Note: **Bonus Questions** are completely optional. But they will be used in contributing towards your 25% of homework scores. For example, if you receive 10 bonus points, then this converts to $10 \times 0.25 = 2.5$ points to your overall score. So if you want to do a bit extra work to improve your grades, this is a chance. You can ask questions about bonus questions, but please note that TA may provide less help to questions regarding these bonus questions because these are out of their normal workload, and the questions are a bit more challenging and out of the required scope of this class. You are expected to figure out the bonus questions more independently if you like to try. On the other hand, I do encourage you to try bonus questions because they are actually interesting extensions of what we have learned in this class, and you may find them quite interesting and enhance your understanding and broadening your horizon; and remember, we do give partial credits.

1. **Basic optimization.** (50 points.)

   The background of logistic regression will be discussed in the next lecture. Here, we just focus on finding out the property of the optimization problem, related to training a logistic regression.

   Consider a simplified logistic regression problem. Given n training samples (x_i, y_i), $i = 1, \ldots, n$. The data $x_i \in \mathbb{R}$, and $y_i \in \{0, 1\}$. To train/fit a logistic regression model for classification, we solve the following optimization problem, where $\theta \in \mathbb{R}$ is a parameter we aim to find:

   $$\max_{\theta} \ell(\theta), \tag{1}$$

   where the log-likelihood function

   $$\ell(\theta) = \sum_{i=1}^{n} \{-\log(1 + \exp\{-\theta x_i\}) + (y_i - 1)\theta x_i\}.$$

   (a) (15 points) Derive the gradient of the cost function $\ell(\theta)$ in (1) and write a pseudo-code for performing **gradient descent** to find the optimizer θ^*. This is essentially what the training procedure does. pseudo-code means you will write down the steps of the algorithm, not necessarily any specific programming language.

   The gradient of the cost function is:

   $$\nabla \ell(\theta) = \sum_{i=1}^{n} \{(y_i - 1)x_i + \frac{\exp\{-\theta x_i\} x_i}{1 + \exp\{-\theta x_i\}}\}$$

   pseudo code for gradient descent:

   initialize θ^0

while $\|\theta^{t+1} - \theta^t\| > \epsilon$:

calculate $\theta^{t+1} = \theta^t + \gamma \sum_{i=1}^{n} \{(y_i - 1)x_i + \frac{\exp\{-\theta x_i\}x_i}{1+\exp\{-\theta x_i\}}\}$

(b) (15 points) Present a **stochastic gradient descent** algorithm to solve the training of logistic regression problem (1).
pseudo code for stochastic gradient descent:

initialize θ^0

while $\|\theta^{t+1} - \theta^t\| > \epsilon$:

select a small subset sample S_k from the whole data.

calculate $\theta^{t+1} = \theta^t + \gamma \sum_{i \in S_k} \{(y_i - 1)x_i + \frac{\exp\{-\theta x_i\}x_i}{1+\exp\{-\theta x_i\}}\}$

(c) (20 points) We will **show that the training problem in basic logistic regression problem is concave.** Derive the Hessian matrix of $\ell(\theta)$ and based on this, show the training problem (1) is concave. Explain why the problem can be solved efficiently and gradient descent will achieve a unique global optimizer, as we discussed in class.

Since there is only one parameter θ in the cost function, the Hessian matrix is 1x1.
In order to derive the Hessian matrix, we need to get the second order derivative of the cost function

$$\ell(\theta) = \sum_{i=1}^{n} \{-\log(1 + \exp\{-\theta x_i\}) + (y_i - 1)\theta x_i\}.$$

Thus:

$$\frac{\partial^2 l(\theta)}{\partial^2 \theta} = \sum_{i=1}^{n} -\frac{\exp\{\theta x_i\}x_i^2}{(\exp\{\theta x_i\} + 1)^2}$$
$$\frac{\partial^2 l(\theta)}{\partial^2 \theta} = -\sum_{i=1}^{n} \frac{\exp\{\theta x_i\}x_i^2}{(\exp\{\theta x_i\} + 1)^2}$$

Since $\exp\{\theta x_i\}$, x_i^2 and $(\exp\{\theta x_i\}+1)^2 > 0$, the second-order derivative of the cost function $\ell(\theta) < 0$. Thus the cost function $\ell(\theta)$ is concave!
For a concave function, any local optimum is also a global optimum. Thus once gradient descent reaches a local minimum, it means that it also got the global minimum.

2. **Comparing Bayes, logistic, and KNN classifiers.** (50 points)

In lectures we learn three different classifiers. This question is to implement and compare them. We are suggest use Scikit-learn, which is a commonly-used and powerful Python library with various machine learning tools. But you can also use other similar library in other languages of your choice to perform the tasks.

Part One (Divorce classification/prediction). (30 points)

This dataset is about participants who completed the personal information form and a divorce predictors scale.

The data is a modified version of the publicly available at https://archive.ics.uci.edu/ml/datasets/Divorce+Predictors+data+set (by injecting noise so you will not replicate the results on uci website). There are 170 participants and 54 attributes (or predictor variables) that are all real-valued. The dataset **q3.csv**. The last column of the CSV file is label y (1 means "divorce", 0 means "no divorce"). Each column is for one feature (predictor variable), and each row is a sample (participant). A detailed explanation for each feature (predictor variable) can be found at the website link above. Our goal is to build a classifier using training data, such that given a test sample, we can classify (or essentially predict) whether its label is 0 ("no divorce") or 1 ("divorce").

Build three classifiers using (Naive Bayes, Logistic Regression, KNN). Use the first 80% data for training and the remaining 20% for testing. If you use scikit-learn you can use train_test_split to split the dataset.

(a) Report testing accuracy for each of the three classifiers. Comment on their performance: which performs the best and make a guess why they perform the best in this setting.

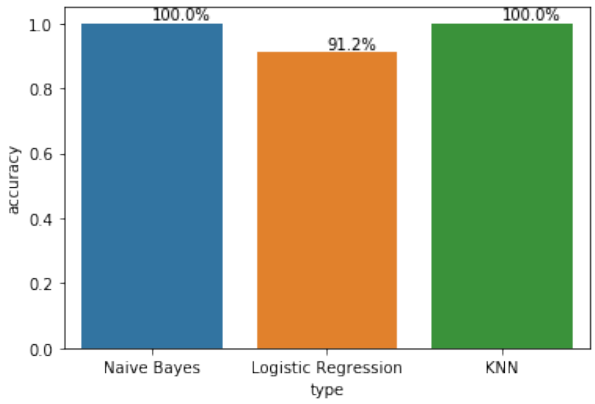


Figure 1: Accuracy by type

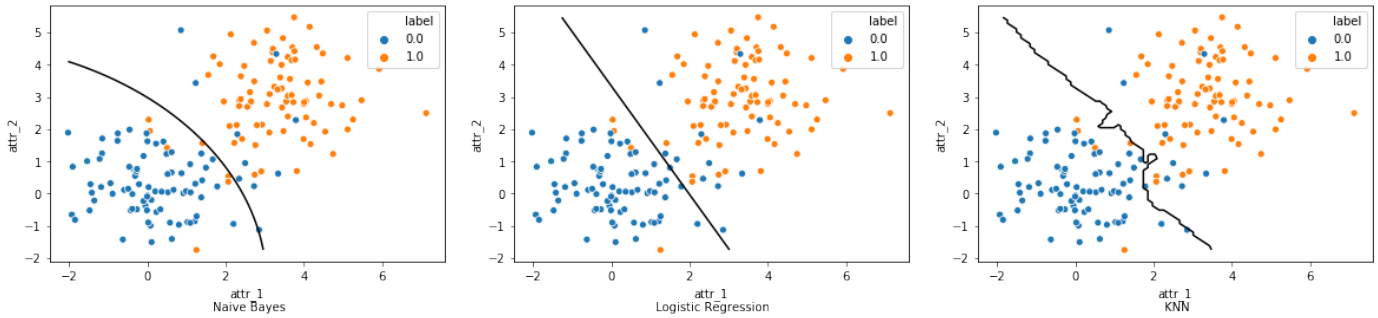
Both Navie Bayes and KNN classifiers perform the best with accuracy at 100%. Logistic Regression only reach a accuracy at 91.2%.

The reason why Naive Bayes performs well could be that the attributes in the models are actually independent which fits well with the assumption of Naive Bayes, also the decision boundary is non-linear to capture complex relationships.

For KNN, the reason why it performs well could be that KNN is robust to noisy in the training data, since we consider multiple neighbors.

The reason why logistic Regression didn't perform as well as the other two could be that logistic regression is not flexible enough to capture complex relationships, because the decision boundary is linear.

(b) Use the first two features to train three new classifiers. Plot the data points and decision boundary of each classifier. Comment on the difference between the decision boundary for the three classifiers. Please clearly represent the data points with different labels using different colors.



(a) Naive Bayes (b) Logistic Regression (c) KNN

Figure 2: Different classifiers

The decision boundary of Naive Bayes is quadratic. The decision boundary of Logistic Regression is linear. The decision boundary of KNN is very flexible.

Part Two (Handwritten digits classification). (20 points) Repeat the above using the **MNIST Data** in our **Homework 3**. Here, give "digit" 6 label $y = 1$, and give "digit" 2 label $y = 0$. All the pixels in each image will be the feature (predictor variables) for that sample (i.e., image). Our goal is to build classifier to such that given a new test sample, we can tell is it a 2 or a 6. Using the first 80% of the samples for training and remaining 20% for testing. Report the classification accuracy on testing data, for each of the three classifiers. Comment on their performance: which performs the best and make a guess why they perform the best in this setting.

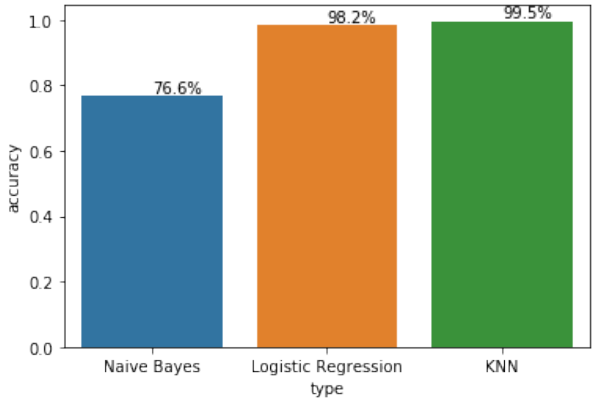


Figure 3: Accuracy by type

Naive Bayes has an accuracy rate of 76.6%. Logistic Regression has an accuracy rate at 98.2%. KNN has an accuracy rate at 99.5%.
Apparently KNN performs the best. The reason why it performs so well could be that KNN is robust to noisy in the training data. All the outliers are spread evenly in the opposite cluster, no aggregation. Thus KNN algorithm won't be tricked by those clustered outliers.

3. **Deriving M-step in EM for GMM.** (Bonus question: 10 points)

Consider the Q function in EM for GMM:

$$Q(\theta|\theta_k) = \sum_{i=1}^{n}\sum_{c=1}^{C} p_{i,c} \log \pi_c + \sum_{i=1}^{n}\sum_{c=1}^{C} p_{i,c} \log \phi(x_i|\mu_c, \Sigma_c)$$

where $\theta = \{\pi_c, \mu_c, \Sigma_c\}_{c=1,\dots,C}$, and $p_{i,c} \propto \pi_c^{(k)} \phi(x_i|\mu_c^{(k)}, \Sigma_c^{(k)})$, $i = 1, \dots, n$, $c = 1, \dots, C$ depend on the parameters from the previous iteration. Here $\phi(x|\mu, \Sigma)$ denotes the pdf of a multi-variate Gaussian with mean vector μ and covariance matrix Σ.

Solve π_c, μ_c and Σ_c that maximize the $Q(\theta|\theta_k)$ function. In other words, we want to find

$$\begin{aligned} \theta_{k+1} := \arg\max_{\theta} \quad & Q(\theta|\theta_k) \\ \text{subject to} \quad & \sum_{c=1}^{C} \pi_c = 1. \end{aligned}$$

Show that the solution θ_{k+1} is given by the following expression

$$\pi_c^{(k+1)} = \frac{\sum_{i=1}^{n} p_{i,c}}{n}$$

$$\mu_c^{(k+1)} = \frac{\sum_{i=1}^{n} p_{i,c} x_i}{\sum_{i=1}^{c} p_{i,c}}$$

$$\Sigma_c^{(k+1)} = \frac{\sum_{i=1}^n p_{i,c}(x_i - \mu_c^{(k)})(x_i - \mu_c^{(k)})^T}{\sum_{i=1}^n p_{i,c}}$$

(Hint: Consider the Lagrangian function for solving this constrained optimization problem. You only need to introduce one Lagrangian multiplier because you only have one constraint. Then solve it from there.)

4. **Naive Bayes for spam filtering**. (Bonus question: 20 points)

   In this problem we will use the Naive Bayes algorithm to fit a spam filter by hand. This will enhance your understanding to Bayes classifier and build intuition.

   Spam filters are used in all email services to classify received emails as "Spam" or "Not Spam". A simple approach involves maintaining a vocabulary of words that commonly occur in "Spam" emails and classifying an email as "Spam" if the number of words from the dictionary that are present in the email is over a certain threshold. We are given the vocabulary consists of 15 words

   $V = \{$secret, offer, low, price, valued, customer, today, dollar, million, sports, is, for, play, healthy, pizza$\}$.

   We will use V_i to represent the ith word in V. As our training dataset, we are also given 3 example spam messages,

   - million dollar offer
   - secret offer today
   - secret is secret

   and 4 example non-spam messages

   - low price for valued customer
   - play secret sports today
   - sports is healthy
   - low price pizza

   Recall that the Naive Bayes classifier assumes the probability of an input $x = [x_1, x_2, \ldots, x_n]^T$ depends on its class y. In our case the input vector x corresponding to each message has length $n = 15$ equal to the number of words in the vocabulary V, where each entry x_i is equal to the number of times word V_i occurs in x.

   (a) Calculate $\mathbb{P}(y = 0)$ and $\mathbb{P}(y = 1)$ from the training data, where $y = 0$ corresponds to spam messages, and $y = 1$ corresponds to non-spam messages.

   (b) List the feature vector x for each spam and non-spam message.

(c) In the Naive Bayes model, the likelihood of a sentence with feature vector x given a class c is

$$\mathbb{P}(x|y=c) = \prod_{k=1}^{n} \theta_{c,k}^{x_k}$$

where $\theta_{c,k} \in (0,1)$ is the weight of word k in class c, which satisfies $\sum_{k=1}^{n} \theta_{c,k} = 1$, $\forall c$. Calculate the maximum likelihood estimates of $\theta_{0,1}$, $\theta_{0,7}$, $\theta_{1,1}$, $\theta_{1,15}$ by maximizing $\mathbb{P}(x|y=c)$ with respect to $\theta_{c,k}$ and given data. (Hint: Consider the Lagrangian function for solving this constrained optimization problem. You only need to introduce one Lagrangian multiplier because you only have one constraint. Consider log-likelihood (i.e., taking log of the cost function). Then solve it from there.)

(d) Given a new message "today is secret", decide whether it is spam or not spam, based on the Naive Bayes classifier, learned from the above data.