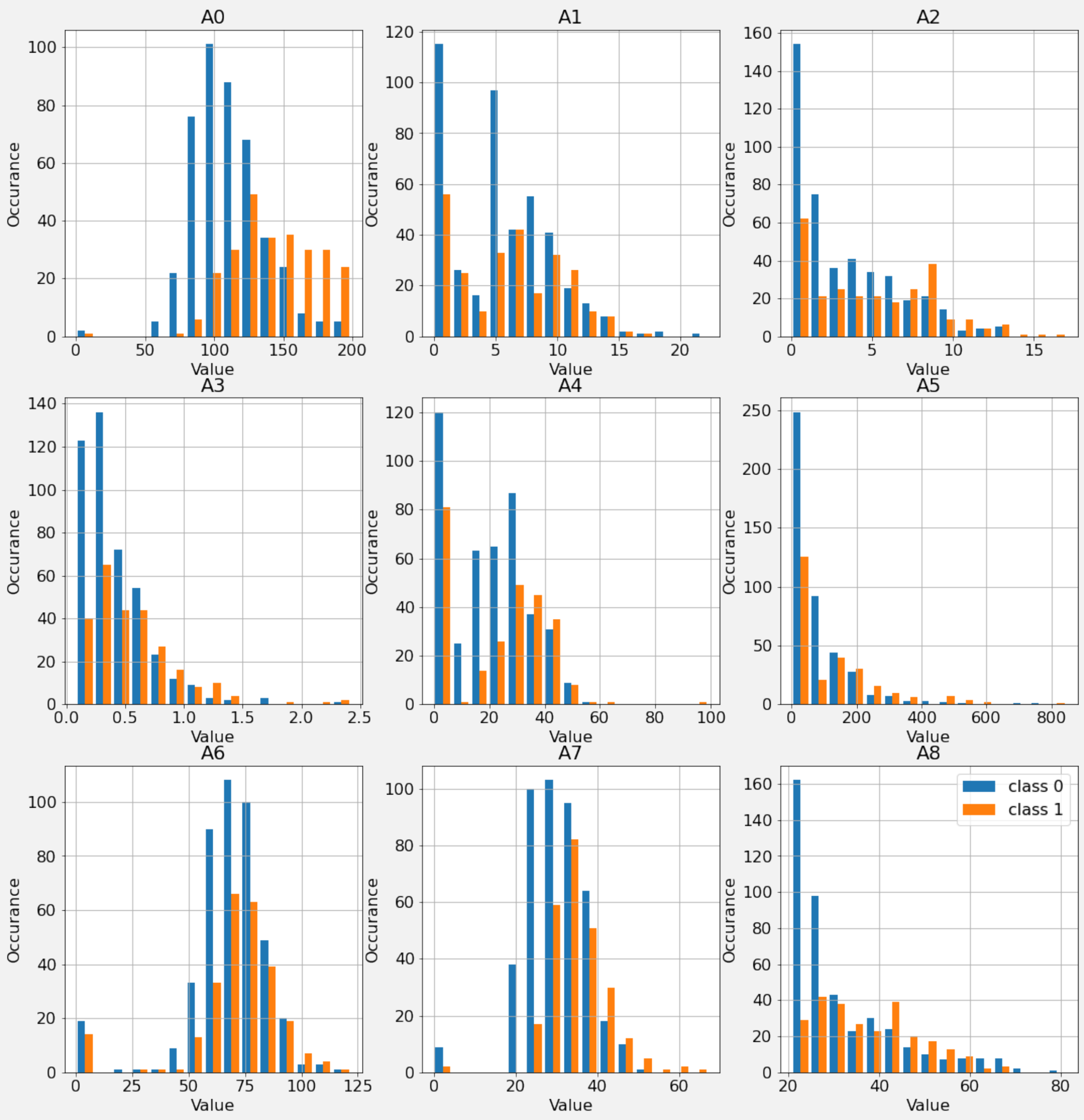
Question 1 : (70 total points) Experiments on a binary-classification data set

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.

Features in Xtrn distributed for each class



Class imbalance problem exist in the dataset: instances of class 0 is more than instances of class 1. So we should always keep in mind to find a data augmentation or loss function that could fight class imbalance. The ranges of attributes are also drastically different, range of over 800 and 2.5 both exist. So the dataset needs to be normalized. Occurrences are generally normally distributed and log scale distributed. But outliers could be spotted in attributes A0, A6 and A7. So some pre-processing might need to be done. The two classes also have some overlap so the data is not perfectly linearly separable.

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)

	A0	A1	A2	A3	A4	A5	A6	A7	A8
Ytrn	0.491	0.087	0.227	0.207	0.108	0.186	0.076	0.304	0.240

(b) Yes. If an attribute is strongly correlated with the label, a linear function is able to achieve a good score when predicting. Using those attributes could help the model captures features, simplifies the model, and increase accuracy.

(c) No. Since non-linear classifier can exploit complex relationships between features inside the model, attributes have small correlation with labels might be important for model in connection with other attributes. PCA is a better solution if feature reduction is needed.

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 $b > 0, d > 0$.

$$\begin{aligned}
r_{(a+bu)(c+dv)} &= \frac{\sum\{((a+bu_i)-(a+b\bar{u}))\,((c+dv_i)-(c+d\bar{v}))\}}{\sqrt{\left\{\sum((a+bu_i)-(a+b\bar{u}))^2\sum((c+dv_i)-(c+d\bar{v}))^2\right\}}} \\
&= \frac{\sum(b(u_i-\bar{u})d(v_i-\bar{v}))}{\sqrt{\sum(b(u_i-\bar{u}))^2\sum(d(v_i-\bar{v}))^2}} \\
&= \frac{bd\sum((u_i-\bar{u})(v_i-\bar{v}))}{\sqrt{b^2d^2\sum(u_i-\bar{u})^2\sum(v_i-\bar{v})^2}} \\
&= \frac{\sum((u_i-\bar{u})(v_i-\bar{v}))}{\sqrt{\sum(u_i-\bar{u})^2\sum(v_i-\bar{v})^2}} \\
&= r_{uv}
\end{aligned} \tag{1}$$

When doing linear transformation, the mean of the set needs to be transformed into $a + b\bar{u}$ and $c + d\bar{v}$. After subsituting all the variables into the equation, a and c are canceled out. b and d are also canceled out after getting those values outside the sum. Afterwards, the equation is equal to the original and r_{uv} does not change under linear transformation. Geometrically speaking, correlation coefficient is the *cos* of the angle between two vectors. Since *cos* is invariant under linear transformation, so does correlation coefficient.

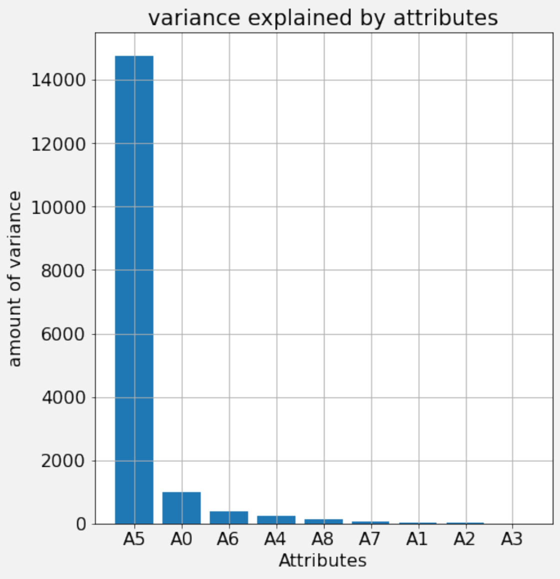
Question 1 continued on next page...

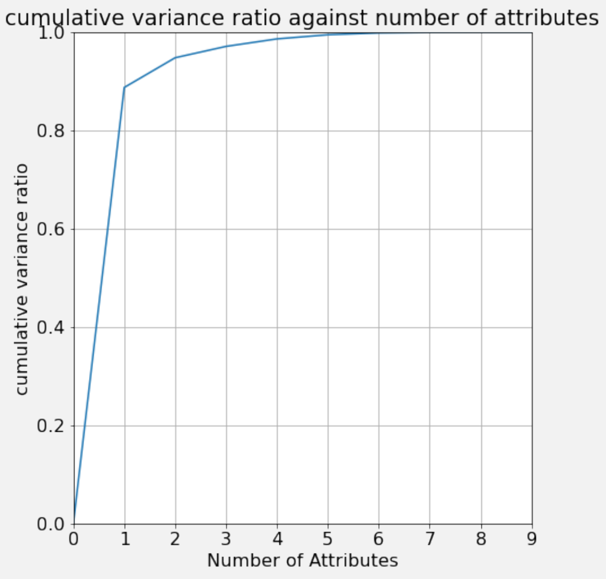
1.4 (5 points) Calculate the unbiased sample variance of each attribute of `Xtrn`, and sort the variances in decreasing order. Answer the following questions.

(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 variance is 16645.64





(b)

(The graph on the right: connected the point of the first attribute to origin)

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.

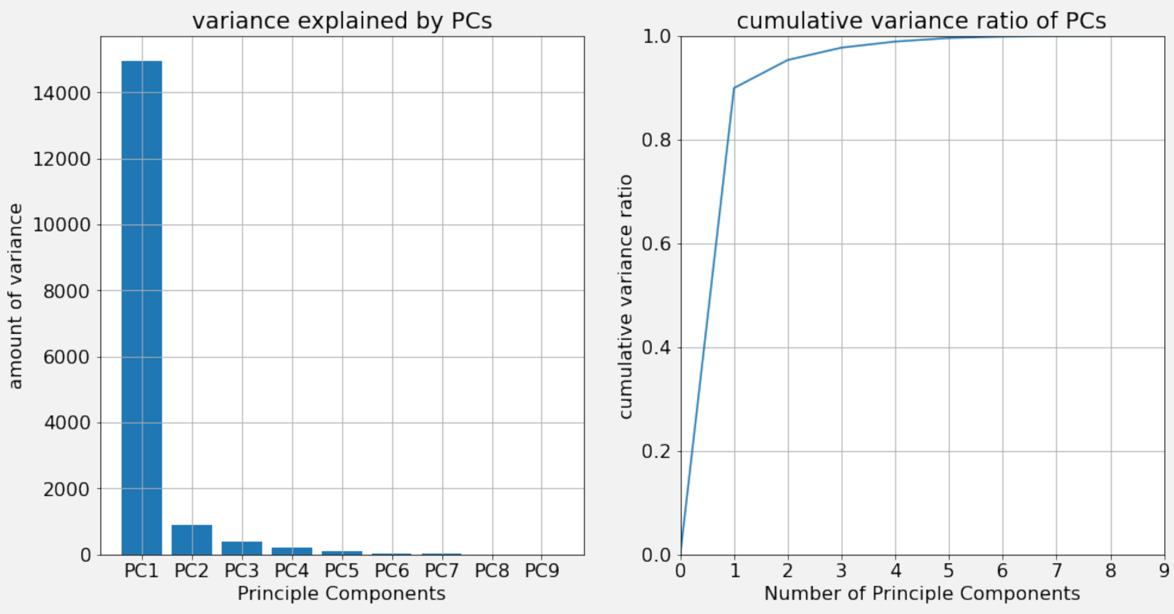
- A graph of the amount of variance explained by each of the principal components.
- A graph of the cumulative variance ratio, where the range of y-axis should be [0, 1].

(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()`.

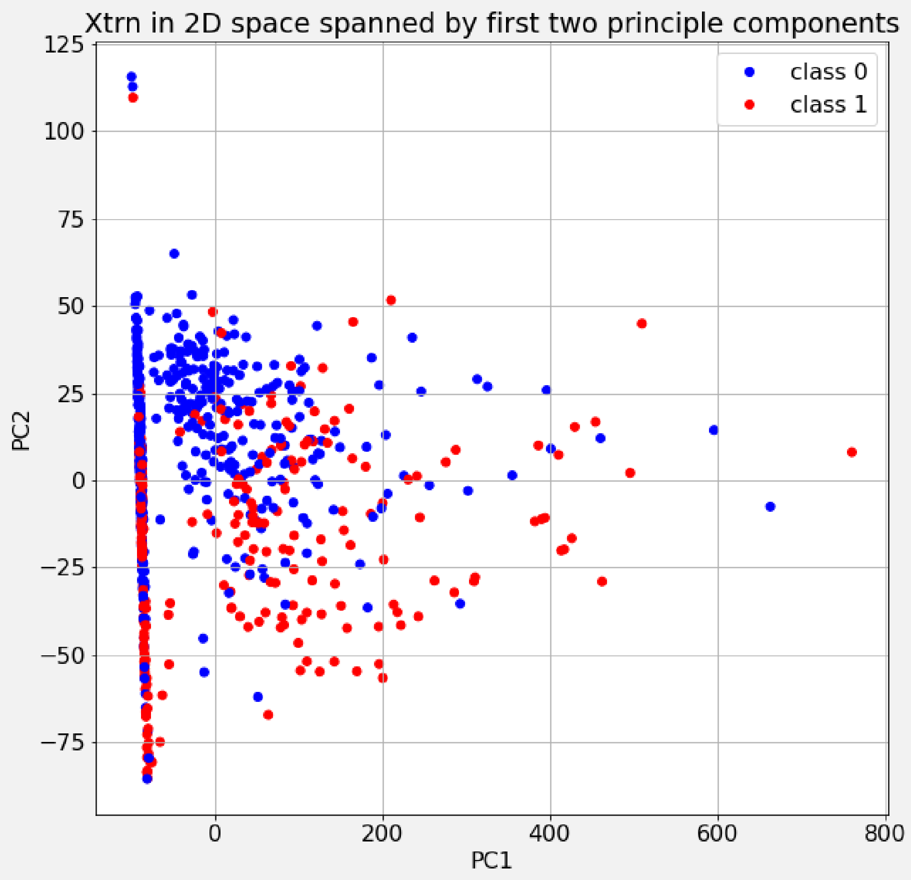
(d) Calculate the correlation coefficient between each attribute and each of the first and second principal components, report the result in a table.

(a) Total amount of unbiased sample variance explained by PCs is 16645.64

(b)



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



(c)

(d)

	A0	A1	A2	A3	A4	A5	A6	A7	A8
PC1	0.386	-0.046	-0.057	0.186	0.458	0.999	0.101	0.232	-0.002
PC2	-0.914	-0.091	-0.225	-0.080	0.097	0.024	-0.255	-0.173	-0.373

Question 1 continued on next page...

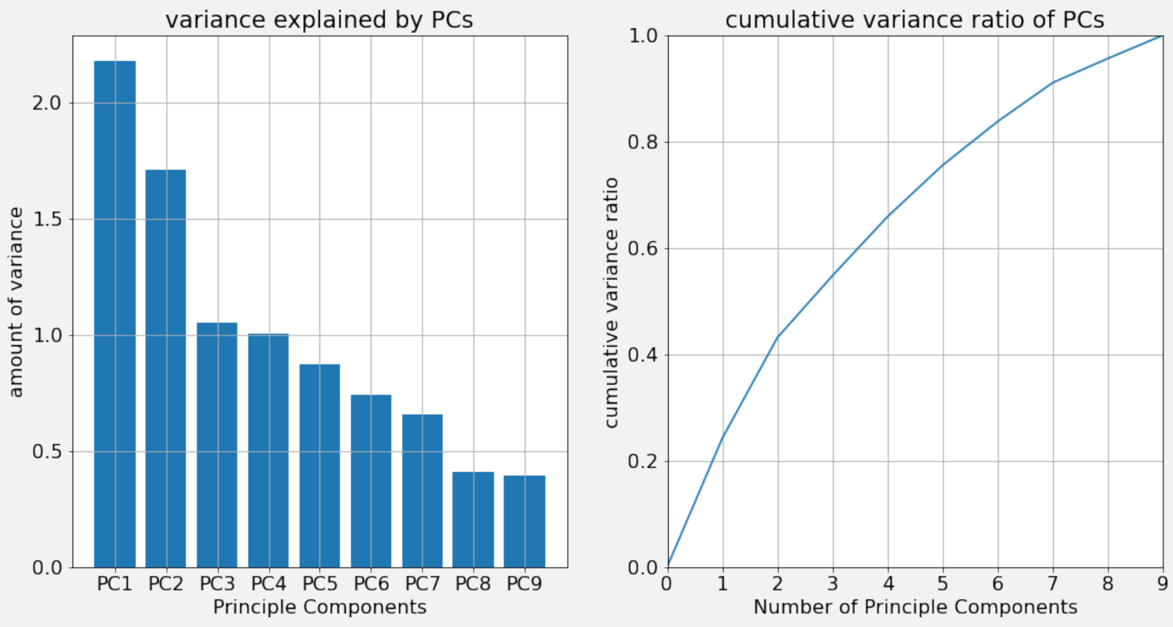
1.6 (4 points) We now standardise the data by mean and standard deviation using the method described below, and look into how the standardisation has impacts on PCA.

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

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

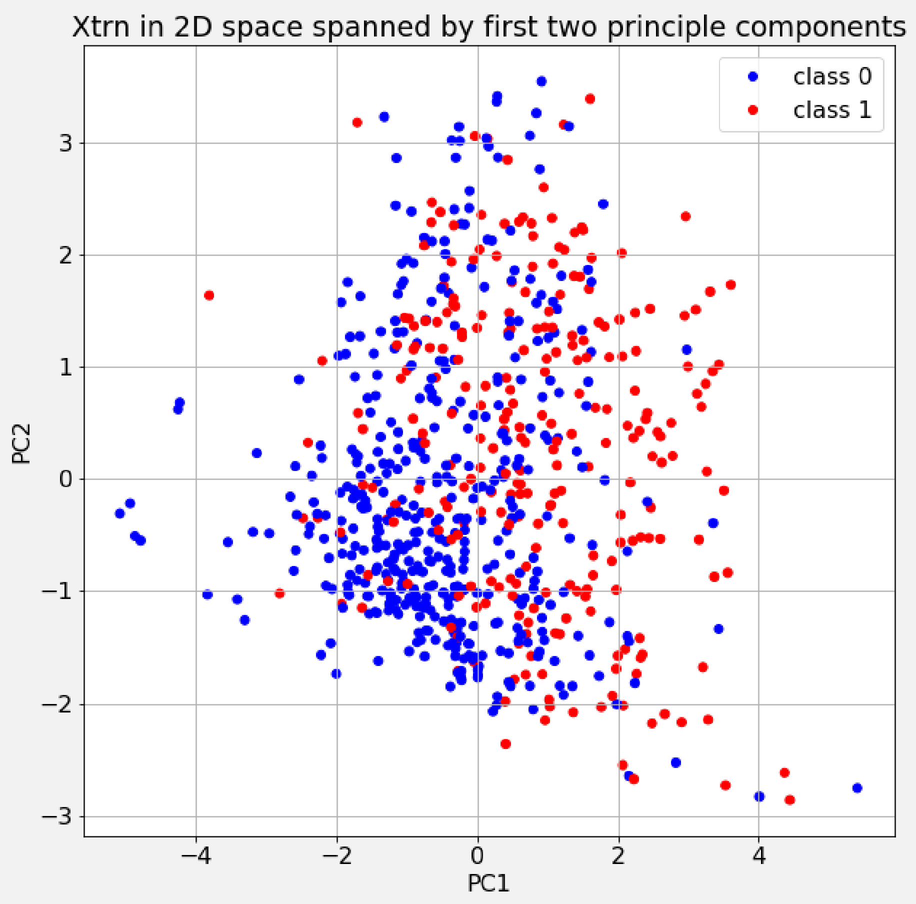
Using the standardised data `Xtrn_s` instead of `Xtrn`, answer the questions (a), (b), (c), and (d) in 1.5.

(a) Total amount of unbiased sample variance explained by PCs is 9.01



(b)

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



(c)

(d)

	A0	A1	A2	A3	A4	A5	A6	A7	A8
PC1	0.601	0.057	0.267	0.366	0.623	0.630	0.523	0.651	0.353
PC2	0.177	0.100	0.760	-0.208	-0.466	-0.370	0.224	-0.168	0.781

Question 1 continued on next page...

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

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

(a) Total amount of variances are equivalent, meaning using full set of PCs explains all the data. The directions of PCs are close to that of the original attributes according to graphs in (b), which are similar, So PCA doesn't do much transformation. Also, A5 & PC1 dominates the variance: comparing with graph of 1.1, the larger the range an attribute have, the greater the variance. This creates the false assumption that the bottom 4 attributes contributes almost nothing to the variance and could be ignored. The same is true for PCs since they are similar to attributes in this case.

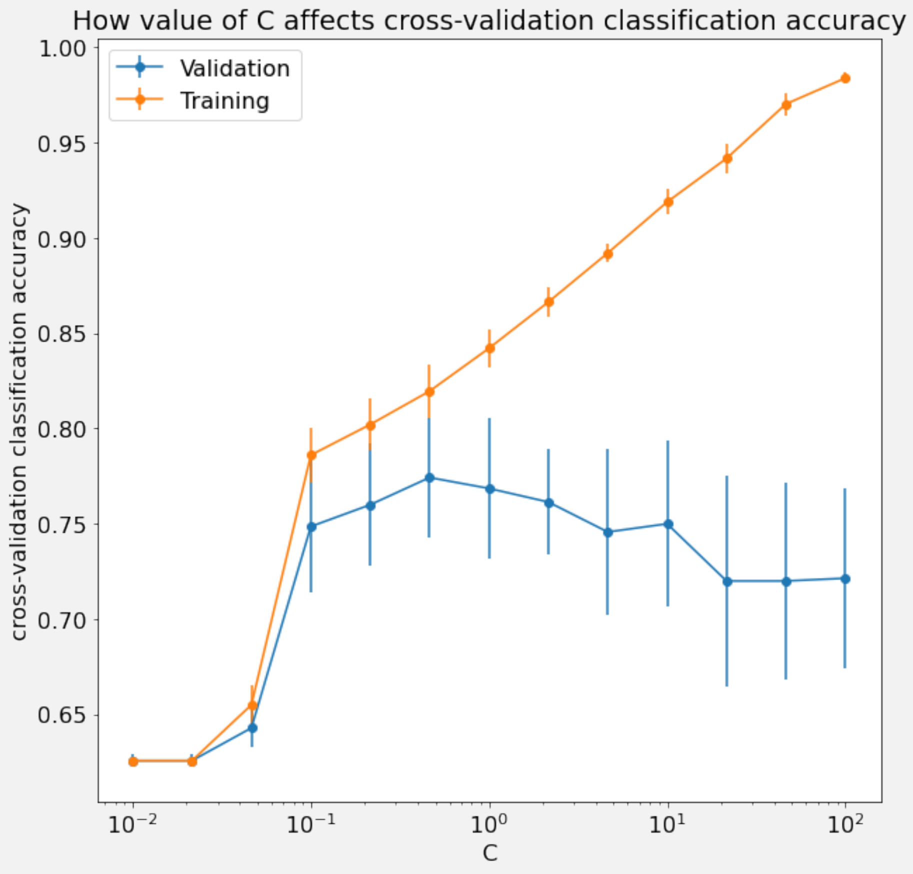
(b) Total variance in 1.6 is 9.0, thus the data is standardized by std. The graphs in 1.6b indicate that each PCs contributes to the model similarity after normalization compare to the logrithmic contributions in 1.5b. So the bottom 4 PCs should NOT be ignored. 1.6c resembles two normal distributions better. Although some overlaps and outliers exists, the graph gives more solid classification than 1.5c, which most instances have similar PC1 value. PC1 and PC2 are no longer extremely strong correlated with attributes in 1.6d, meaning they transformed well by PCA. Since attributes have different scales, standardizing the data could spread the attributes within similar scales and not create bias against attributes with small range. This ensures the data is internally consistent and improve its quality. So I am strongly advised to standardize.

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) Mean training accuracy keeps increasing suggesting the model is learning and using small C value is underfitting. But variance of training accuracy drops to almost 0 when using largest C values, mean validation accuracy decrease while its standard deviation increase, and the gap between training and validation accuracy keeps expanding. This is a clear sign of overfitting with large C values

(c) Highest mean CV accuracy: 77.4%, yielded by C = $10^{1/3}$

(d) Accuracy for test set: 0.75; Number of correct instances: 75

Question 1 continued on next page...

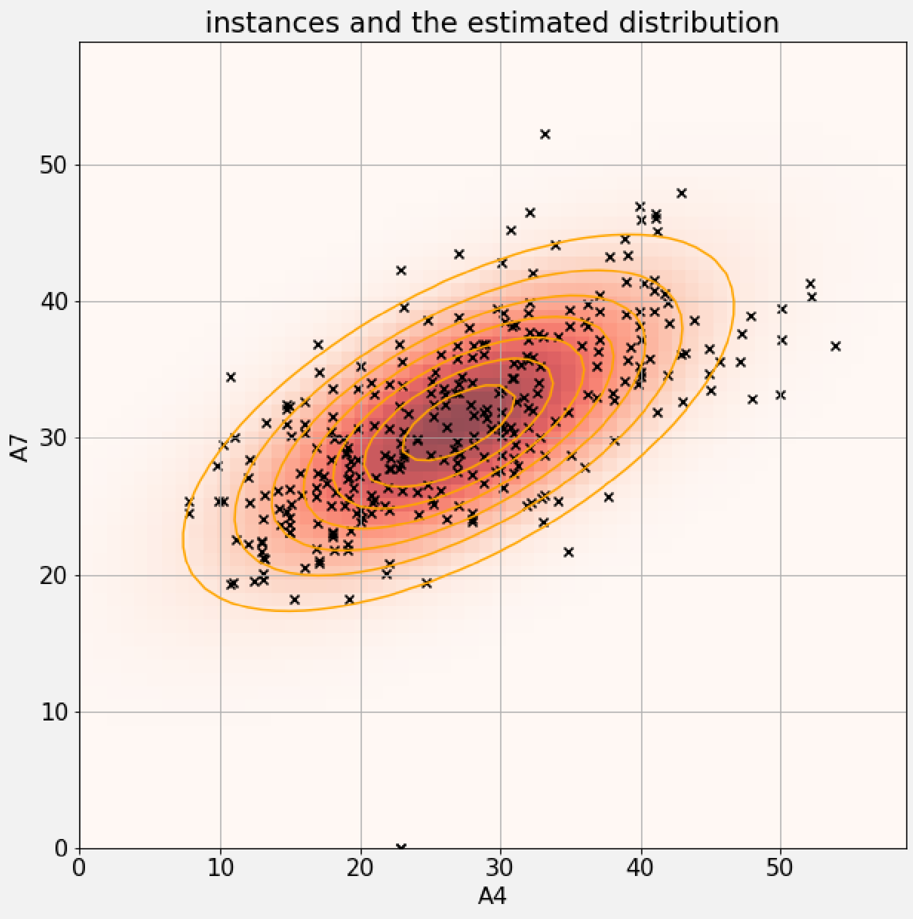
1.9 (5 points) We here consider a two-dimensional (2D) Gaussian distribution for a set of two-dimensional vectors, which we form by selecting a pair of attributes, A4 and A7, in `Xtrn` (NB: not `Xtrn_s`) whose label is 0. To make the distribution of data simpler, we ignore the instances whose A4 value 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 vector: [27.02, 31.09]

Covariance matrix: $\begin{bmatrix} 95.14 & 41.47 \\ 41.47 & 46.69 \end{bmatrix}$



(b)

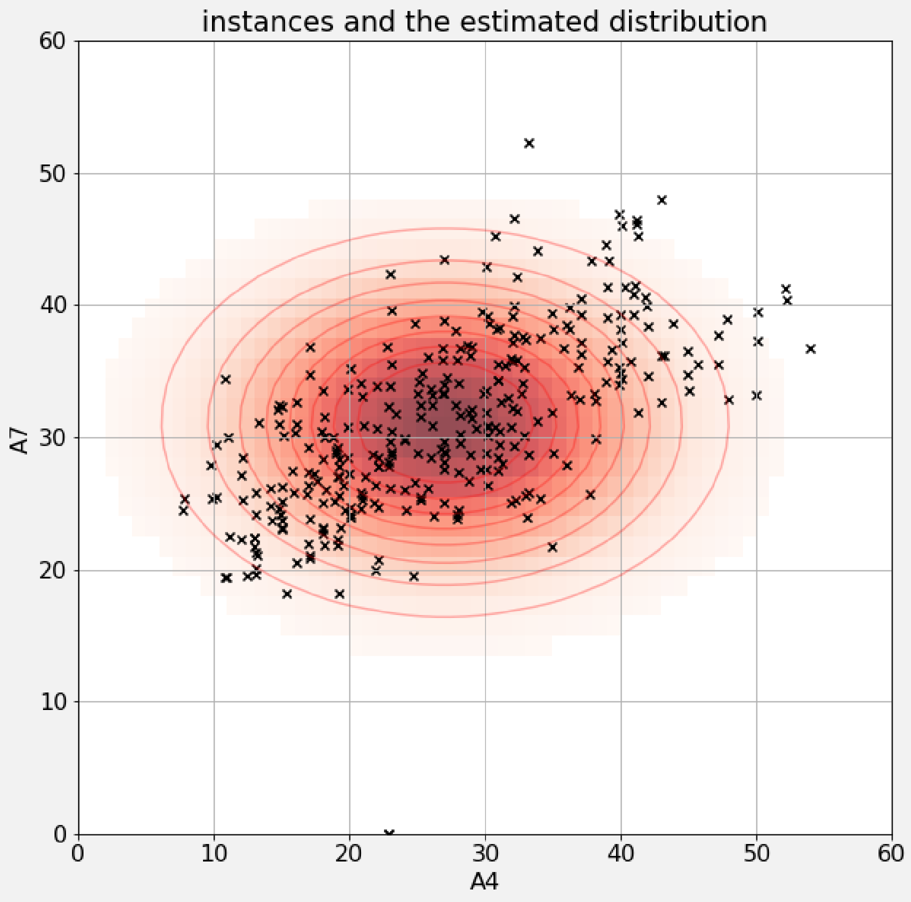
Question 1 continued on next page. . .

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

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

(a) Mean vector: [27.02, 31.09]

Covariance matrix: $\begin{bmatrix} 95.14 & 0 \\ 0 & 46.69 \end{bmatrix}$



(b)

(c) The mean vector of the two graphs are equivalent, suggesting the estimate distribution fits to the correct location. However, since Naive Bayes assumes that the two variables are independent, off diagonal values of covariance matrix is 0, the shape of the contours only stretches horizontally. However, off diagonal values of cov matrix in 1.9a are not zero thus the two variables are correlated and the contours in 1.9b are inclined. Since the scatter plot clearly shows that inclined counters are necessary, the two variables are dependent and Naive Bayes assumption is not satisfied. If Naive Bayes is used, the estimated distribution would be off and the accuracy would be lower.

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

2.1 (5 points)

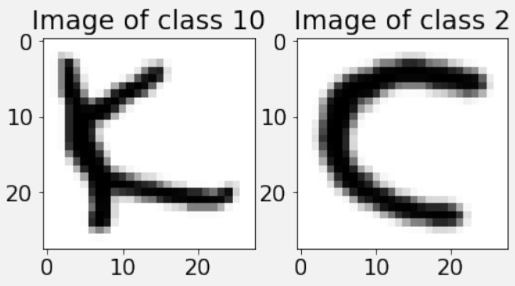
(a) Report (using a table) the minimum, maximum, mean, and standard deviation of pixel values for each `Xtrn` and `Xtst`. (Note that we mean a single value of each of min, max, etc. for each `Xtrn` and `Xtst`.)

(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)

	min	max	mean	std
Xtrn	1.0	0.0	0.177	0.335
Xtst	1.0	0.0	0.176	0.333

(b)



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) Euclidean distance is the same. Since Xtrn_m only subtracted the mean vector, the center of distribution is moved from the mean vector to the origin but everything else remains the same. In the Euclidean distance equation, the mean vector got cancel out when subtracting two elements.

(b) Since we assume Xtrn and Xtst comes from the same distribution, we assume their mean vectors are equivalent. According to central limit theorem, if Xtst is small, its mean vector would be different from that of Xtrn and would not be a good representation of the mean of the real distribution. So we should not use the mean vector of Xtst. However, if we are certain that Xtst comes from a different distribution than Xtrn, such as doing transfer learning, using the mean vector of Xtst is acceptable.

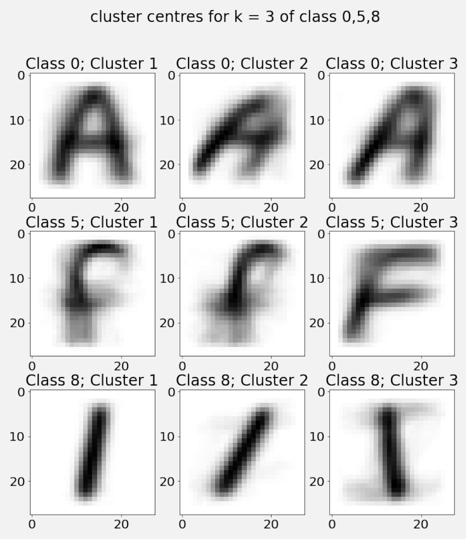
Question 2 continued on next page. . .

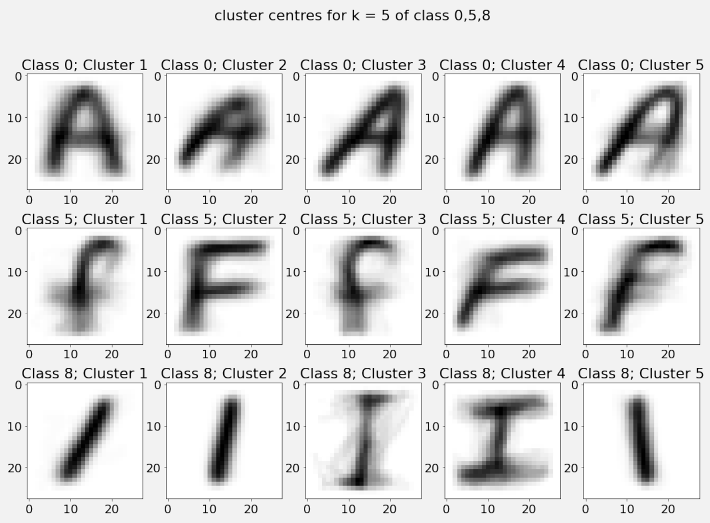
2.3 (7 points) Apply k-means clustering to the instances of each of class $0, 5, 8$ (i.e. 'A', 'F', 'I') in `Xtrn` with $k = 3, 5$, for which use Sklearn's `KMeans` with `n_clusters=`k and `random_state=0` while using default values for the other parameters. Note that you should apply the clustering 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) Cluster centers are more blurry since it is the average of images. That of digit "A" also looks different, meaning the dataset exhibit enough variability and might be a good representation of the real world. The first two centers of "A" in k=5 is very close to that in k=3 but the rests in k=5 shows variability. So centers more than k=3 is needed to efficiently captures this class. However, two types of digit "I", with or without horizontal bars, are shown. Meanwhile, both capitalized and uncapitalized digit "F" are inside our dataset. Such variability within one class would increase the difficulty of classification. Also, the coursework introduction mentions that incorrect labels exist in the dataset, making classification harder and perfect score impossible.

Question 2 continued on next page...

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

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

(a) SSE in k means is defined as the sum of squared distance between samples and its nearest cluster centeroid. With k+1 clusters, each cluster becomes more concentrated since each is assigned less samples (stays the same if same samples being assigned and zero point assigned to the new cluster). With more concentrated clusters, the average distance between samples and corresponding centroids would never increases. With a fixed total number of samples, the SSE would never increases.

Alternatively, if K equals to the number of samples, each cluster centroids is located at the exact position of each sample. So the SSE equals to 0. If K continues to increase, no sample would be assigned to the new clusters so SSE would remains the same. If K decreases, more samples would be assigned to the same cluster and cluster centroid would be drift from the position of sample. So SSE continues to increase. If K = 1, SSE is largest since it represents the distance from all points to a single mean. If we reverse this process, SSE would never increase.

(b) Since in each update step of K means, the new cluster centroid is the mean of all points assigned to that cluster. Since the sum of squared distance between samples and their mean would always be smaller than that of any other point and k means repeats this process until converges: no cluster assignment changes. So before converges, SSE would always decrease and after converges, SSE would remains the same. So SSE would never increase for t+1.

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) Training set Accuracy: 0.916
Test set Accuracy: 0.722

(b)

Top 1 to 5:	L	R	I	K	N
Class numbers	11	17	8	10	13
Number of misclassifications	53	48	42	38	36

(c) For digit "L", more than half of the misclassifications belongs to digit "I" and the reverse is true for digit "I". The model misclassified digit "K" the most to digit "R". For "R", the top two misclassification comes to digit "K" and "A". That of "N" is "H" and "W". So most of the misclassifications have similar structure, like "L" & "I" or "R" & "K". The fact that the model is incapable to capture high level synthetic structure of the letters might cause the misclassifications: similarity in regional features is treated as equivalent of class. Also, in-class-variability and incorrect labels discussed in 2.3b might also be a reason of mistakes.

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) Use K-means to separate capitalized and non-capitalized digits, such as “F” and “f”, to reduce in-class variability

Apply PCA to the data before logistic regression to extract significant dimensions since most elements in the digit vector is zeros.

Fine tune hyper-parameter C by grid search CV to find the best model on this dataset

I will use the second and the third approaches.

(b) Principle Component Analysis (PCA) reduces the dimension of the dataset by transforming a large set of variables into a smaller one while preserving most of the original information. PCA first finds a direction, named “the first principle component”, that maximizes the variance of the dataset. Then it finds the second principle component by finding the direction that both orthogonal to the first principle component and maximizes the remaining variance. PCA repeats such process until the number of principle components equals to the original dimensions. Then, the top principle components that explains most of the data variance — preserves most of the information — are preserved to form a smaller set of variables.

Hyper-parameter C is the inverse of regularization strength , $C = 1/\lambda$, defined by L2 regularization $\lambda \sum_1^n w_i^2$. Since L2 regularization adds the sum of squared parameters into the loss function, it prevents the loss from getting too small and reduce overfitting. The strength of which is controlled by C: the smaller the C value, the greater the effect of L2 regularization.

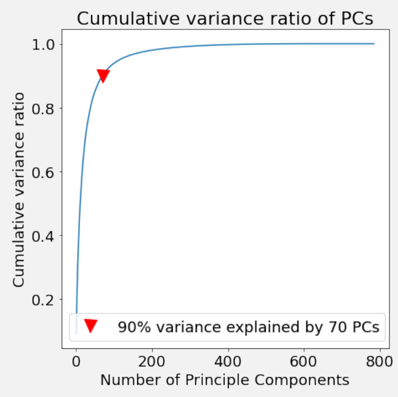
Since the original data contains a lot of zero values, I first apply PCA to the data for dimensionally reduction. By this, I am proposing the assumption that PCA could extract the most important information in the original dataset and the following logistic regression could capture those information efficiently. I also assume that this approach would be more time efficient since training on a reduced dataset would be faster.

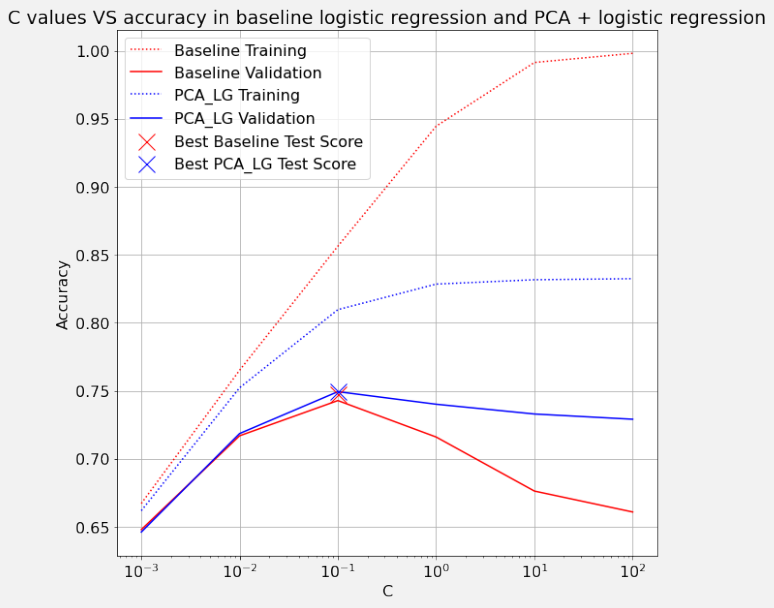
When finding the hyper-parameter C, three fold grid search CV is used. The training data is split into three folds, each is used for validation and the rest is used for training for three iterations. This improves the reliability of the validation score since every instance in the training set is used for validation. After selecting the hyper-parameter that outputs the best validation score, the same setting is used on the test set to further verify the model.

Question 2 continued on next page. . .

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

(c) After applying PCA to the original data, the cumulative variance ratio increases logarithmically as the number of PCs increase. In the left graph bellow, I could discover that 90% of variance could be explained by only the first 70PCs in a total of 784 PCs, meaning we could ignore 91% of the original features but still preserves sufficient information. The test set Xtst_m is also reduced dimension.





Then I set up an experiment to compare the performance of logistic regression with and without PCA for preprocessing. I define the baseline as the model used in 2.5, with 784 input dimensions, and the model "PCA_LG" as the logistic regression model with PCA as preprocessing, preserving the top 70 PCs. By controlling every other variables the same, max iteration = 1000, penalty = L2, solver = lbfgs, and random state = 0, I apply three fold grid search cross validation with C = [0.001, 0.01, 0.1, 1, 10, 100] on the two models.

As in the right graph above, both model exhibit more severe overfitting when the C value increase. This is expected since the larger the C, the weaker the L2 penalty. The best validation scores of both model are achieved when C = 0.1. Meanwhile, the validation accuracy of PCA_LG is greater than that of the baseline model for C >= 0.01. This might caused by the fact that PCA successfully extracts the most important information for the classification task. Meanwhile, with larger C, overfitting is less obvious in PCA_LG model than the baseline since PCA_LG model have far less parameters than the baseline.

The training time of the best PCA_LG is 31.9% of that of the best baseline model. This correspond with my initial assumption that using PCA as pre-processing could be more time efficient. The test set accuracy measured on Xtst_m for the best hyper-parameter, C = 0.1, in the baseline model is 74.73% and that of PCA_LG is 74.96%. Although the result for PCA_LG is slightly higher, I could only conclude that the two models behaves similarly under best hyper-parameter because of the small size of the test set. But it is a surprise to conclude that PCA_LG could achieve comparable result by adopting only 8.9% of the original features.

	Test set Accuracy	Best C
Baseline Logistic Regression:	74.73%	0.1
PCA + Logistic Regression (PCA_LG):	74.96%	0.1

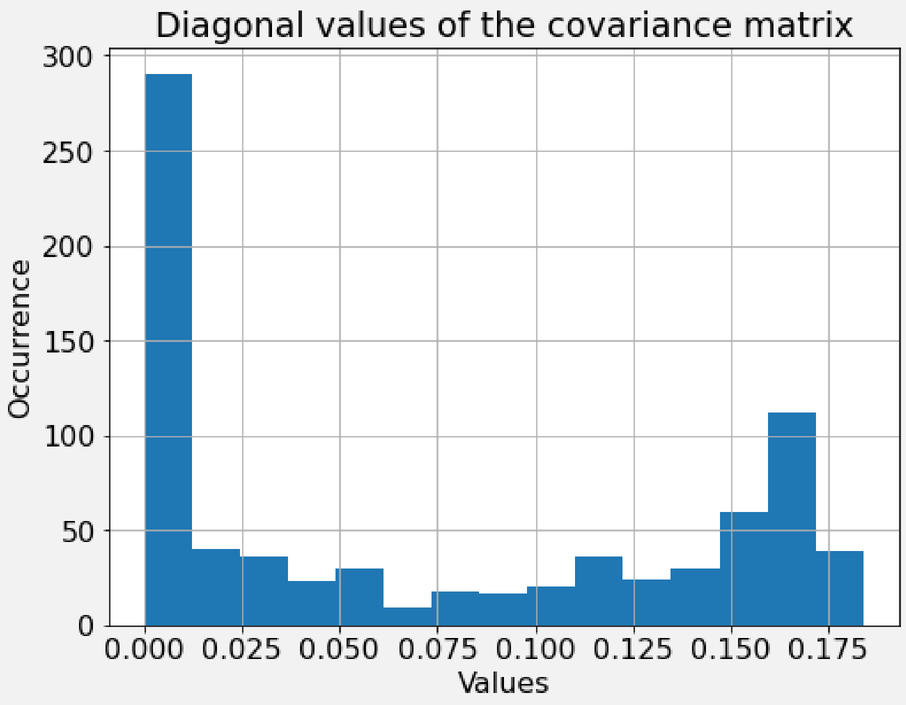
Question 2 continued on next page...

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: 0.1838
Mean: 0.0017

(b) Minimum: 0.0
Maximum: 0.1838
Mean: 0.0723

(c) 

(*continued from the previous page for Q*)

(d) Error message is [LinAlgError: singular matrix]

The covariance matrix is singular, I.e. not invertible. This means there is linear dependence among the variables. So the covariance matrix does not achieve full rank. According to the values and graph above, the diagonal of covariance matrix have a lot of zero values. Since the data contains images and they have white background, variables representing background are zeros, so they are identical and linear dependent.

(e) Add a little bit of noise to the data to make variables not identical

Add a small damping coefficient c>0 to the diagonal of the covariance matrix

Apply PCA to shrink the dimension of the data to avoid dependent variables

Question 2 continued on next page...

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

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

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

(c) Briefly comment on the result you obtained.

(a) log-likelihood of the first instance of class 0 in the test set: -838252.18

(This is what required according to the correction on the Learn page)

Gaussian Mixture generates distribution and estimates covariance matrix on a iterative basis. At start, the real covariance matrix of the data is unknown. GMM first generates a distribution according to the prior (or randomly) and adjust this distribution to approach the dataset step by step until convergence. So the covariance matrix of the final distribution is modified from a prior (or random) iteratively. Since this cov matrix could only approximate that of original distribution, it is different from the one in 2.7, achieved by direct calculation. So this new covariance matrix is not singular and the value could be returned this time.

(b) Accuracy for TRAINING set: 1.0
Correctly classified instances in TRAINING set: 7800
Accuracy for TEST set: 0.693
Correctly classified instances in TEST set: 1803

(c) The model is experiencing significant overfitting since the training accuracy is 100% and test accuracy is lower. The test accuracy is also lower compare to the logistic regression baseline above – by about 0.05. If methods that are designed to fight overfitting could be explored and more hyper-tuning being possible, the GMM model might achieve a good score.

Question 2 continued on next page...

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

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

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

(a) Maximum Likelihood Estimation requires to calculate partial derivative of the likelihood with respect to mean and covariance. But this process would stuck under the likelihood of GMM: latent variables are present in GMM and mean and covariance are unknown in mixture models. Since MLE assumes all variables being present, it cannot be applied to GMM since the latent variables are unknown. On the other hand, Expectation Maximization estimates the latent variables by producing parameters for GMM on every iteration.

(b) When executing parameter estimation, k-means tries to minimize Euclidean distance, which only consider the mean to update the centroid. So the resulting clusters could only be spherical. Expectation Maximization not only take the mean into account, but it also bring the covariance into consideration on every iteration. So EM could adjust to fit elliptical cluster when converging. Although the MLE step following k-means assign a probability distribution and covariance matrix to the cluster defined by k-means, softens the boundaries, the shape of the cluster could not be adjusted. So k-means + MLE could only model spherical cluster. Its off diagonal values of the covariance matrix would all be zeros and diagonal values are equivalent. EM, on the other hand, could spot dependences between variables and have full covariance matrix.

Furthermore, while doing the k-means step, the boundaries are hard so each cluster only consider the instances belong to them when calculating the new centroid. It is after the MLE step where each cluster could assign a weight to all instances in the dataset. The EM algorithm, however, assign weight to every instances in each step from start to finish to estimate the parameters of distribution.

Meanwhile, when fitting two distributions with the same mean but different variance, e.g. one with large contour and one with contour close to its mean. EM could fit the correct distributions since it have soft boundaries. But k-means have hard boundaries so it have no other choice but to split the two distributions in half. The following MLE could not reverse this process. So in this case, k-means + MLE could not successfully fit those specific distributions.