Part 1: Basic questions

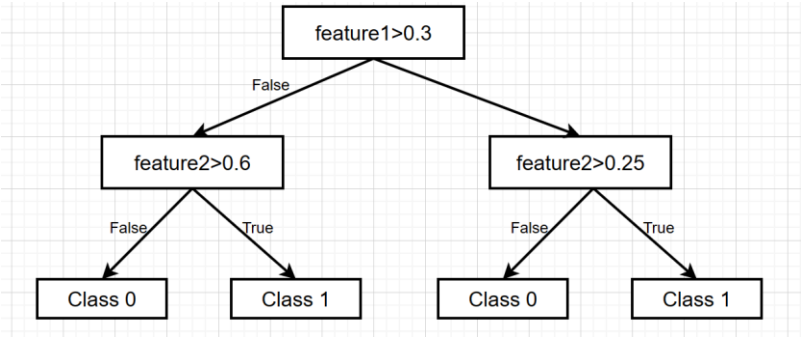
Question 1 of 12: Classification of electrocardiogram signals (10points)

We are supposed to consider two solutions to solve this problem, one is feature-based solution and the other is feature-free solution.

a) As for feature-based solution, in EGGs data set, we can extract many features and data may be multidimensional. RandomForestClassifier or GradientBoostingClassifier is more suitable.
Firstly, we need to do data pre-processing. If there is a significant imbalance in the data, we can apply oversampling or undersampling to obtain a more balanced training set.
Secondly, we need to assign labels to different features. Through medical articles or other methods, we can label each ECG signal as normal or abnormal. We can select the features by brute force. Besides, we can use sequential forward selection to get the appropriate features. Then, we can normalize the data and trying to see if feature selection (e.g. SelectKBest) improves the quality of the model.
Then, we train the model. We divide the data into train set, test set and validation set and we can use cross-validation during development. We can apply machine learning models such as RandomForestClassifier or GradientBoostingClassifier to find which performs best. After selecting the model, we can tune the hyperparameters (e.g., max_depth and ensemble size) of the model. By using such as GridSearchCV, run a cross-validation separately for each hyperparameter values and select the value that provides the highest accuracy in the cross-validation. Besides, we can plot ROC curves based on the change in different thresholds, and select them with higher true positive rate at a lower false positive rate, which has higher AUC value.
Finally, we can use the test set to check the performance of the model, for example we can plot the confusion matrix, accuracy score for evaluation. We can also use sklearn.metrics.classification_report to get the specific performance of the model, e.g. F1 score, precision, etc. If the model meets the requirements and expectations, we can apply this model to practice.
(b). As for feature-free solution, we can use convolutional neural networks to solve this problem and we do not need to do feature preprocessing.
First, we converted the digital signal into a visual image. We give labels (normal or abnormal) to the collected ECG signal dataset, then normalize the dataset (e.g., set the pixel values are between 0 and 1). Then we divide it into training set, test set and validation set. Preprocess the ECG signals and make sure they are all of equal length.
Next, we design the architecture of the model. We will use two-dimensional convolutional and pooling layers, followed dense layers to compute the output. In the last layer, we use sigmoid with 2 outputs(normal or abnormal). Besides, we need to choose the appropriate activation functions (e.g., ReLU) and regularization. In order to prevent overfitting, we add Dropout layer. The model also needs to choose an appropriate loss function, such as *binary_crossentropy*, to calculate the error between the predicted and true labels during training. To train the model efficiently, we can consider different approaches to optimization (e.g., Adam or SGD) to minimize the loss function. We can compare our CNN model with pre-trained model (e.g., based on ImageNet) to validate if the model is effective in a particular application.
Finally, we train our model. After designing the model's structure, we can use the test set to evaluate the performance of the model, and we can compute such as F1 score to compare the performance between different hyperparameter (e.g. Learning Rate Schedule).

Question 2 of 12: Decision trees (5 points)

a. feature1 > 0.3,
False→ feature2 = 0.6
If feature2 > 0.6,
Return Class 1
else, return Class 0
True → feature2 = 0.25
If feature2 > 0.25,
return Class 1
else, return Class 0
I draw a chart as follow



b. The decision boundary can divide the feature space into four regions, but it contains only two classes. At the first depth of the tree, the decision boundary divides the feature space into two regions, by using one of the features. At the next depth of the tree, the decision boundary divides the first partition, which produces two more regions. We can get the decision boundary similar to Q2.1, like a step. Another special case is the rectangle formed in the corner of the image, which divides the class 0 and the other regions are class 1.
c. I think setting the maximum allowed depth to 2 will not suffer from overfitting. By limiting the depth of the tree to 2, the decision boundary is simpler. A model with a depth of 2 can fit the data in fewer ways and is unlikely to be affected by noise in the data, which reduces the risk of overfitting. However, in some special data set, when the train set has more noise, it will also produce the overfitting.
d. I think updating one of the features from bytes to megabytes may not have a significant impact on the performance of the decision tree model. The impurity of the sample set is calculated by the Gini coefficient in the decision tree algorithm. The smaller the Gini coefficient is, which can produce the purer the sample set, and the better the classification result. The Gini coefficient is calculated by probability which is independent of the size of the feature value units. So, changing the feature value to megabytes will have little effect on the performance of the decision tree. However, when we change file size from bytes to megabytes, the scaled feature values affect the tree segmentation and the weight of the features. So, we need to normalize them during the data preprocessing such as divide other features value in the file size by 10^6.

Question 3 of 12: Problems. . . (4 points)

a. In the article, these datasets are stitched together from multiple sources and may contain duplicate items. Also, the sources of some datasets were confused in the model, which may have led the researchers to confuse important features of the model training.
In the Xiaopeng car, the driver was mistakenly judged by Autopilot to be driving with the driver's eyes closed due to small eyes, which deducted Smart Driving points for driving while fatigued. The eye track system working principle is that it captures the driver's facial features through a single vision in order to make a judgment. The driver's facial features are used to classify the degree of eye opening and closing, and the degree of mouth opening and closing, so that the driver's fatigue and distracted driving can be judged. However, the awkwardness of the system is that due to differences in driver's eye size, height, and facial undulations, the system may be misjudged.
This is due to the poor data quality of the enterprise development tools. The training set is too small or biased towards certain examples, the model may not generalize well to new data and may suffer from overfitting. These data are used to train the model by the characteristics of most drivers, but the behavior of drivers with special characteristics is not well predicted, which causes misclassification.
Another example is the misjudgment of Tesla Autopilot. In strong sunlight conditions, both the driver and Autopilot failed to notice the white body of the tractor-trailer and therefore failed to activate the braking system in time, which caused the crash. The main reason was that the radar accurately measured a huge obstacle ahead, but because of the reflective area of the truck and its overly tall body, it looked more like a traffic sign, bridge or elevated road hanging over the road from the millimeter wave radar's perspective and was ignored. During the development of the model, the training data may be affected by the weather conditions during the training process, which may lead to adding additional data and causing incorrect recognition. This is where there is a possibility of mislabeling features in the data, resulting in data imbalance and confusion of some important features, thus identifying white trucks as clouds.

Question 4 of 12: Neural network classification and regression (4points)

a. (1). Dropout(p=1.0). Dropout works by making the activation value of a certain neuron stop working with a certain probability p during forward propagation, which makes the model more generalizable because it does not rely too much on certain features. When p=1, which means that the dropout layer eliminates all the neurons during the training process. Therefore, the probability of dropout should be set to a value less than 1.0, e.g., p = 0.5.
(2). SoftMax output. Our task is binary image classification, so there are two outputs.

b. (1). Softmax output. Softmax function is more suitable as an activation function for multi-classification models, however, our model is a nonlinear regression model. Therefore, we need to remove the activation function.
(2). There is one output instead of 4. But in the hidden layer, we need to set the activation function(e.g. ReLU) for the nonlinear regression.

Question 5 of 12: Hyperparameter tuning in a gradient boosting classifier (4 points)

a. When the maximum depth increases from 1 to 4, the cross-validation score increases aand model becomes more complex and fit the training data, which means it is underfitting. However, as the maximum depth continues to increase, the cross-validation score decreases gradually. This is because the model becomes overfitting. From the figure we can see that the highest accuracy

appears in when the max_depth = 4, which indicates that the optimal decision tree depth for this dataset is 4, and the model can capture the underlying patterns and relationships in the data well using a decision tree with a depth of 4. After that, when we continue increase the depth, but it does not improve the performance of the model and the model become overfitting, so the cross-validation score decreases.

b. When n_estimators increasing, the training accuracy increases. From the figure we can see that the cross-validation score appears to remain constant between 10 and 100, which means it reach the maximum value and the model has reached its limit to fit the training data. When the n_estimators continue to increase, the cross-validation score begins to decrease. At this point the model is overfitted to the training data and the accuracy remains constant at n=1000. With the continued increase in the value of n_estimators, the accuracy gradually decreases and the level of overfitting becomes higher, which leads to poor performance of the model.

Question 6 of 12: K-means clustering (5 points)

a. The detailed convergence steps are as follows

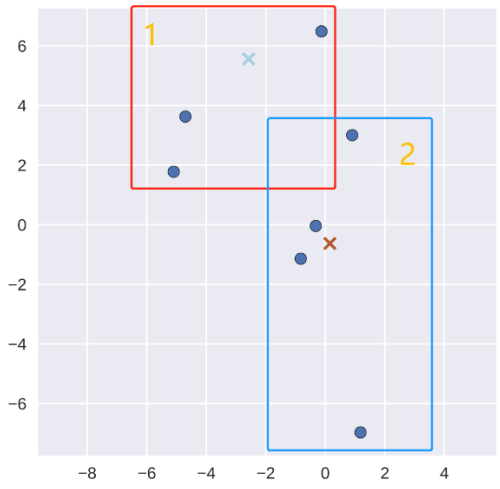


Figure 6.1 Initial image

First, I divided the seven points into two categories which depend on the distance, as shown in the Figure 6.1. List the coordinates of their points in the table 6.1. I calculated their clustering centers by calculating the average value between each coordinate point.

1.	(-0.2, 6.4)
	(-4.8, 3.8)
	(-5.2, 1.8)
2.	(0.8, 3)
	(-0.3, 0)
	(-0.8, -1.2)
	(1.2, -7)

Table 6.1 Point coordinates

$$O_{\text{center},1} = \left(\frac{-0.2 - 4.8 - 5.2}{3}, \frac{6.4 + 3.8 + 1.8}{3}\right) = (-3.4, 4)$$

$$O_{\text{center},2} = \left(\frac{0.8 - 0.3 - 0.8 + 1.2}{4}, \frac{3 + 0 - 1.2 - 7}{4}\right) = (0.225, -1.3)$$

Therefore, the clustering centers after the first iteration are shown in the figure below. Then, I performed clustering and divided it into two parts. Repeat the above steps for the second iteration.

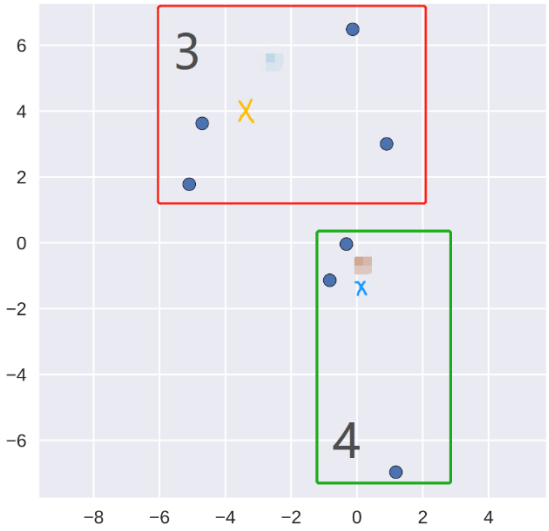


Figure 6.2 Clustering center and classification after the first iteration

3.	(-0.2, 6.4)
	(-5.2, 1.8)
	(0.8, 3)
	(-4.8, 3.8)
4.	(-0.3, 0)
	(-0.8, -1.2)
	(1.2, -7)

Table 6.2 Point coordinates after the first iteration

$$O_{\text{center},3} = \left(\frac{-0.2 - 5.2 + 0.8 - 4.8}{4}, \frac{6.4 + 1.8 + 3 + 3.8}{4}\right) = (-2.35, 3.75)$$

$$O_{\text{center},4} = \left(\frac{-0.3 - 0.8 + 1.2}{3}, \frac{0 - 1.2 - 7}{3}\right) = (0.03, -2.73)$$

The new clustering center is shown in the figure below:

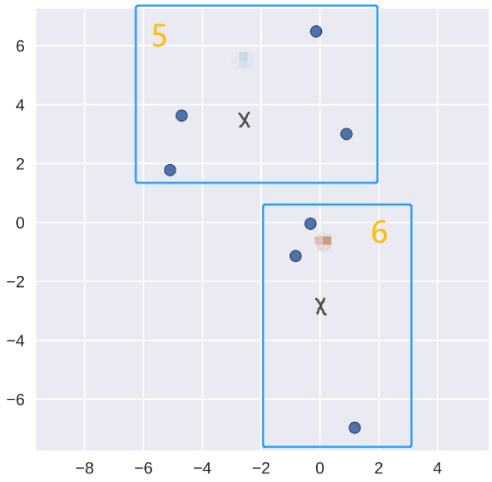


Figure 6.3 Clustering center and classification after the second iteration

It can be seen that the two clustering centers found after two iterations can distinguish the two classes well.

b. We can choose centroids in several ways. We can calculate the sum of squares of the errors of the data, then corresponding points of the original data and select the smallest value as the final choice. Besides, we can run the algorithm several times and pick the best clustering. In addition, we can use the KMeans++ algorithm to make the initial centroids as far as possible from each other and select the final point after several iterations.

Part 2: Questions for the high grades

Question 7 of 12: Convolutional neural networks for image classification (8 points)

a. [[0,0,0], [0,0,0], [0,0,0]]
[[0,0,0], [0,0,0], [0,0,0]]
[[1,1,1], [1,1,1], [1,1,1]]
These weights are a 3x3x3 matrix, where the first dimension correspond to the Red and second is Green of the filter, the third dimension corresponds to the Blue values. Because we want to activate blue patches of the image, so we set the weights of the red and green channels to 0 while the weights of the blue channel to 1.

b. When the filter is applied to the blue parts of the input image, it will highlight the blue part of the image. When the filter is applied to other parts of the input image, it may transfer them to black and not highlight them.

c. The blue area in the figure will be more significant, and the spatial resolution will be reduced. The 2x2 max pooling layer will emphasize the maximum value in the blue channel at each position. At the same time, it reduces the complexity of subsequent layers in the network.

d. The steps are as follows:
(1). Add Flatten layer, output flattened to a vector of length is 1600 (10×10×16)

(2). Add a Dense layer and ReLU activation function
(3). Add a Dropout layer (p = 0.5)
(4). Add Dense layer and ReLU activation function
(5). Add the output layer with 5 outputs and a Softmax activation function.

Question 8 of 12: Art recommendation (4 points)

a. Because we have user rating on visual art works, I decide to build a recommendation system by user-based collaborative filtering. First, we need to normalize the feature values for each artwork like select the special lines in artworks. We build a matrix decomposition model. We represent the users' ratings as a matrix R, the matrix represents users (rows) and artworks features (columns), and the cells the ratings. Then, decompose the user-item matrix R into two lower-dimensional matrices M and N, where M represents user preferences and N represents artworks features. In our data, there may be some missing cells, so we need to predict them. We can use $\hat{r}_{ui} = p_u \cdot q_i$, where p_u is the user's vector and q_i is the item's vector to calculate them and full fill the matrix. We also can calculate the similarity between users using the Jaccard formula or the cosine similarity. Then, we use this matrix to calculate the similarity between different users. After that, we can identify users who are similar to the target user and recommend artworks that are highly rated by similar users while the target user does not see it before.
Finally, we train the model. By matrix factorization method, we minimize the loss and add regularizer as following function: $\sum_{u,i \in M}(r_{ui} - p_u \cdot q_i)^2 + \lambda(||p_u||^2 + ||q_i||^2)$ to get user vectors and item vectors. Then, we use SDG to minimize the difference between the predicted ratings and the actual ratings in the training data. We obtain a smaller $\hat{r}_{ui}$ by reducing the gradient and make recommendations.
We make the following assumptions:
(1) If a user is likely to rate a product highly, they are more likely to engage with it.
(2) Assume we know something about the visual art works such as special features.
(3) Each user has their own linear rating function.

Question 9 of 12: Probabilities in decision tree classifiers (6 points)

a. When setting the max_depth hyperparameter to "None" may lead to overfitting, resulting in a smaller probability. This causes the decision tree to be explored until it can no longer be classified, with the estimated probability of that last point being 1 for its category and 0 for all other categories. At the end of the classification, there is only one sample per child node, which does not reflect the true category probability in the region of the feature space represented by that leaf node.
b. I think the probabilities are poor means the model is overfitting and the probability distribution does not match the actual situation. At this point there is only one feature left for model classification. These leaf nodes have a very low sample size or even only one feature value, which leads to lower probability estimation and poor generalization ability. This can cause worst predictions on new data.
It not to be poor means it conforms to the probability distribution, and the predicted probability fits the probability generated by the classification. In decision tree classifier, the probability of the model output can be considered reliable if the leaf nodes have sufficient sample size and enough

features to make accurate predictions, which means that the model generalizes well to new data and the predictions can be more accurate.

c. In medical diagnosis, such as the determination of breast cancer. When we diagnose patients by machine learning models, it is important to have meaningful probabilities to indicate the that a patient has a certain disease or condition. If there are poor probabilities, because of overfitting we could not predict whether the patient is sick or not, which will cause the miss judgement.

Question 10 of 12: Perceptron learning (4 points)

a. By analyzing perceptron algorithm, I found that the values of y in the formula are 1 and -1. When positive instance is misclassified, which means it is predicted to be negative. At that time, $y \cdot w \cdot x \leq 0, w = w - \eta \cdot y \cdot x$. When η is larger, w changes significantly, which means that the weights change relatively large, which causes the network fails to converge and jumps directly over the lowest place to the other side of the regression line, thus neglecting to find the optimal value.
When η is small, the change of w is smaller, which means that the change of weights is smaller and the network converges very slowly, which will increase the time to find the optimal value. There may be a situation where the local extrema converge and we cannot find the real optimal solution.
As the learning rate increases, the loss decreases first and then increases. When η is larger, The accuracy gradually increases and oscillates around a certain value. When η is small, The model may reach the maximum value of accuracy, but depending on the number of iterations, it may or may not reach it.

Question 11 of 12: Neural network regression (8 points)

a. We can use this model to make predictions about the gains of financial products. Our inputs are macroeconomic indicators, data on financial products, etc. We train these data on this neural network to get the standard deviation of the expected return and stock market return for predicting a specific input. In a standard NN regressor, the output is a single mean value, which cannot provide us the information about how discrete the sample data are and whether they conform to the normal distribution pattern.
b. When we train a MLPRegressor, we typically use the mean squared error as loss function. The equations are as follows:

$$loss_{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

There are some different between these two-loss function. In MSE loss function, it calculates the mean of the squared differences, which is more heavily penalized because it penalizes by squaring the errors, this means that it is sensitive to outliers. However, negative log-likelihood calculates normal distribution probability density function. When x is small, the penalty is larger and severe for probabilities, which are farther from the confidence interval while less sensitive to outliers. The similar is that both these two functions can use gradient-based optimization.
c. I think it would be. The loss function in the model is related to the probability. In classification task, it can predict the probability of which class the item belongs to better.

d. We use exp activation here because we want to make sure that the predicted values are always positive.
I do not think so. If we remove the exp, some values would be negative.

e.

Input: a list of examples features vectors **x**

a list of reference outputs **y**

learning rate **lr**

number of epochs **n**

batch size **bz**

counter **t = 1**

initialize the parameters $\boldsymbol{W_i, b_i, w_\mu, b_\mu, w_\sigma, b_\sigma}$

Split the training data X, Y into batches of bz

#Forward

For each epoch (1 to n):

For each batch in (X, Y):

t = t +1

$$\boldsymbol{a} = \boldsymbol{W_i} \cdot \boldsymbol{x} + \boldsymbol{b_i}$$

$$\boldsymbol{h} = \boldsymbol{ReLU}(\boldsymbol{a})$$

$$\mu = \boldsymbol{w_\mu} \cdot \boldsymbol{h} + b_\mu$$

$$\sigma = exp(\boldsymbol{w_\sigma} \cdot \boldsymbol{h} + b_\sigma)$$

#Compute Loss

$$\boldsymbol{Loss} = log\sqrt{2\pi} + log\sigma + \frac{1}{2}(\frac{y-\mu}{\sigma})^2$$

$$Loss_t = Loss_t + \boldsymbol{Loss}$$

#Backward

$$\boldsymbol{Backward}(\boldsymbol{Loss_t}, \boldsymbol{p})$$

#Update initialize the parameters

$$\boldsymbol{W_i} = \boldsymbol{W_i} - \boldsymbol{W_i} \cdot \boldsymbol{Backward}(\boldsymbol{Loss_t}, \boldsymbol{p}) \cdot \mathbf{lr}$$

$$\boldsymbol{b_i} = \boldsymbol{b_i} - \boldsymbol{b_i} \cdot \boldsymbol{Backward}(\boldsymbol{Loss_t}, \boldsymbol{p}) \cdot \mathbf{lr}$$

$$\boldsymbol{w_\mu} = \boldsymbol{w_\mu} - \boldsymbol{w_\mu} \cdot \boldsymbol{Backward}(\boldsymbol{Loss_t}, \boldsymbol{p}) \cdot \mathbf{lr}$$

$$\boldsymbol{b_\mu} = \boldsymbol{b_\mu} - \boldsymbol{b_\mu} \cdot \boldsymbol{Backward}(\boldsymbol{Loss_t}, \boldsymbol{p}) \cdot \mathbf{lr}$$

$$w_\sigma = w_\sigma - w_\sigma \cdot Backward(Loss_t, p) \cdot \text{lr}$$

$$b_\sigma = b_\sigma - b_\sigma \cdot Backward(Loss_t, p) \cdot \text{lr}$$

#Average Loss

$$Loss_{ave} = \frac{sum\ Loss}{length\ of\ the\ Loss}$$

Return New parameter: $W_i, b_i, w_\mu, b_\mu, w_\sigma, b_\sigma$

Question 12 of 12: Bug routing (6 points)

a. First, we preprocess the data. We remove redundant information such as punctuation from the bug report, convert uppercase letters to lowercase letters, and complete the abbreviations, etc. Then we use TfidfVectorizer to set stop words, mark the text as words and phrases, then convert them into numeric representations.
Then, we want to select relevant features, such as e.g., date of discovery, memory consumption, timing measurements, etc. We can group similar bug reports by clustering. After data balancing, we divide the data set into train set, validation set and test set.
Then, we select the model. When we developing this system, we should consider that different departments have different tasks and we need to train a hierarchical classification model, where each label corresponds to a specific sector in the hierarchy. We can assign different tasks to different departments through keyword search. So, we can choose *Decision Trees*, *RandomForestClassifier*, or *GradientBoostingClassifier*, these models can handle multi-class where each node corresponds to a specific level of the hierarchy and add Local Classifier per Node.
We evaluate these models by cross-validation, and choose the model with highest score. Then, we do some hyperparameter optimization (e.g. by *GridSearchCV*). During hyperparameter tuning period, we can choose to fine-tune the number of trees or layers, the learning rate, the regularization strength, etc. Then we compare different parameters' *accuracy_score* to decide which parameters we can choose. Also, we can plot the confusion matrix, calculate their F1 score and compare with Trivial baseline to evaluate the performance of the model. What is more, we can plot the ROC curve and calculate the AUG value to get the better hyperparameters.
Finally, we validate the model by the validation set. We can calculate hierarchical classification metrics, such as hierarchical precision, F1 score, and tree distance to make judgments about the performance of the model.

b. We can classify the images. In a cross-border e-commerce platform, sellers upload products' images and machine learning classifies products based on their color, material, and size. In the layered structure, the color can be divided into sub-labels such as red, yellow and blue, and the size can be divided into different sub-labels according to the height of the customer.