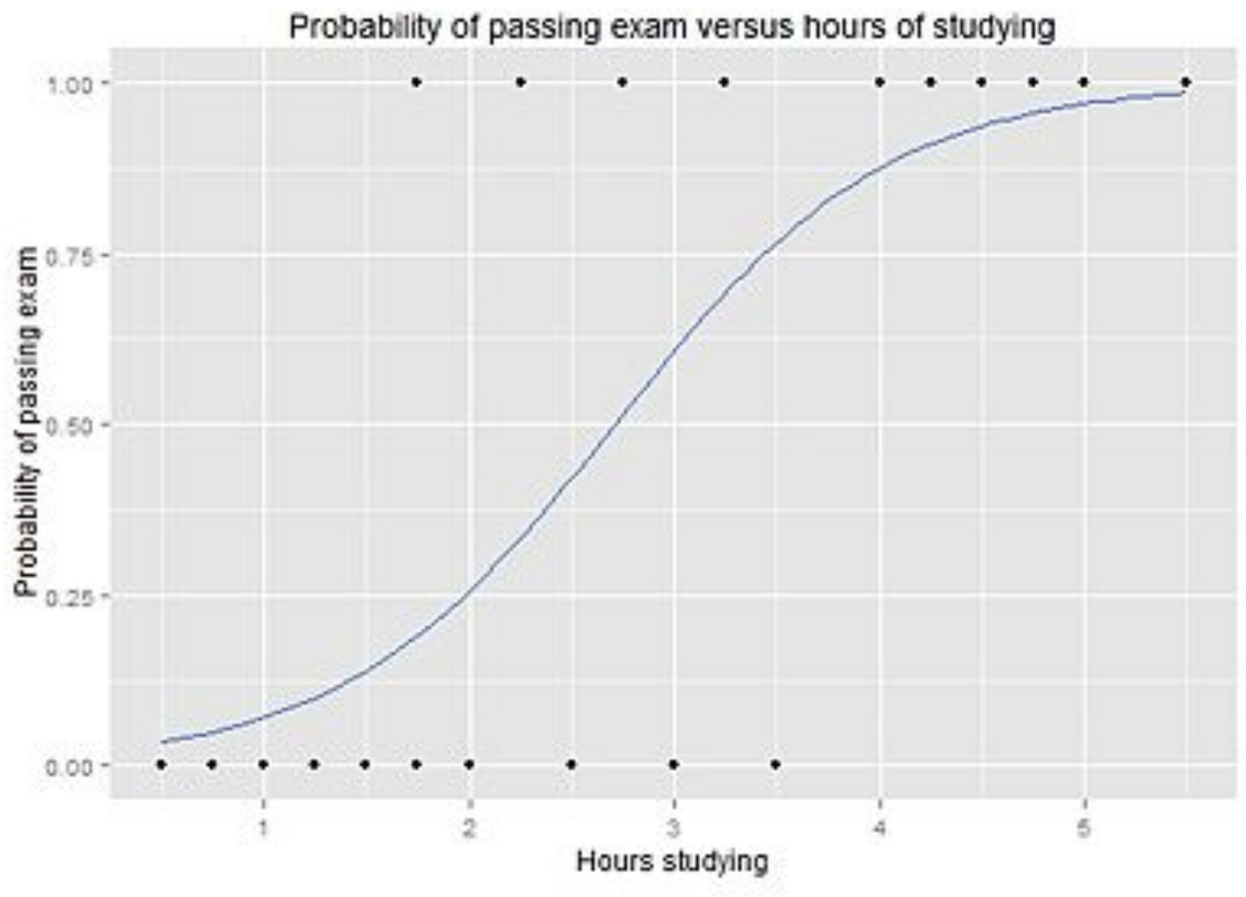
Logistic regression

What is Logistic Regression?

- supervised machine learning method that uncovers a linear relationship between one independent continuous variables and one dependent binary variable
  - y, is exponential family of probability distributions

sigmoid: s-shaped curve: that can take any real-valued number and map it into a value between 0 and 1



1 / (1 + e^-value)

Logistic function

- P: probability of success
- success: labeled ‘1’, failure:labeled ‘0’
- Odd of success

$$Odds = \frac{p}{1-p}$$

- Logistic function

$$\ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p = X\beta$$

$$p = \frac{e^{X\beta}}{1+e^{X\beta}} = \frac{1}{1+e^{-X\beta}}$$

- When $e^{X\beta}$ tends towards 0, p tends towards 0.

- When $e^{X\beta}$ tends towards ∞, p tends towards 1.

Estimating the coefficients

The best coefficients would result in a model that would predict a value very close to 1 for the default class and a value very close to 0 for the other class. The intuition for maximum-likelihood for logistic regression is that a search procedure seeks values for the coefficients that minimize the error in the probabilities predicted by the model to those in the data (e.g. probability of 1 if the data is the primary class).

Maximum likelihood estimation:

$$p = \frac{e^{X\beta}}{1+e^{X\beta}} = \frac{1}{1+e^{-X\beta}}$$

$$l(\beta_0, \beta_1, ..., \beta_p) = \prod_{i:y_i=1} p(x_i) \prod_{j:y_j=0} (1-p(x_j))$$

- joint probabilities of success and failure within the data set
- include the coefficients parameter
- maximize the function to select the combination of coefficients that produces the highest likelihood for the data set

Estimating the coefficients

For example:

$$\begin{aligned}
\ell(\beta_0, \beta) &= \sum_{i=1}^{n} y_i \log p(x_i) + (1 - y_i) \log 1 - p(x_i) \\
&= \sum_{i=1}^{n} \log 1 - p(x_i) + \sum_{i=1}^{n} y_i \log \frac{p(x_i)}{1 - p(x_i)} \\
&= \sum_{i=1}^{n} \log 1 - p(x_i) + \sum_{i=1}^{n} y_i (\beta_0 + x_i \cdot \beta) \\
&= \sum_{i=1}^{n} - \log 1 + e^{\beta_0 + x_i \cdot \beta} + \sum_{i=1}^{n} y_i (\beta_0 + x_i \cdot \beta)
\end{aligned}$$

Typically, to find the maximum likelihood estimates we'd differentiate the log likelihood with respect to the parameters, set the derivatives equal to zero, and solve.

$$\begin{aligned}
\frac{\partial \ell}{\partial \beta_j} &= - \sum_{i=1}^{n} \frac{1}{1 + e^{\beta_0 + x_i \cdot \beta}} e^{\beta_0 + x_i \cdot \beta} x_{ij} + \sum_{i=1}^{n} y_i x_{ij} \\
&= \sum_{i=1}^{n} (y_i - p(x_i; \beta_0, \beta)) \, x_{ij}
\end{aligned}$$

Prediction

Predict the probability of success

$$p(X) = \frac{e^{X\beta}}{1+e^{X\beta}} = \frac{1}{1+e^{-X\beta}}$$

$$\hat{y}_i = \begin{cases} Success\ (1) & \hat{p}_i \geq c \\ Failure\ (0) & \hat{p}_i < c \end{cases}$$

usually c=0.5

Prepare Data

The assumptions made by logistic regression about the distribution and relationships in the data are much the same as the assumptions made in linear regression.
Ultimately in predictive modeling machine learning projects you are laser focused on making accurate predictions rather than interpreting the results. As such, you can break some assumptions as long as the model is robust and performs well.

- Binary Output Variable: This might be obvious as we have already mentioned it, but logistic regression is intended for binary (two-class) classification problems. It will predict the probability of an instance belonging to the default class, which can be snapped into a 0 or 1 classification.
- Remove Noise: Logistic regression assumes no error in the output variable (y), consider removing outliers and possibly misclassified instances from your training data.
- Gaussian Distribution: Logistic regression is a linear algorithm (with a non-linear transform on output). It does assume a linear relationship between the input variables with the output. Data transforms of your input variables that better expose this linear relationship can result in a more accurate model. For example, you can use log, root, Box-Cox and other univariate transforms to better expose this relationship.
- Remove Correlated Inputs: Like linear regression, the model can overfit if you have multiple highly-correlated inputs. Consider calculating the pairwise correlations between all inputs and removing highly correlated inputs.
- Fail to Converge: It is possible for the expected likelihood estimation process that learns the coefficients to fail to converge. This can happen if there are many highly correlated inputs in your data or the data is very sparse (e.g. lots of zeros in your input data).

Evaluating goodness of fit

Deviance: similar to RSS

$$G^2 = 2[\ln(l_S) - \ln(l_M)]$$

- ls:likelihood of the fitted model
- lm: likelihood of saturated model
- saturated model: has a parameter fit every observations in the dataset, leading to overfitting
- smaller deviance: a better fit

McFadden's Pseudo R^2_L : similar to R^2

$$D_{\text{null}} = -2\ln\frac{\text{likelihood of null model}}{\text{likelihood of the saturated model}}$$

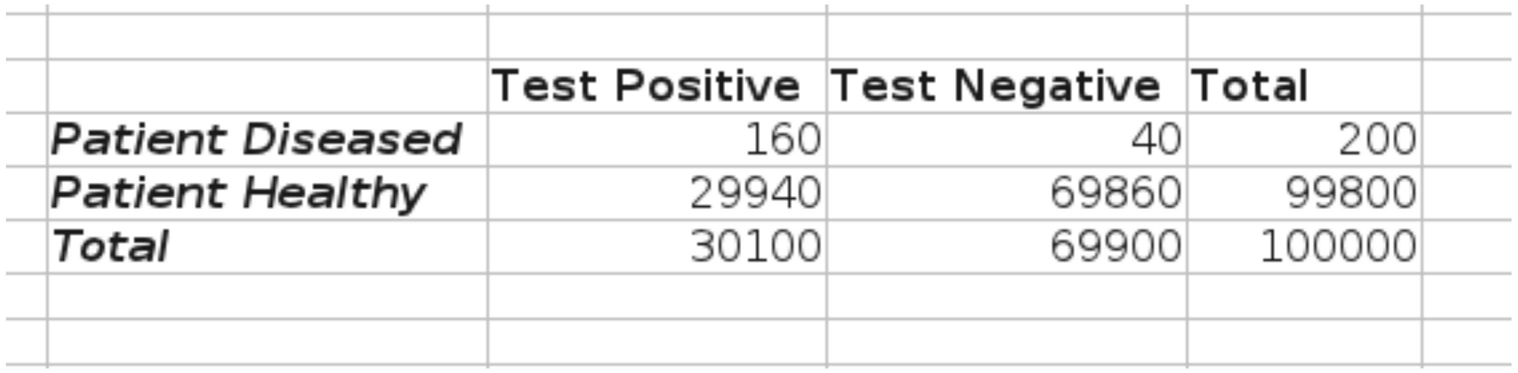
$$D_{\text{fitted}} = -2\ln\frac{\text{likelihood of fitted model}}{\text{likelihood of the saturated model}}.$$

$$R^2_{\text{L}} = \frac{D_{\text{null}} - D_{\text{fitted}}}{D_{\text{null}}}$$

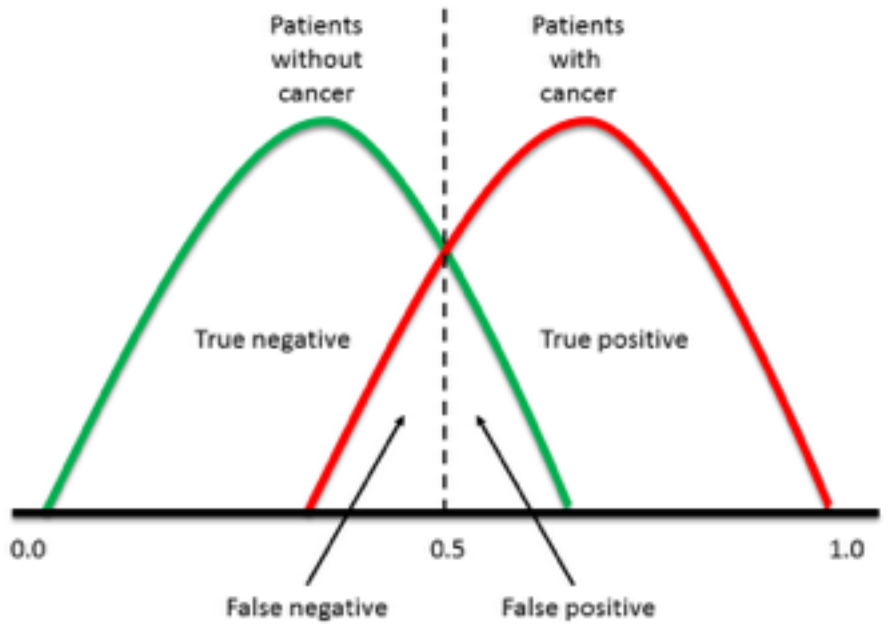
- null model: only includes intercept term (with no predictors)
- higher R^2_L: a better fit
- Null Deviance is calculated based on comparing the saturated model against the model that only includes intercept ; it is a way of assessing the overall maximum deviance because it compares the most complicated model to the most simplistic model.
- Fitted/Residual Deviance is calculated based on comparing the saturated model against the model at hand, it is a measure of how well fits the data in general.

Classification model assessment

- Use thresholded to decide the classification. If a classifier produces a score between 0.0 (definitely negative) and 1.0 (definitely positive), it is common to consider anything over 0.5 as positive.
- Any threshold applied to a dataset (in which PP is the positive population and NP is the negative population) is going to produce true positives (TP), false positives (FP), true negatives (TN) and false negatives (FN)
- Accuracy = (1 – Error) = (TP + TN)/(PP + NP) = Pr(C), the probability of a correct classification.
- Sensitivity = TP/(TP + FN) = TP/PP = the ability of the test to detect disease in a population of diseased individuals.
- Specificity = TN/(TN + FP) = TN / NP = the ability of the test to correctly rule out the disease in a disease-free population.



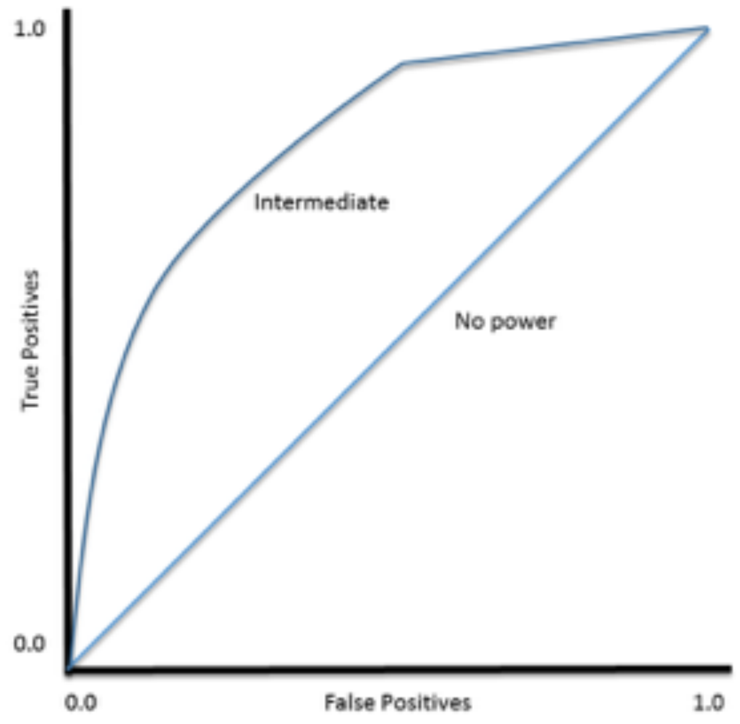
	Test Positive	Test Negative	Total
Patient Diseased	160	40	200
Patient Healthy	29940	69860	99800
Total	30100	69900	100000



- Sensitivity = TP/(TP + FN) = 160 / (160 + 40) = 80.0%
- Specificity = TN/(TN + FP) = 69,860 / (69,860 + 29,940) = 70.0%
- The test will correctly identify 80% of people with the disease, but 30% of healthy people will incorrectly test positive.

ROC (receiver operating characteristic) curve

- One way to overcome the problem of having to choose a cutoff is to start with a threshold of 0.0, so that every case is considered as positive. We correctly classify all of the positive cases, and incorrectly classify all of the negative cases. We then move the threshold over every value between 0.0 and 1.0, progressively decreasing the number of false positives and increasing the number of true positives.
- ROC curve: plot sensitivity (true positive rate) vs specificity (false positive rate)
- ROC curve: select a threshold for a classifier which maximizes the true positives and minimizes the false positives
- AUC (the area under the curve): assess the performance of the classifier
- AUC of less than 0.5 : indicate that something interesting is happening


True Positives
Intermediate
No power
False Positives