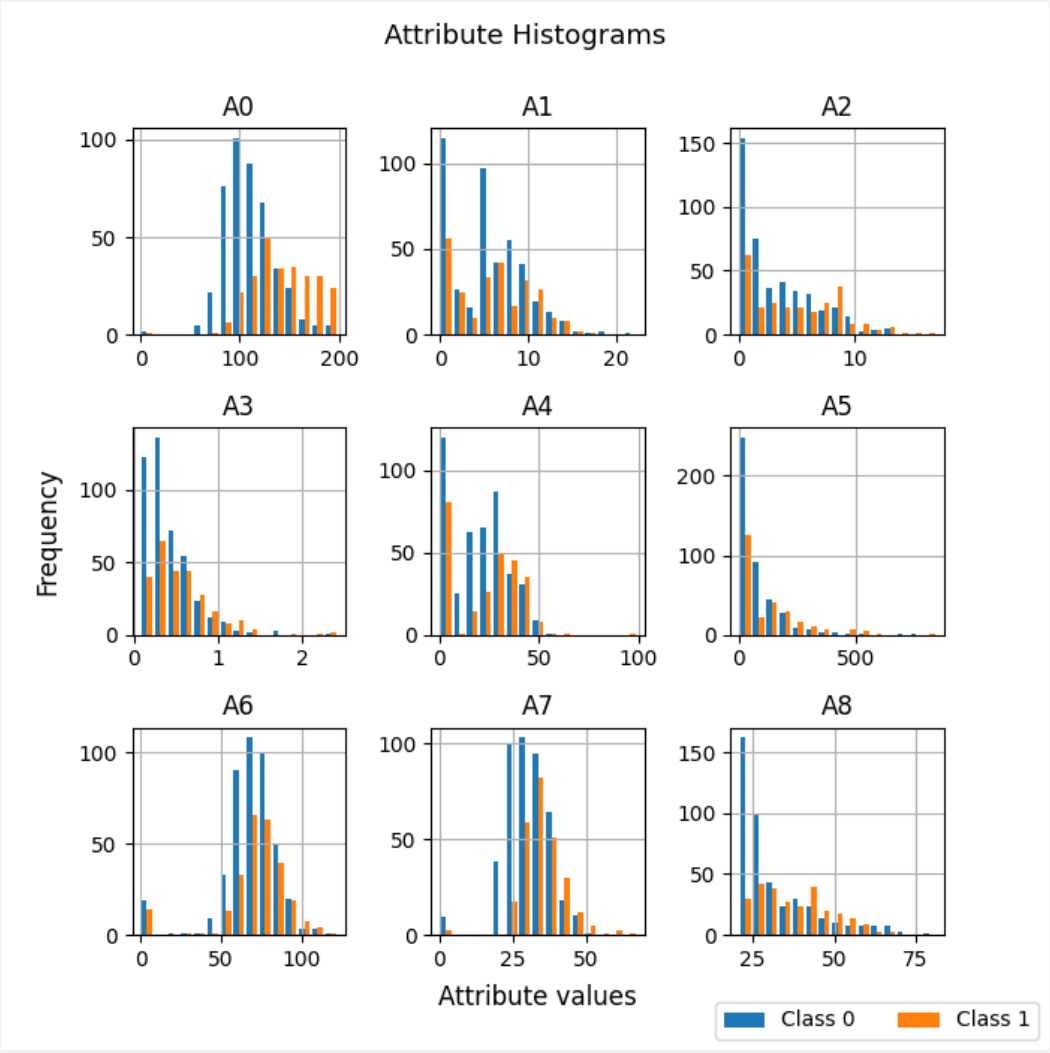
Question 1 : (70 total points) Experiments on a binary-classification data set

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.



Generally, we observe that class 0 and class 1 vary in a similar way for most of the attributes. This may indicate that many of the attributes in our dataset are redundant and do not contribute towards the classification result of instances. The attribute seems to have the most difference between the distributions of class 0 and class 1 is attribute A0 where the instances of class 0 are mostly centered around 100 whereas the majority of class 1's instances are greater than a 100, with a peak around 130. Another thing to point out is that class 1 for some attributes seems be more skewed than class 0. For example, for attributes A3 and A5, class 1 is right-skewed. This shows that that there may be a range for attributes that describes a healthy person (class 0) but the values that an ill person (class 1) can take are not as predictable (they are at the extremes/ out of range).

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	`r` (3 s.f.) between attributes and label `Ytrn`
A0	0.491
A1	0.087
A2	0.227
A3	0.207
A4	0.108
A5	0.186
A6	0.076
A7	0.304
A8	0.240

(b) Attributes which have a large correlation with the label are the ones that have the most effect on the classification of an unseen data point, thus it is a good idea to include them in classification tasks as including them we will get a higher classification accuracy.

(c) That would be a good idea as it seems that those attributes are not a good predictor of the label. By removing them from the dataset, we can reduce the complexity of our model which can avoid overfitting of the training set and thus get a greater classification accuracy on unseen datasets. Moreover, removing irrelevant attributes can decrease substantially the training time and hyperparameter tuning of our classifier.

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 $b > 0, d > 0$.

Consider two new variables as follows:
$\boldsymbol{u'} = a + b\boldsymbol{u}$ and $\boldsymbol{v'} = c + d\boldsymbol{v}$.

The correlation coefficient between $\boldsymbol{u}$ and $\boldsymbol{v}$ is

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

The correlation coefficient between $\boldsymbol{u'}$ and $\boldsymbol{v'}$ is

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

We acknowledge that for a variable x and a constant c:

$\bar{x} = \frac{1}{N} \sum_{i=0}^N x_i, \ and \ \sum_{i=1}^N c = \frac{1}{N} Nc = c$

Thus, $\bar{u'} = a + \frac{b}{N} \sum_{i=1}^N u_i = a + b\bar{u}, \ and \ \bar{v'} = c + \frac{d}{N} \sum_{i=1}^N v_i = c + d\bar{v}$

Rewriting the correlation coefficient:

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

By cancelling out b and d we get:

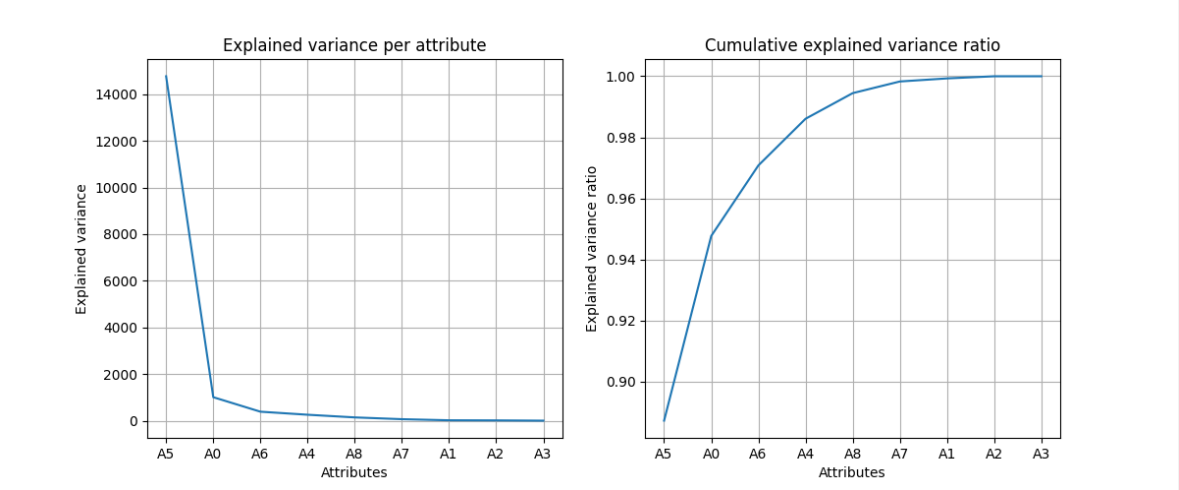
$r_{\boldsymbol{u'v'}} = \frac{\sum_{i=1}^N (u_i - \bar{u})(v_i - \bar{v})}{\sqrt{\sum_{i=1}^N (u_i - \bar{u})^2 \sum_{i=1}^N (v_i - \bar{v})^2}} = r_{uv}$ as required.

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 variances is 16645.64 (2 d.p.)

(b)



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

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

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

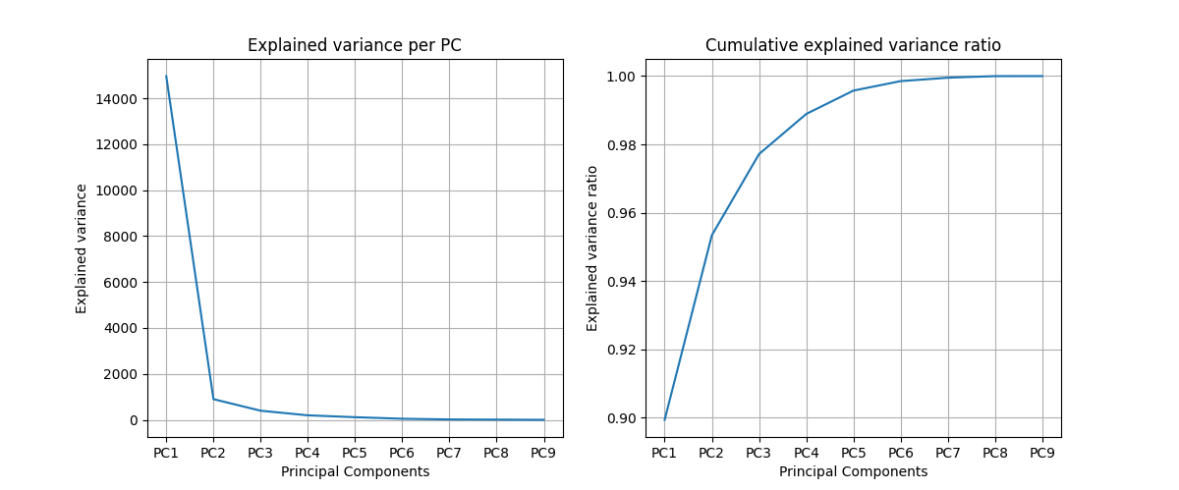
- A graph of the amount of variance explained by each of the principal components.
- A graph of the cumulative variance ratio, where the range of y-axis should be [0, 1].

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

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

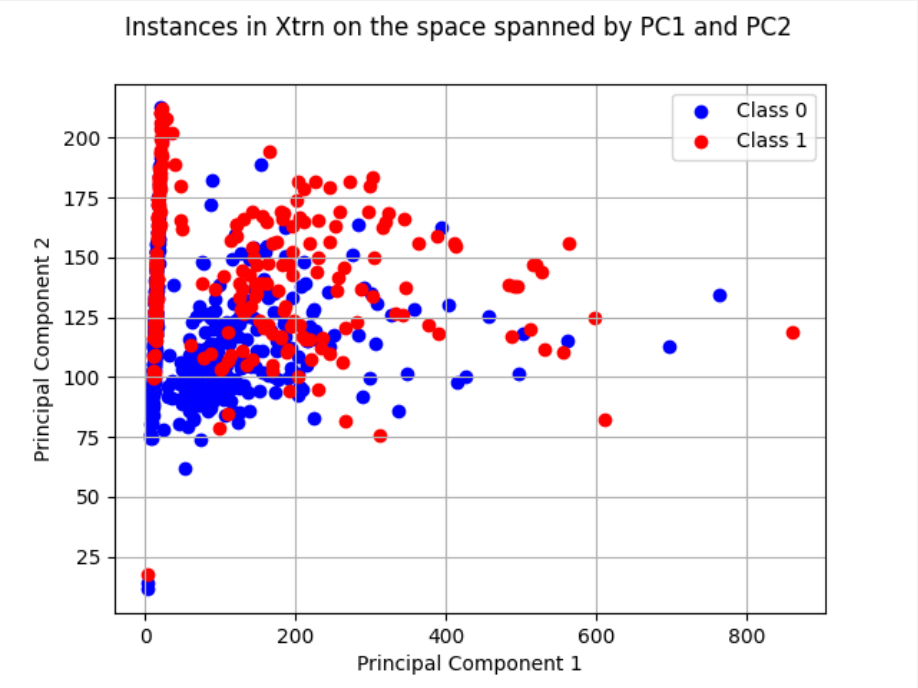
(a) The total amount of unbiased sample variance explained by the whole set of principal components is 16645.64 (2 d.p.).

(b)



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

(c)



(d)

Attribute	Pearson's Correlation Coefficient (3 s.f.)	
	PC1	PC2
A0	0.386	0.914
A1	-0.046	0.091
A2	-0.057	0.225
A3	0.186	0.080
A4	0.459	-0.097
A5	1.00	-0.024
A6	0.101	0.255
A7	0.232	0.173
A8	-0.002	0.373

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

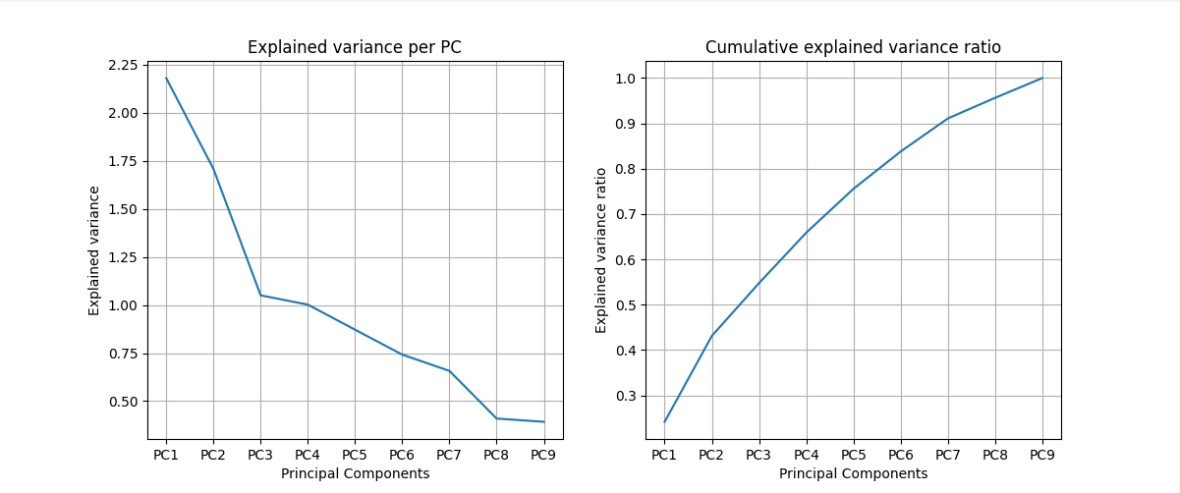
Create the standardised training data `Xtrn_s` and test data `Xtst_s` in your code in the following manner.

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

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

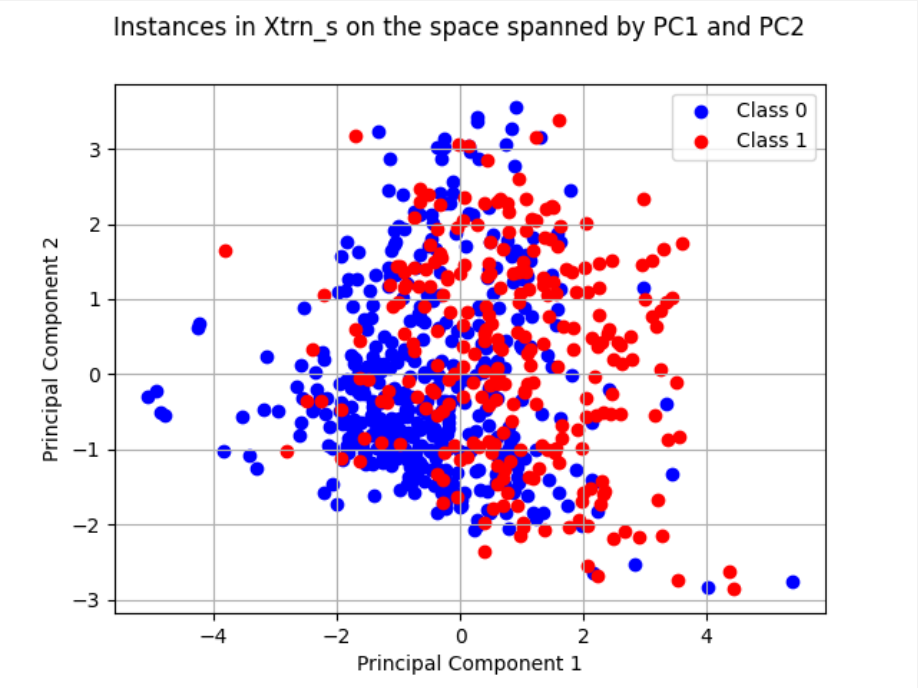
(a) The total amount of unbiased sample variance explained by the whole set of principal components is 9.01 (2 d.p.).

(b)



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

(c)



(d)

Attribute	Pearson's Correlation Coefficient (3 s.f.)	
	PC1	PC2
A0	0.601	0.177
A1	0.057	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

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 total amount of variances in 1.4(a) is equal to the total amount of variance explained by all the principal components in 1.5(a). This is expected since PCs are just the attributes in order of decreasing variance. The set of graphs between part 1.4 and 1.5 are identical with the only difference that in part 1.5 we are plotting against principal components instead of attributes, where the first principal component is the attribute which carries the most variance. It is apparent from the graphs that A5 and A0. or correspondingly PC1 and PC2 carry the most variance across the dataset, where the rest of attributes vary very little.

(b) The difference between the data in part 1.5 and part 1.6 is that in 1.6 we use the normalized data of 1.5, i.e. we transformed the data in a way such that the values will be centred around zero with unit standard deviation. The correlation coefficients, and thus, the explained variance per PC have changed after we have scaled our data. All of the PCs seem to have some amount of contribution in 1.6, unlike in 1.5 where the variance explained by PC3 onwards seemed to be unsubstantial. For this reason, for this particular dataset it is important that we standardise the data before PCA as we understand better the scale by which each attribute contributes t the total variance of the data. In addition when normalized, we can get more distinct groups of classes when plotting PC2 against PC1, as also seen in part 1.6(c) compared to part 1.5(c).

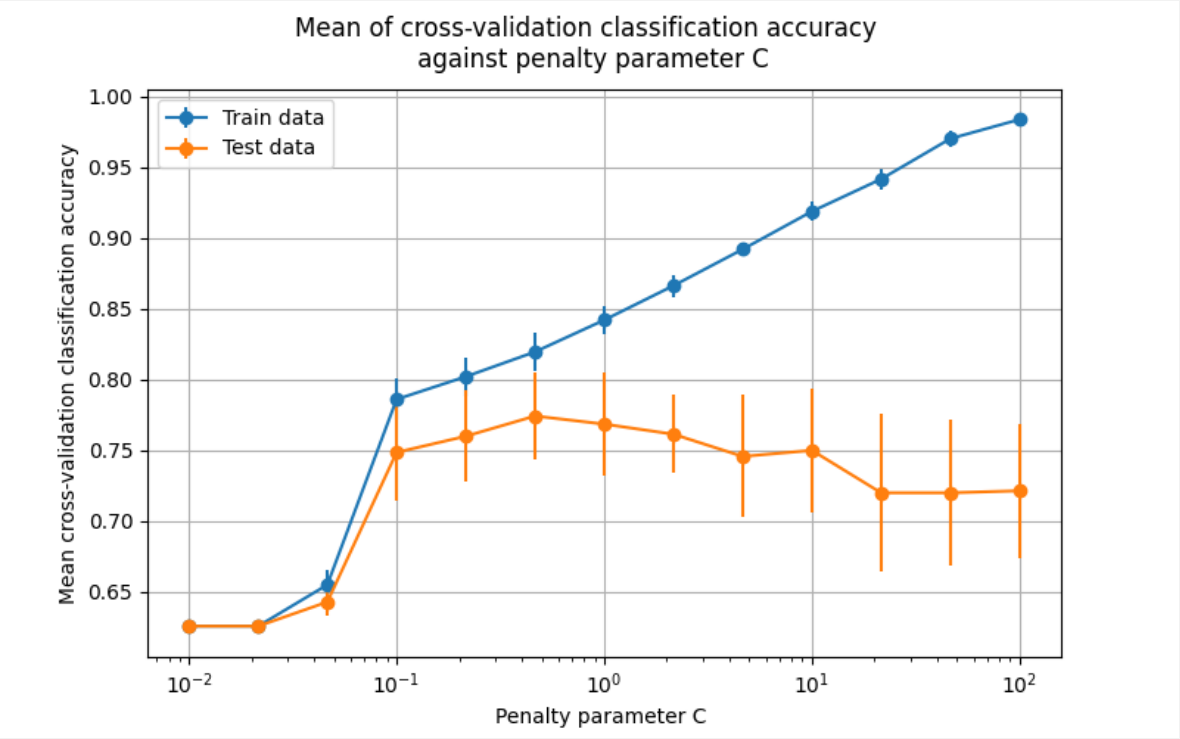
Question 1 continued on next page...

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) We observe that as C increases, the mean cross-validation accuracy of the train data increases and its standard deviation decreases. This is expected, as a high C corresponds to a smaller hyperplane margin where misclassifications are rarer to take place. For the test data however, it seems that at around C=1 the mean cross-validation accuracy starts to drop and its standard deviation to increase, indicating that the model becomes a worse predictor. This is because the model starts to overfit on the training data, and is unable to do well on unseen data. From the graph we can see that the optimum value for C is around 0.8 where the mean cross-validation accuracy for the test data is the highest at 0.78.

(c) The highest mean cross-validation accuracy is 0.774 (3 s.f.) which is for the regularization parameter C = 0.464 (3 s.f)

(d) The number of instances classified correctly is 75 and the classification accuracy is 0.75.

Question 1 continued on next page...

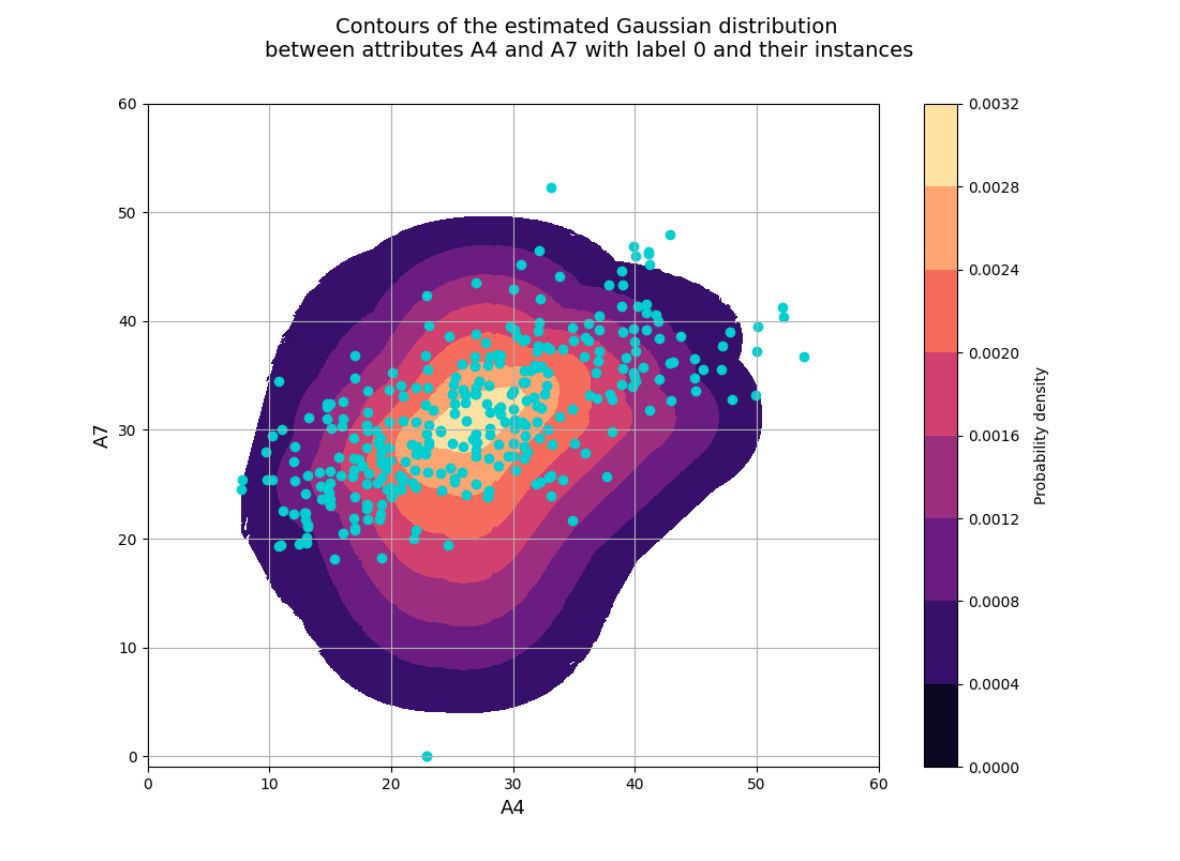
1.9 (5 points) We here consider a two-dimensional (2D) Gaussian distribution for a set of two-dimensional vectors, which we form by selecting a pair of attributes, A4 and A7, in `Xtrn` (NB: not `Xtrn_s`) whose label is 0. To make the distribution of data simpler, we ignore the instances whose A4 value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

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

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

(a) mean = $\begin{bmatrix} 27.021 & 31.093 \end{bmatrix}$ covariance matrix = $\begin{bmatrix} 95.141 & 41.470 \\ 41.470 & 46.693 \end{bmatrix}$

(b)



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.

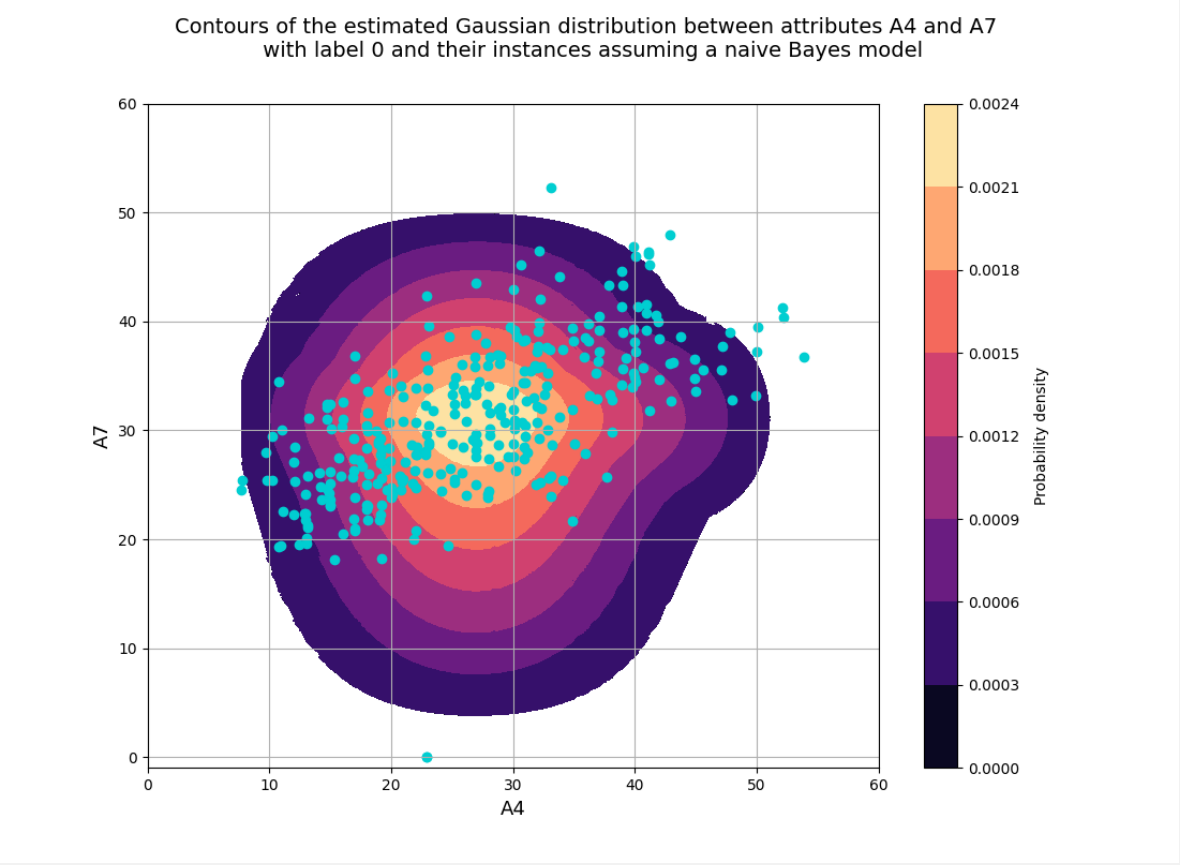
(a) Report the sample mean vector and unbiased sample covariance matrix of the Gaussian distribution.

(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) $\text{mean} = \begin{bmatrix} 27.021 & 31.093 \end{bmatrix}$ $\text{covariance matrix} = \begin{bmatrix} 95.141 & 0.0 \\ 0.0 & 46.693 \end{bmatrix}$

(b)



(c) When assuming naive-Bayes we assume conditional independence between the two attributes, A4 and A7, thus the covariance between them is 0. Thus, in 1.10 the Gaussian distribution is much more centralised compared to the distribution in 1.9 where we have a full covariance matrix which makes the distribution skewed. When not assuming independence, in 1.9, the attributes A4 and A7 are positively correlated, thus the distribution is elongated towards the A4=A7 axis. Finally, in this particular case, I don't think it is proper to assume a naive-Bayes as we are using a medical data set where correlations between attributes are important and could lead to important conclusions on the health state of individuals.

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.

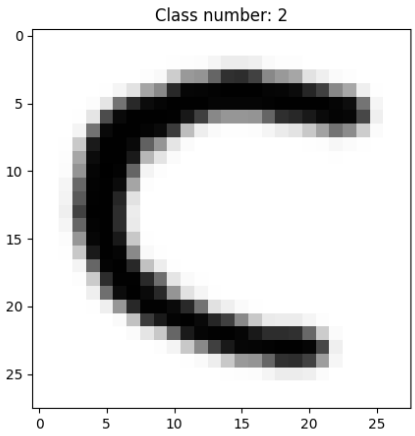
(a)

	`Xtrn`	`Xtst`
max	1.0	1.0
min	0.0	0.0
mean	0.177	0.176
st. deviation	0.335	0.333

(b)

instance 0:

instance 1:

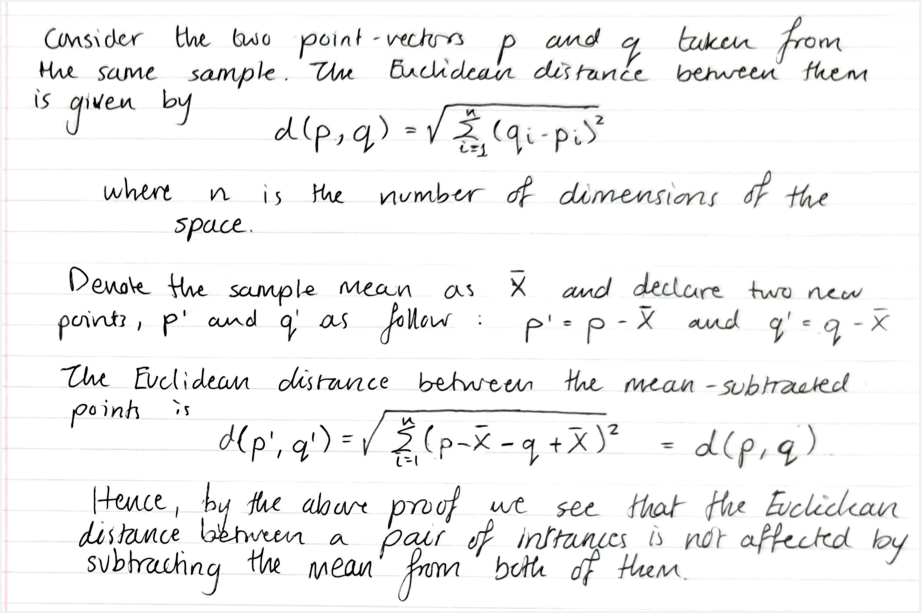


2.2 (4 points)

(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)



Consider the two point-vectors p and q taken from the same sample. The Euclidean distance between them is given by

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$$

where n is the number of dimensions of the space.

Denote the sample mean as $\bar{x}$ and declare two new points, p' and q' as follow: $p' = p - \bar{x}$ and $q' = q - \bar{x}$

The Euclidean distance between the mean-subtracted points is

$$d(p',q') = \sqrt{\sum_{i=1}^{n}(p - \bar{x} - q + \bar{x})^2} = d(p,q)$$

Hence, by the above proof we see that the Euclidean distance between a pair of instances is not affected by subtracting the mean from both of them.

(b) The same mean must be used to centre both the training as well as the testing data. We build a model for the purpose of using it on unseen data to get predictions, so using the testing data's mean would result in data leakage as we must treat the test set as unseen data and thus shall not be used in data processing. Secondly, the testing set is far too small to give an unbiased estimate of the mean so the training set must be used.

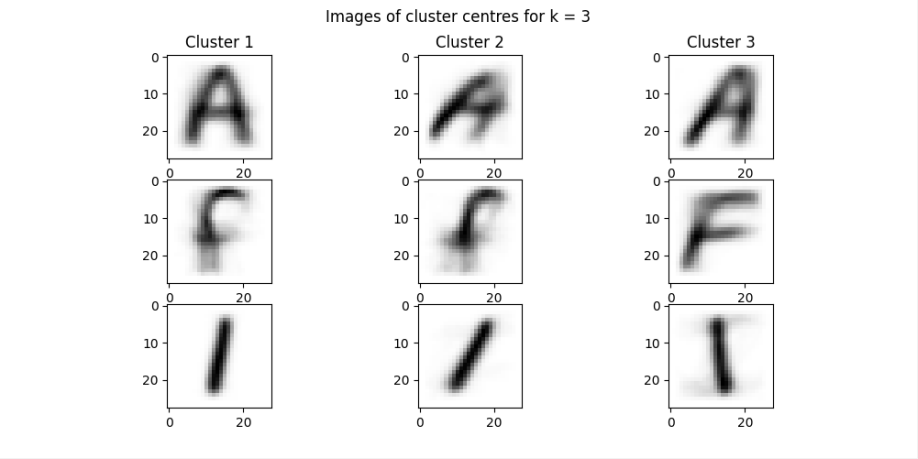
Question 2 continued on next page...

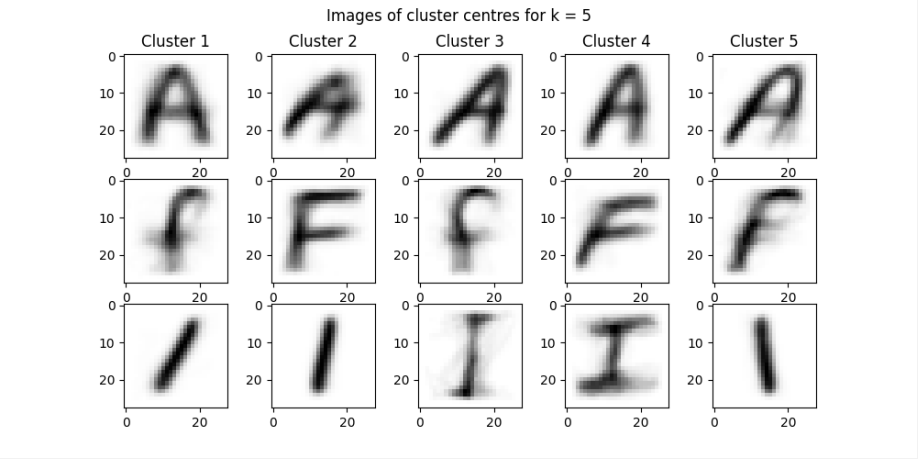
2.3 (7 points) Apply k-means clustering to the instances of each of class $0, 5, 8$ (i.e. 'A', 'F', 'I') in `Xtrn` with $k = 3, 5$, for which use Sklearn's `KMeans` with `n_clusters=`k and `random_state=0` while using default values for the other parameters. Note that you should apply the clustering 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) When k=5 we are able to get a greater variety of the different handwriting styles, since we get 2 extra cluster centres for each class compared to k=3. Cluster centres in k=3 appear more fuzzy, since 3 clusters are not enough to describe the whole variety set of different handwriting styles (eg. tilted, curved, etc.), whereas with k=5 it seems that we get concrete cluster centres.

Using this data set for classification tasks may be challenging because of the high-dimensionality and the high level of noise that the data has. It is suggested that feature selection or dimensionality reduction takes place before attempting to build a classifier for the data so that we avoid overfitting the model to the training set by building a complex model.

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.

SSE is the sum of the squared differences between each instance and its cluster's mean. It can be used as a measure of variation within a cluster. If all cases within a cluster are identical the SSE would then be equal to 0.

(a) Since the k-means algorithm allocates each point to the cluster centroid in closest proximity to it, by adding a new cluster, there is a greater chance of having a centroid in closest distance to the point. Thus, this can only lead to a decrease in the sum of squared error.

(b) Your Answer for (b) Here

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 is 0.916. The classification accuracy of the test set is 0.722.

(b) The five classes that were misclassified most in the test set with decreasing order of misclassifications are

Class	# of misclassifications
11 ('L')	53
17 ('R')	48
8 ('I')	42
10 ('K')	38
13 ('N')	36

(c) A possible reason for the misclassification of these classes is that the letter they represent is too similar to other letters, for example, handwritten 'L' may be similar to an 'I' with horizontal lines on each end of the 'i'. Moreover, 'R' is too similar to 'Q' and 'K'. Similarity here is in terms of how many pixel positions two letters may share.

Question 2 continued on next page...

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

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

(a) **1. Stochastic Gradient Descent (SGD):** SGD algorithms are a variant to the usual gradient descent carried out by a logistic regression classifier. SGD calculates the gradient using just a randomly-chosen portion of the data points instead of using all of them as gradient descent does. The randomness introduced by SGD allows it to escape from local minima to reach a better minimum - oftentimes getting close to the global minimum. Most times, SGD can converge significantly faster than the ordinary gradient descent algorithm, however a drawback of SGD can be the extensive hyperparameter tuning needed due to the many hyperparameters involved with it. Thus, SGD is most likely to be useful in huge datasets where getting the hyperparameters optimized is worth the trouble for the decrease in computation time.

2. Hyperparameter tuning: Hyperparameter tuning is the problem of choosing a set of optimal values for the hyperparameters of our model. In In logistic regression, there are several of these values.Some examples are the penalty method of the model and C, the penalty strength. We can apply hyperpameter tuning using a Grid Search. A Grid Search allows to define a 'grid' of the hyperparameters we want to test out in the model, then passes all combinations of hyperparameters separately into the model and gives the cross-validated accuracies of the different combinations. We can then choose the best combination of hyperparameter values to use in our model.

3. Feature selection: The feature selection method is useful when dealing with high-dimensional data as it aims to improve the classification accuracy by removing any features (or attributes) from the data that are irrelevant, or that introduce noise to the data that produces poor predictions. High-dimensionality may lead to overfitting as it can increase the complexity of a model substantially. Thus we can find how important each of the attributes are for our model and remove entirely from our dataset the bad or neutral ones to create a more accurate classification algorithm.

I will try to improve the classification accuracy of the logistic regression model through a feature selection followed by hyperparameter tuning.

(b) I used Sklearn's recursive feature elimination with cross-validation (RFECV) algorithm to select the number of features with the following parameters: {estimator=lr, cv=StratifiedKFold(3), scoring="accuracy"} and fitted this model to the `Xtrn_m` data, and then transformed it using the `fit_transform` method. I then fitted the transformed train data to a Sklearn's GridSearchCV model using the parameters {estimator=lr, param_grid=dict(penalty=['l1', 'l2'], C=[0.0001,0.001,0.01,0.1,1,10,100,1000], solver=['lbfgs', 'liblinear'], max_iter=[1000, 2000]), cv=3} where lr=LogisticRegrssion(max_iter=1000, random_state=0).

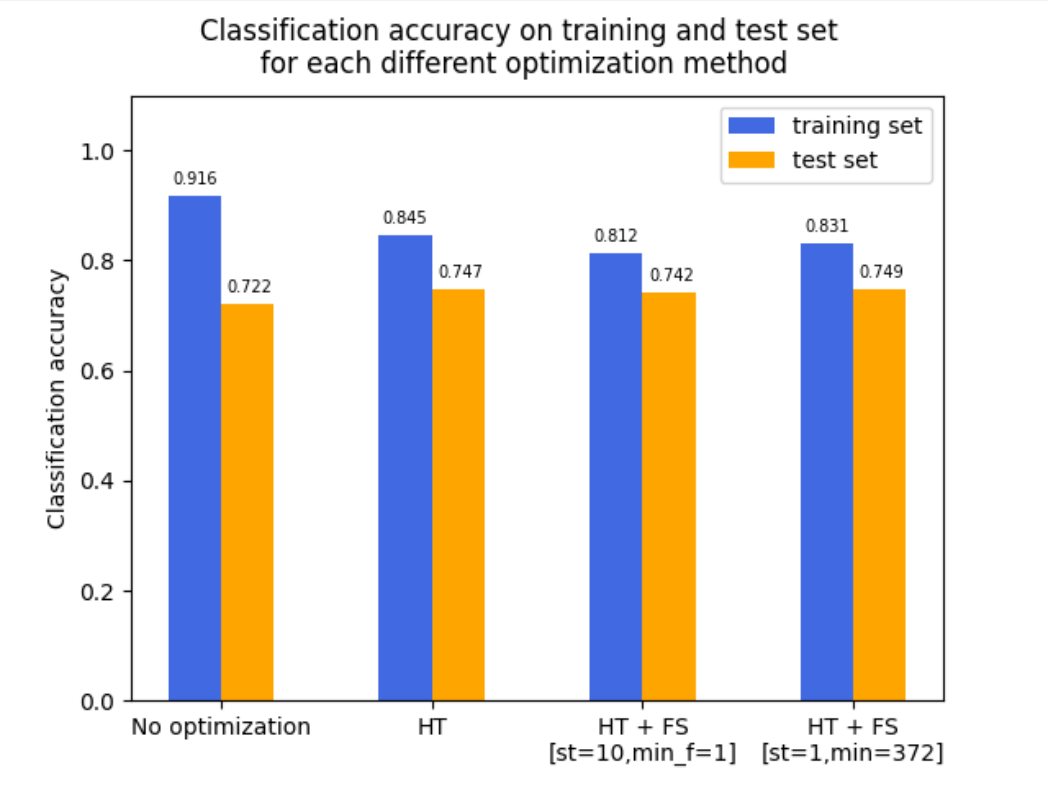
Question 2 continued on next page...

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

(c) **1.** Results on just hyperparameter tuning without feature selection on the normalized data Xtrn_m:
Best average cross-validation score: 0.746282 using hyperparameters
{'C': 0.1, 'max_iter': 1000, 'penalty': 'l2', 'solver': 'lbfgs'}
Rerunning a logistic regression model with the above parameters we get:
classification accuracy on training set: 0.844872
classification accuracy on test set: 0.747308

2. Results on hyperparameter tuning with feature selection on the normalized data Xtrn_m with parameters step=10, min_features_to_select=1.
Optimal number of features=224
Best average cross-validation score: 0.741923 using hyperparameters
{'C': 0.1, 'max_iter': 1000, 'penalty': 'l2', 'solver': 'lbfgs'}
Rerunning a logistic regression model with the above parameters we get:
classification accuracy on training set: 0.811923
classification accuracy on test set: 0.741923

3. Results on hyperparameter tuning with feature selection on the normalized data Xtrn_m with parameters step=1, min_features_to_select=372.
Optimal number of features=384
using hyperparameters {'C': 0.1, 'max_iter': 1000, 'penalty': 'l2', 'solver': 'lbfgs'}
classification accuracy on training set: 0.831410
classification accuracy on test set: 0.748846



From the above results we can say that our result in 2.5 (No optimization) was overfitted since using optimization techniques, such as hyperparameter tuning (HT) and feature selection (FS) we got a smaller classification accuracy on the training set but a greater on on the test set. The highest classification accuracy on the test set (0.749 3 s.f.) is given by the combination of FS with step=1 and minimum features to select=372 (half the original number of features).

Question 2 continued on next page...

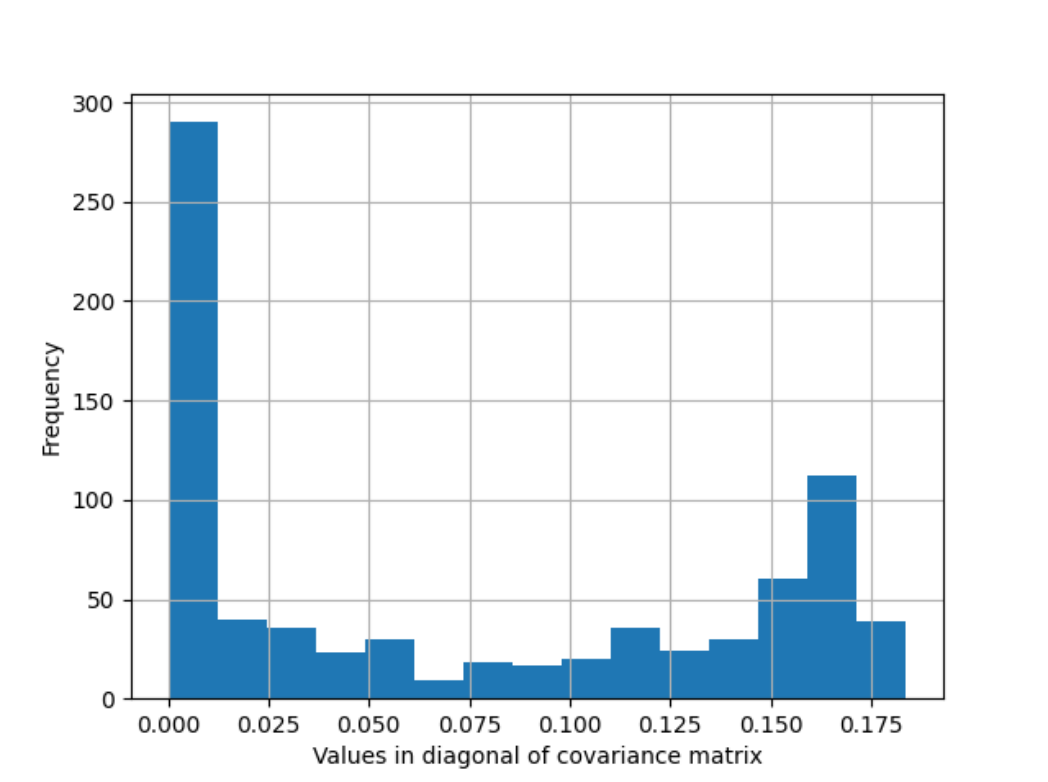
2.7 (9 points) Using the training data of class 0 ('A') from the training set `Xtrn_m`, calculate the sample mean vector, and unbiased sample covariance matrix using Numpy's functions, and answer the following.

(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) minimum = -0.0975
maximum = 0.184
mean = 0.00171

(b) minimum = 0.0
maximum = 0.184
mean = 0.0723

(c)



(*continued from the previous page for Q*)

(d) The error I have received is
`numpy.linalg.LinAlgError: singular matrix`.
I got this error because the `multivariate_normal` function by default doesn't allow singular covariance matrices. This is because singular matrices are non-invertible, and thus have a determinant of zero. Since `multivariate_normal` divides by the determinant of the covariance matrix, the error is raised. In our case the covariance matrix is singular, thus it is incompatible with the multivariate Gaussian model in this case.

(e) One possible solution is to set the parameter `allow_singular` of the `multivariate_normal` function to `True`, so that it can accept singular covariance matrices. Secondly, we could add a very tiny, non-vanishing noise to a singular matrix to make it non-singular. By adding this tiny noise, we get an invertible covariance matrix whose values are still close enough to the real values of the original singular matrix. Finally, we could use the Sklearn's `GaussianMixture` model with `n_components` = 1 to get a multivariate Gaussian model which adds a noise of 1e-6 to the covariance matrix by default.

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

(a) Train the model using the data of class 0 ('A') in the training set, and report the log-likelihood of the first instance in the test set with the model. Explain why you could calculate the value this time.
(b) We now carry out a classification experiment considering all the 26 classes, for which we assign a separate Gaussian distribution to each class. Train the model for each class on the training set, run a classification experiment using a multivariate Gaussian classifier, and report the number of correctly classified instances and classification accuracy for each training set and test set.
(c) Briefly comment on the result you obtained.

(a) The log-likelihood of the first instance in the test set with the Gaussian Mixture model is -1712612.74.
I was able to calculate the value this time because the GaussianMixture function adds a small constant noise over the diagonal of the covariance matrix which makes it non-singular. This noise is given by the parameter reg_covar and by default it's value is 1e-6.

(b) I have trained a Gaussian Mixture Model of a single Gaussian distribution for each of the 26 classes fitting the model to the instances of the corresponding class in Xtrn_m. I have then recorded the number of correct predictions and calculated the classification accuracy for each of the 26 classifiers on the whole dataset of Xtrn_m and then again on the Xtst_m. The number of correct classifications for the whole set of classifiers on the training data was 300, while again for the whole set of classifiers on the test data it was 100. Then, the classification accuracy for the whole set of classifiers for both the training set and the test set was 0.0385.

(c) The results show that all the classifiers were able to predict correctly all the instances of the class they were trained with and predict falsely everything else, in both the training data and test data sets. This makes sense since each model was only trained with a single Gaussian distribution on one class and then tried to predict on all of the other classes. Since the model has not been trained on any of those other classes, it wouldn't be able to make a positive prediction for those. The models however, got very well trained on the class they were fitted on, that could predict correctly all of the unseen instances of that class in the test set.

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) GMMs is a clustering method, thus it is fitted without taking into consideration the labels of our data (even if we have them). Hence, MLE cannot be used for estimating the parameters of GMMs directly because the Gaussian mixture model is a latent variable model, i.e., it depends on latent variables - the means, the covariance matrices, and the probabilities of the clusters (or classes) are all unknown. Thus, an intermediate step is needed in estimating maximum likelihoods, which is why the Expectation Maximization algorithm is used for training GMMs.

(b) The EM algorithm has 2 components that take place iteratively until the optimised parameters are found. These are the Expectation (E)-step and the Maximization (M)-step. It iteratively assigns probabilities to clusters given the points position (soft-clustering procedure), and estimates the mean and covariance matrix of each cluster given the data. On the other hand, the k-means + MLE algorithm behaves similarly but lacks two main features. First the k-means builds clusters differently. It results to hard clustering, by only considering the mean of the data points in updating the centroids, while the EM algorithm considers both the mean and the covariance matrices in updating the Gaussian distributions. Thus, k-means attempts to cluster the data points in a circular fashion, irrespective of the underlying distribution of the data. Secondly, the k-means and MLE do not interact iteratively, as it happens through EM. But since MLE is a supervised method, it takes the concrete labels created by k-means and tries to estimate the parameters using those clusters. As a result, we take the ill-formed clusters of k-means (ill-formed as it lacks consideration of the covariance matrices) and try to optimize the model parameters from there on. Conclusively, the differences described above explain why the GMM clustering is executed using the EM algorithm instead of k-means + MLE.