FIT2086 Lecture 7 Summary
Classification

Dr. Daniel F. Schmidt

October 15, 2019

1 Part I: Logistic Regression

Classifiers. A classifier is a particular type of supervised learning method. As a refresher, the supervised learning problem involves taking $p+1$ measurements on n individuals, and nominating one of the $p+1$ measurements as a target; the remaining p measurements are used to predict this target variable. In Lecture 6 we examined the situation in which the targets were continuous and the relationships between the predictors and the target was linear. This was called linear regression.

In Lecture 7 we examined an alternative setting in which the targets are not continuous, or even numerical; rather they are categorical, i.e., they can take on one of K different values. The values are not assumed to be numerical in the sense that we cannot perform usual numerical operations such as addition or multiplication to them. An example might be education status, with $K = 3$ values: 'High school', 'Tertiary', 'Postgraduate'. Even if these values were coded as 1 for 'High School', 2 for 'Tertiary' and 3 for 'Postgraduate' it is very clear that 'Tertiary' is not equal to twice 'High school', as the values are simple codes and have no numerical meaning.

Formally, a classifier is a supervised learning algorithm that takes the features/predictors of an item or individual and produces a probability of the item belonging to each of the K different classes, i.e.,

$$\mathbb{P}(Y = y \,|\, X_1 = x_1, X_2 = x_2, \ldots, X_p = x_p).$$

That is, a classifier is formally defined as the conditional probability of the target Y given the values of the p explanatory variables/predictors/features $x_1, \ldots, x_p$.

Direct Construction of Classifiers. In the particular case that the predictors $X_1, \ldots, X_p$ are also categorical we can directly construct the conditional probability using the conditional probability rule we learned in Lecture 1:

$$\mathbb{P}(Y = y \,|\, X_1 = x_1, \ldots, X_p = x_p) = \frac{\mathbb{P}(Y = y, X_1 = x_1, \ldots, X_p = x_p)}{\mathbb{P}(X_1 = x_1, \ldots, X_p = x_p)}$$

where $\mathbb{P}(Y = y, X_1 = x_1, \ldots, X_p = x_p)$ is the joint probability of $Y, X_1, \ldots, X_p$, and

$$\mathbb{P}(X_1 = x_1, \ldots, X_p = x_p) = \sum_y \mathbb{P}(Y = y, X_1 = x_1, \ldots, X_p = x_p)$$

is the marginal probability of the predictors $X_1, \ldots, X_p$. So, to build a classifier directly in this manner we need the joint probability distribution of the target and the p features. As an example, consider

	No Heart Disease ($H = 0$)	Heart Disease ($H = 1$)
No Mutation ($M = 0$)	0.35	0.30
Mutation ($M = 1$)	0.10	0.25

Table 1: Population joint probabilities of heart disease/LDLR mutation.

the following (toy) example: we have measured the LDLR mutation status M of an individual ($M = 1$ means presence, $M = 0$ means absence of the genetic mutation) and we want to predict the probability of whether they have heart disease H ($H = 1$ means they heart disease) or not. In this case both M and H are binary random variables, and the complete joint probability distribution is specified by the simple 2×2 table given in Table **??**. Then, we have the following conditional probabilities of having heart disease, given the two possible values the predictor (M) could take:

$$\begin{aligned}\mathbb{P}(H = 1 \,|\, M = 0) &= \frac{\mathbb{P}(H = 1, M = 0)}{\mathbb{P}(M = 0)} = 0.4615 \\ \mathbb{P}(H = 1 \,|\, M = 1) &= \frac{\mathbb{P}(H = 1, M = 1)}{\mathbb{P}(M = 1)} = 0.7143\end{aligned}$$

From this we see that the model predicts a person to be $0.461/0.714 \approx 1.54$ times more likely to have heart disease if they have mutation, than if they do not.

Learning Simple Classifiers. The obvious problem with the approach described in the previous section is that it relies on our knowing the *population* joint probabilities of our targets and predictors. In the toy example, we were told these probabilities – but in reality, we don't know what they are. All we have is a set of data consisting of measurements for n individuals of our target, say h_i and whether or not they had a mutation, say m_i. So how can we proceed? Well, one way we could attack the problem is to *estimate* the population joint probabilities from the sample we have using the empirical, or observed, probabilities (proportions), i.e.,

$$F(H = h, M = m) = \frac{1}{n} \sum_{i=1}^{n} I(h_i = h \text{ and } m_i = m)$$

where $F(H = h, M = m)$ is the empirical joint distribution of H and M and $I(\cdot)$ is the indicator function that returns a one if the condition inside the function is met, and a zero otherwise. The weak law of large numbers guarantees us that these estimates will converge to the true, population probabilities as the sample size $n \to \infty$, so in this sense, this approach is reasonable. Let us return to our toy example; imagine that we have sampled $n = 8$ individuals and measured their heart disease and mutation status, giving us two vectors of data:

$$\mathbf{m} = (1, 1, 0, 1, 1, 1, 0, 0) \text{ and } \mathbf{h} = (1, 0, 1, 1, 0, 0, 1, 0).$$

We can now fill in our table of estimated population probabilities for M and H by counting how many times $m_i = 1$ when $h_i = 1$, and so on; these are shown in Table **??** for this data. Using these

	No Heart Disease ($H = 0$)	Heart Disease ($H = 1$)
No Mutation ($M = 0$)	1/8	2/8
Mutation ($M = 1$)	3/8	2/8

Table 2: Estimated joint probabilities of heart disease/LDLR mutation

probabilities we can construct a classifier; we find that

$$\begin{aligned}
\hat{\mathbb{P}}(H=1\,|\,M=0) &= \frac{\hat{\mathbb{P}}(H=1,M=0)}{\hat{\mathbb{P}}(M=0)} = \frac{2/8}{1/8+2/8} = 0.666 \\
\hat{\mathbb{P}}(H=1\,|\,M=1) &= \frac{\hat{\mathbb{P}}(H=1,M=1)}{\hat{\mathbb{P}}(M=1)} = \frac{2/8}{3/8+2/8} = 0.2
\end{aligned}$$

where we have used $\hat{\mathbb{P}}$ to emphasise the fact that these probabilities are estimates of the population probabilities derived from our sample. In this case the estimated conditional probabilities are not very close to the exact conditional probabilities we would get if we knew the population – that is not unexpected given that our estimates of the joint probabilities were based on only $n = 8$ observations from our population. If we had $n = 500$ we would find that the estimated probabilities would be quite close to the theoretical, population probabilities are our classifier would be quite good.

However, this fact does reveal the fundamental weakness with this approach. While conceptually simple, and theoretically sound for very large n, in practice it does not work well because of the number of samples required to get good estimates for more complex models. In the above toy example we had two binary variables, H and M, which meant there were $2 \times 2 = 4$ different combinations of H and M, and therefore four joint probabilities we needed to estimate. Now imagine we add a predictor; we measure another mutation, and have two mutation variables, M_1 and M_2. Even if these were both binary we would now have $2 \times 2 \times 2 = 8$ different joint probabilities to estimate. More generally, if we have p *binary* predictors, then we have 2^{p+1} different joint probabilities we need to estimate from our sample. This is an example of exponential growth in the number of predictors: even a moderate p will rapidly outstrip our sample size no matter how large it realistically can be.

Classifiers as Conditional Probabilities. The problem is not so much a specific problem with building classifiers, but rather a more general problem with our approach. The previous approach solves a more difficult problem (estimating the joint probabilities) on the way to solving an easier problem (estimating the conditional probabilities). A general maxim in science is that one should never solve a harder problem on the way to solving an easier one. A solution to the exponential growth problem then is to side-step estimating the joint probabilities altogether and focus directly on estimating the quantity of interest, i.e., we should directly model the conditional probabilities:

$$\mathbb{P}(Y=y\,|\,X_1=x_1,\ldots,X_p=x_p) = f(x_1,\ldots,x_p)$$

where $f(\cdot)$ is some function that takes our predictors and produces a conditional probability of our target begin y. This reduces the problem back to the setting of supervised learning that we previously examined, except that now the functional relationship between the predictors and the target is modelling the conditional probability of the target being a specific category.

2 Part II: Logistic Regression

Linear Predictors. We have previously examined linear regressions as supervised learning techniques. Logistic regression adapts linear models to the specific case of binary classification, i.e., when $Y \in \{0,1\}$. Given values of our p predictors, a linear regression predicts a continuous target using a weighted linear combination of the predictors plus an intercept:

$$\mathbb{E}[Y_i] = \beta_0 + \sum_{j=1}^{p} \beta_j x_{i,j} \equiv \eta_i$$

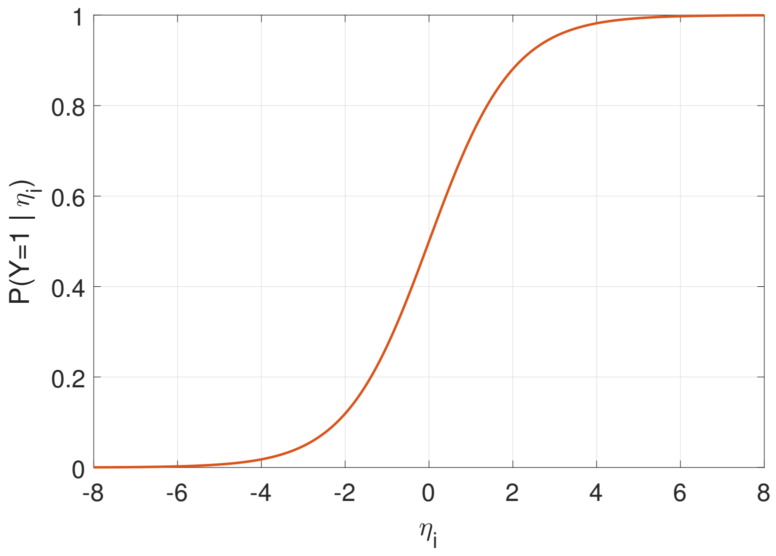

Figure 1: The logistic function. As $\eta_i \to -\infty$, then $\mathbb{P}(Y_i = 1 \,|\, \eta_i) \to 0$, and as $\eta_i \to \infty$, then $\mathbb{P}(Y_i = 1 \,|\, \eta_i) \to 1$.

where "$a \equiv b$" denotes that a is equivalent to b. The symbol η_i is a short-hand for the linear predictor for individual i. A naïve approach to extending linear regression to classification might simply be to ignore the fact that the targets are binary, and use

$$\mathbb{P}(Y_i = 1 \,|\, x_{i,1}, \ldots, x_{i,p}) = \eta_i.$$

That is, we could directly model the conditional probability of $Y_i = 1$ as a linear combination of our predictors plus an intercept, and maybe even fit a model using least-squares. While the model would fit, in the sense that least-sqaures would give us estimates, there is a serious problem: the resulting model could easily lead to values of $\eta_i < 0$ or $\eta_i > 1$ for certain values of the predictors. This is clearly nonsensical.

Logistic Regression. A solution to this problem is to bound the values of $\eta_i \in (0, 1)$. There are a lot of ways to achieve this; one approach is to take the linear predictor η_i and pass it through a "squashing" function of the form:

$$\mathbb{P}(Y_i = 1 \,|\, x_{i,1}, \ldots, x_{i,p}) = \frac{1}{1 + e^{-\eta_i}} \tag{1}$$

The function $g(x) = 1/(1 + e^{-x})$ is called the logistic function, which is where logistic regression gets its unusual name. The logistic function has the property that $g(x) \to 0$ as $x \to -\infty$ and $g(x) \to 1$ as $x \to \infty$, so that it is strictly bounded to lie in $(0, 1)$. It has the nice property of being smooth, in the sense that it is (infinitely) differentiable. Figure **??** demonstrates the logistic function; the key properties are that as η increases the probability of $Y_i = 1$ increases, and that it is bounded to lie in $(0, 1)$.

Logistic Regression and Log-Odds. At first glance it may seem that the choice of logistic function is arbitrary; indeed, we could easily envisage an infinite variety of squashing functions we could use to map our linear predictor η_i to $(0, 1)$. However, the logistic function actually arises naturally from an interpretation of binary classifiers in terms of odds. The odds for $Y = 1$ are defined as

$$O = \mathbb{P}(Y = 1)/\mathbb{P}(Y = 0),$$

which describes *how many times more likely* Y is to take on the value 1 (i.e., a success) than it is to take on a 0 (i.e., a failure). For example, if we toss a coin with success probability of $\theta = 0.75$, then the odds of seeing a head are $0.75/0.25 = 3$, and the odds in favour of seeing a tail are $0.25/0.75 = 1/3$. The odds are a very natural way of describing the likelihood of binary events. If we take the odds O and pass them through the logarithm function we obtain the log-odds $L = \log O$; in some sense the log-odds are even more natural as they render the odds symmetric in the sense that the log-odds in favour of $Y = 1$ are equal to minus the log-odds in favour of $Y = 0$, and vice versa; from our example above, if $\theta = 0.75$ the log-odds in favour of $Y = 1$ are $\log 3$ and the log-odds in favour of $Y = 0$ are $\log 1/3 = -\log 3$. This appealing as what we label as a "success" and as a "failure" is arbitrary. Further more, while the odds must lie in $(0, \infty)$ the log-odds range from $(-\infty, \infty)$ due to the properties of the logarithm.

A logistic regression model is actually formulated by directly modelling the conditional log-odds of success using:

$$\log\left(\frac{\mathbb{P}(Y_i = 1 \,|\, x_{i,1}, \ldots, x_{i,p})}{\mathbb{P}(Y_i = 0 \,|\, x_{i,1}, \ldots, x_{i,p})}\right) = \beta_0 + \sum_{j=1}^{p} \beta_j x_{i,j} \equiv \eta_i,$$

i.e., a logistic regression specifies that the conditional log-odds of success for individual i is equal to a weighted linear combination of their associated predictors, plus an intercept, i.e., η_i. This gives us a very clean interpretation of the coefficients:

- the intercept β_0 is the log-odds of success when all the predictors are zero, i.e., $x_{i,1} = x_{i,2} = \ldots = x_{i,p} = 0$;
- the coefficient β_j is the increase in log-odds per unit change in predictor x_j

Under the logistic regression model, the odds of success for individual i are then equal to e^{η_i}. When $\eta_i > 0$, $e^{\eta_i} > 1$ which implies that $Y_i = 1$ is more likely than $Y_i = 0$; conversely, when $\eta_i < 0$, $e^{\eta_i} < 1$ and $Y = 0$ is more likely than $Y_i = 1$. When $\eta_i = 0$, the likelihood of $Y_i = 1$ is the same as $Y_i = 0$.

To verify that equating the log-odds with the linear predictor does in fact lead to the logistic transformation (**??**) we first note that $\mathbb{P}(Y_i = 0 \,|\, x_{i,1}, \ldots, x_{i,p}) = 1 - \mathbb{P}(Y_i = 1 \,|\, x_{i,1}, \ldots, x_{i,p})$ by the rules of probability, and we can write:

$$\log\left(\frac{\mathbb{P}(Y_i = 1 \,|\, x_{i,1}, \ldots, x_{i,p})}{1 - \mathbb{P}(Y_i = 1 \,|\, x_{i,1}, \ldots, x_{i,p})}\right) = \eta_i$$

If we exponentiate both sides of the equation we obtain

$$\frac{\mathbb{P}(Y_i = 1 \,|\, x_{i,1}, \ldots, x_{i,p})}{1 - \mathbb{P}(Y_i = 1 \,|\, x_{i,1}, \ldots, x_{i,p})} = \exp(\eta_i)$$

and solving this equation for the conditional probability $\mathbb{P}(Y_i = 1 \,|\, \cdots)$ yields

$$\mathbb{P}(Y_i = 1 \,|\, x_{i,1}, \ldots, x_{i,p}) = \frac{\exp(\eta_i)}{1 + \exp(\eta_i)} = \frac{1}{1 + \exp(-\eta_j)}$$

where we used the fact that $1/e^a = e^{-a}$; it is clear that this is the logistic function (**??**), as was previously claimed.

Estimating Logistic Regressions. Of course, like with any supervised learning method, we generally do not know the population values for the coefficients $\boldsymbol{\beta}$ and intercept β_0. Instead, we have a sample of data from our population and we use this to estimate, or learn, a suitable model of the population. A standard approach to doing this is by applying the principle of maximum likelihood (Lecture 3); that is, we choose values of $\boldsymbol{\beta}$ and β_0 that result in our model assigning the largest possible probability to the sample $(y_1, \ldots, y_n)$ we have observed. To do this we need an expression for the likelihood of our sample $(y_1, \ldots, y_n)$ given our predictors. It turns out this is quite straightforward.

The first thing to notice is that if our data is binary, then we can model each observation $y_1, \ldots, y_n$ as if they were realisations of n different Bernoulli random variables $Y_1, \ldots, Y_n$. The difference between a logistic regression and the simple Bernoulli models we have examined so far is that we now model each observation as coming from a different Bernoulli distribution, each with its own probability of success $\theta_i(\cdot)$ determined by the predictors and the coefficients; that is, we say that

$$Y_i \sim \text{Be}\left(\theta_i(\beta_0, \boldsymbol{\beta})\right)$$

where

$$\theta_i(\beta_0, \boldsymbol{\beta}) = \frac{1}{1 + \exp\left(-\beta_0 - \sum_{j=1}^{p} \beta_j x_{i,j}\right)} \tag{2}$$

is the probability of success for individual i, given the predictors $x_{i,1}, \ldots, x_{i,p}$ and the parameters $\beta_0, \beta_1, \ldots, \beta_p$ as determined by the logistic transformation of the linear predictor. This is analogous to the case of normal linear regression in which we model each observation as coming from a different normal distribution with a mean determined by the linear predictor. Though each observation is no longer identically distributed, we still assume they are independent, and so the joint-probability (i.e., the likelihood) of the n observations is just

$$\begin{aligned} p(\mathbf{y} \,|\, \beta_0, \boldsymbol{\beta}) &= \prod_{i=1}^{n} p(y_i | \beta_0, \boldsymbol{\beta}) \\ &= \prod_{i=1}^{n} \theta_i(\beta_0, \boldsymbol{\beta})^{y_i} \left(1 - \theta_i(\beta_0, \boldsymbol{\beta})\right)^{1-y_i} \end{aligned}$$

which follows from the definition of the Bernoulli distribution. Taking the negative logarithm of the above expression yields the negative log-likelihood, which is

$$\begin{aligned} L(\mathbf{y} \,|\, \beta_0, \boldsymbol{\beta}) &= -\sum_{i=1}^{n} \left[y_i \log \theta_i(\beta_0, \boldsymbol{\beta}) + (1 - y_i) \log\left(1 - \theta_i(\beta_0, \boldsymbol{\beta})\right)\right] \\ &= \sum_{i=1}^{n} \left[-y_i \eta_i + \log\left(1 + e^{\eta_i}\right)\right] \end{aligned}$$

where the second step comes from substituting (**??**) into $\theta_i(\cdot)$ and simplifying. In the case of logistic regression, the usual approach of differentiating the negative log-likelihood with respect to the parameters $\boldsymbol{\beta}$ and β_0 and solving for the values that set the gradient to zero does not yield simple, closed form solutions; i.e., the equations can not be solved directly. Instead, most packages use a numerical search to find the values that minimise the negative log-likelihood. Luckily, a special property of the logistic transformation is that the resulting logistic regression is guaranteed to have at most only a single, unique minimising value of $\boldsymbol{\beta}$ and β_0. The time complexity of finding such estimates for most basic algorithms is $O(p^3)$, i.e., scales cubically in the number of dimensions.

Assessing Fit and Selecting Models. For logistic regressions there is no direct equivalent of the R^2 measure we used to quantify how well a linear model fits a set of data. However, we can use the minimised negative log-likelihood $L(\mathbf{y} \,|\, \hat{\beta}_0, \hat{\beta})$ as a measure of goodness-of-fit, and use the decrease in negative log-likelihood over the intercept only model, i.e.,

$$L(\mathbf{y} \,|\, \hat{\beta}_0) - L(\mathbf{y} \,|\, \hat{\beta}_0, \hat{\beta})$$

as a measure of how much improvement including the predictors has made to our fit. The model using only the predictor is equivalent to assuming that $\mathbb{P}(Y = 1)$ is the same for all individuals, and so this tells us how much including individual-specific predictor information will improve our fit. The likelihood can also be used to help select which predictors should be included in a similar fashion to linear models. For a specific set of predictors we can form an information criterion score of the form

$$L(\mathbf{y} \,|\, \hat{\beta}_0, \hat{\beta}) + k\alpha_n$$

where k is the number of predictors in the model and α_n is a penalty term. Standard choices include $\alpha_n = 1$ for the AIC, $\alpha_n = 3/2$ for the KIC and $\alpha_n = (1/2)\log n$ for the BIC. We then search for the set of predictors that has the minimum information score, i.e., the best trade off between the complexity of the model and its ability to fit data. Forward and backward stepwise selection procedures, similar to those used in linear regression (Lecture 6), are often employed.

An alternative approach to selecting whether to include a predictor is through hypothesis testing; here we exploit the fact that $\beta_j = 0$ implies that predictor j is not associated with the target $\mathbf{y}$. We can then test the hypothesis

$$H_0 : \beta_j = 0 \text{ vs } H_A : \beta_j \neq 0$$

with smaller p-values being stronger evidence against the null hypothesis that $\beta_j = 0$, i.e., that there is no association. If the p-value is sufficiently small we may reject the null and conclude that there is an association between predictor j and the target.

Predicting with a Logistic Regression. Once we have fitted a logistic regression to a sample of data, we can use the resulting model to make predictions about new data arising from the same population. This is very straightforward given the simple structure of the logistic regression model; in particular, if we have estimates $\hat{\beta}_0$ and $\hat{\boldsymbol{\beta}}$, and we obtain some new measurements of our p features for a new individual, say $x'_1, \ldots, x'_p$, we first calculate the linear predictor for this individual

$$\hat{\eta} = \hat{\beta}_0 + \sum_{j=1}^{p} \hat{\beta}_j x'_j$$

We can then use this to estimate that odds in favour of $Y' = 1$ for this individual using $\hat{O} = e^{\hat{\eta}}$, or to estimate the probability that $Y' = 1$ for these features using:

$$\mathbb{P}(Y' = 1 \,|\, x'_1, \ldots, x'_p) = \frac{1}{1 + \exp(-\hat{\eta})}.$$

Furthermore, if we need to make a guess at the most likely value of Y', i.e., what class they are most likely to be in, we can choose the value of $Y' \in \{0, 1\}$ that has the highest probability; i.e., if $\hat{\eta} > 0$ we would estimate $\hat{Y}' = 1$, and if $\hat{\eta} < 0$ we would estimate $\hat{Y}' = 0$.

Extensions. A particular strength of the logistic regression approach to building classifiers is that it builds directly on regular linear regresson. As such, the standard tools for extending linear models are available to logistic regressions. Categorical variables can be handled by creating indicator variables, and potential non-linearities can be corrected for by transforming predictors appropriately, e.g., logarithmic transformations or polynomial transformations.

	$y_i = 0$	$y_i = 1$
$\hat{y}_i = 0$	True Negative (TN)	False Negative (FN)
$\hat{y}_i = 1$	False Positive (FP)	True Positive (TP)

Table 3: A confusion matrix for predicting binary targets.

3 Assessing Classifiers

Making predictions. This material is not specific to logistic regression models *per se*; rather it is applicable to any classification model (we shall see some alternatives later in the subject) that produces probabilities of an individual being in one of two different classes. Imagine we have trained such a model, and we obtain a body of new data from the same population. We would like to have some way of assessing how well our classifier works at predicting the classes of these new individuals. This is a well studied problem, and we will examine four metrics that we can use to assess the performance of classifier: (i) classification error; (ii) sensitivity/specificity; (iii) area-under-the-curve (AUC) and (iv) logarithmic loss.

To set the scene, let $\mathbf{y}' = (y'_1, \ldots, y'_{n'})$ be a vector of recorded classes of n' new individuals that we will use to test our model. Let $\mathbf{x}'_i = (x'_{i,1}, \ldots, x'_{i,p})$ be the feature vector for associated with individual i. Then, for each of the new individuals, we can calculate our best guess at which class it belongs to using:

$$\hat{y}'_i = \underset{y \in \{0,1\}}{\arg\max} \left\{ \mathbb{P}(Y'_i = y \,|\, x'_{i,1}, \ldots, x'_{i,p}) \right\}$$

where the probabilities are estimated using the model we have learned from our training data. That is, we choose to predict an individual to be in the class that our model says they are most likely to be in; if $\mathbb{P}(Y'_i = 1) > \mathbb{P}(Y'_i = 0)$ we choose $\hat{y}'_i = 1$, else we use $\hat{y}'_i = 0$.

Classification Accuracy, Sensitivity, Specificity. Using these quantities, the simplest measure of performance is classification accuracy. This is defined formally as

$$\text{CA} = \frac{1}{n'} \sum_{i=1}^{n'} I\left(y'_i = \hat{y}'_i\right)$$

where $I(\cdot)$ is the indicator function that returns a one if the condition inside the parenthesis is met, and a zero otherwise. The classification accuracy is simply the proportion of times that our classifier has correctly guessed the class of a new individual. A classification accuracy of 1 means our classifier has correctly predicted the class for every individual in our test set, and a classification accuracy of 0 means they have incorrectly classified every individual. In practice, a classification accuracy of 1/2 is realistically the worst accuracy a classifier can achieve for binary targets, as if it achieves a score lower than 1/2 we can simply swap the output of our classifier and achieve a better score. More generally, once we have predictions of classes from our classifier for each of the individuals we can form what we call a confusion matrix. This is a 2×2 table that tabulates the number of times the predicted class agrees with the actual class of an individual, as well as the number of times they disagreed, and how they disagreed. An example is shown in Table **??**.

A true negative (TN) occurs when our classifier predicts $\hat{Y}' = 0$ and $y' = 0$, i.e., we correctly identify an individual as a 0. A true positive (TP) occurs when our classifier predicts $\hat{Y}' = 1$ and $y' = 1$, i.e., we correctly identify an individual as a 1. A false negative (FN) occurs when our prediction is $\hat{Y}' = 0$ but the true value is $y' = 1$, and a false positive (FP) occurs when our prediction is $\hat{Y}' = 1$ but the true value is $y' = 0$. This is a more nuanced breakdown of the behaviour of our classifier than simply reporting the overall accuracy. In fact, the classification accuracy can be derived from this information using:

$$\mathrm{CA} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

where TP + TN is the total number of correct classifications (true positives plus true negatives). However, the confusion matrix can be used to form other useful statistics that let us characterise the behaviour of the classifier; for example, two important statistics are the sensitivity, or true positive rate:

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

and the specificity, or true negative rate:

$$\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}$$

The sensitivity is the proportion of times we correctly classified individuals for which $y'_i = 1$. The specificity is the proportion of times we correctly classified individuals for which $y'_i = 0$. In general a high sensitivity can always be achieved at the expense of specificity, and vice versa. For example, if we have a classifier that predicts everyone to be in class 1 we will have a sensitivity of 1, but a very poor specificity. Conversely, a classifier that predicts everyone to be in class 0 will have a sensitivity of 1 but a very poor sensitivity. In general it is important to try and trade these off, or assess a classifier based on how the decisions it makes are going to be used. In some cases, making a false positive might have critical effects – for example, if we misidentify a person as having a serious disease and erroneously administer drugs with serious side effects or authorise unnecessary surgery. However, in other cases it might have little effect if we make a false positive; for example, if we erroneously conclude an individual prefers Android phones to Apple phones it might make little real difference beyond a small loss in sales.

Area-under-the-curve. So far we have decided to predict an individual to be in class 1 if

$$\mathbb{P}(Y'_i \,|\, x'_{i,1}, \ldots, x'_{i,p}) > 0.5$$

and predict them to be in class 0 otherwise. More generally, we could set a "detection threshold" $T \in (0, 1)$ for classifier, and make the decision to predict an individual to be in class 1 if

$$\mathbb{P}(Y'_i = 1 \,|\, x'_{i,1}, \ldots, x'_{i,p}) \geq T,$$

and predict them to be in class 0 otherwise. Classifying an individual into the most likely class is equivalent to using a threshold of $T = 1/2$. It is clear that by changing the value of T, we can change the behaviour of the predictions made by our classifier. If $T = 0$ then our classifier will predict everyone to be in class 1 (and thus have maximal sensitivity), and if $T = 1$ then our classifier will (essentially) predict everyone to be in class 0 (and thus have maximal specificity). Let $\mathrm{TPR}(T)$ and $\mathrm{TNR}(T)$ be the true positive rate and true negative rate of our classifier with decision threshold T. By varying T from 0 through to 1 we produce a whole series of different predictions, and we can plot the resulting true positive rates and true negative rates for these different values of T on a graph which we call the "receiver operating curve", or ROC. The area under this curve is called the "area-under-the-curve", or AUC, and is a single measure of the performance of a classifier. The AUC varies from 0 through to 1, and the larger this area the better the performance of the classifier at discriminating between individuals that are in class 0 and class 1.

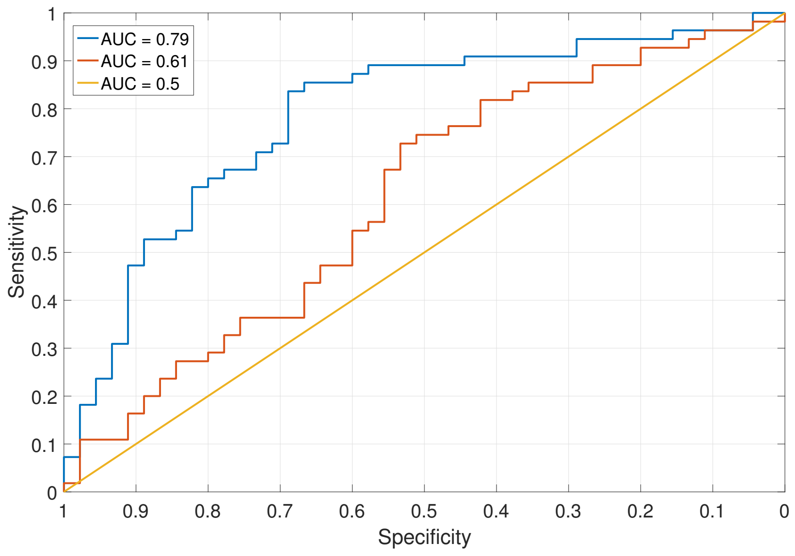

Figure 2: Receiver operating curves for two classifiers (orange and blue) as well as for random guessing (yellow)

The AUC lies strictly between 0 and 1, with an AUC of 1 meaning our classifier is perfect, an AUC of 0 meaning our classifier is perfectly incorrect, and an AUC of 0.5 meaning our classifier performs no better than a random coin toss. An example of an AUC for two different classifiers is shown in Figure **??**. The horizontal accuracy shows the specifity, and the vertical axis the sensitivity of our classifier as T is varied. The yellow diagonal line is the AUC of a random coin toss and is our base-line we compare all other models against. The orange line shows a classifier that is better than a random guess, as the sensitivity and specificity are almost always higher and the AUC is 0.61. The blue line shows another classifier that has a much higher AUC of 0.79; it is clear this classifier is superior to the orange classifier as its sensitivity and specificity are always either better, or at least no worse, for all values of T.

How do we interpret an AUC? It turns out the AUC does have a very specific probability interpretation that can initially be somewhat confusing. An AUC of p means that if we randomly sampled an individual i from our test group who had a $y' = 1$ (i.e., we randomly sampled a person from class 1), and randomly sampled an individual j who had $y' = 0$ (i.e., we randomly sampled a person from class 0), then

$$\mathbb{P}\left[\mathbb{P}(Y_i' = 1 \,|\, x_{i,1}', \ldots, x_{i,p}') > \mathbb{P}(Y_j' = 1 \,|\, x_{j,1}', \ldots, x_{j,p}')\right] = p$$

That is, the AUC can be interpreted as the probability that a random individual i sampled from class 1 will be rated more likely by our classifier to be in class 1 than a random individual j sampled from class 0. So an AUC of 0.6 means our classifier rates 60% of the individuals who are in class 1 as more likely to be in class 1 than individuals in class 0. Therefore an AUC of 1 means that our classifier rates all individuals who are in class 1 as being more likely to be in class 1 than all individuals who are in class 0. Don't stress if this is initially a little confusing – even many people who routinely use AUC often misunderstand the precise definition!

Logarithmic Loss. The final performance measure we consider is logarithmic loss. To compute the logarithmic loss we use the following procedure: for each sample i in our test group, we score

$$L(y_i') = \begin{cases} -\log \mathbb{P}(Y_i' = 1 \,|\, x_{i,1}', \ldots, x_{i,p}') & \text{for } y_i' = 1 \\ -\log \mathbb{P}(Y_i' = 0 \,|\, x_{i,1}', \ldots, x_{i,p}') & \text{for } y_i' = 0 \end{cases} .$$

This is the essentially the negative-log-probability of the test data point y_i' under our classification model. The total logarithmic loss is then given by adding these scores together, i.e.,

$$L(\mathbf{y}') = \sum_{i=1}^{n} L(y_i')$$

which is the complete negative-log-likelihood of this new, future data under the model we fitted to our *training data*. The smaller the score, the better our classifier predicts this data. More importantly, the log-loss measures how well the model predicts the probabilities of an individual being in a class (calibration). The logarithmic loss is intuitively a good measure because we have already stated that the smaller the negative log-likelihood, the more compatible the data is with our model. We use this principle to fit our model, so intuitively, if a model also has a small negative log-likelihood on future data, it also fits the future data well.

In contrast to classification accuracy, which measures how good our guesses at the most likely class are, the log-loss measures how well the model predicts the probabilities of an individual being in a class. This is a crucial difference as the probabilities tells you how confident you should be in your predicted class. For example, both $P(Y = 1 \,|\, x_1, \ldots, x_p) = 0.501$ and $P(Y = 1 \,|\, x_1, \ldots, x_p) = 0.99$ would predict the most likely class for Y to be $Y = 1$. However, we are much more confident about latter prediction than the former. Therefore if we have a model that does well at estimating the conditional probabilities of an individual being in a class then we can be more comfortable with using these probabilities to determine how confident we are in our predictions. A model with smaller logarithmic loss does a better job of estimating the probabilities of an individual being in one of the two target classes than a model with a larger logarithmic loss.