Summary

May 17, 2018

1 LINEAR REGRESSION

A regression model that describes the relation between one target variable(continuous) and one or more **independent** variables.

1.1 R^2 VALUE

R^2 is the square of the correlation coefficient and represents the estimated percentage of the variance in our target variable Y that can be explained by our regression model.

1.2 ADJUSTED R^2 VALUE

Adjusted R^2 penalizes for things such as large coefficients and extra variables to try and limit overfitting so it is often a better measure of model efficacy.

1.3 T-SCORES FOR COEFFICIENTS

The t-scores are values that the coefficients score in the Student's T Distribution

1.4 P-VALUE FOR COEFFICIENTS

The p-value represents the probability of finding such a t-score if the actual value of the coefficient were 0. It measures our degree of belief that the coefficient for each variable(feature) should be zero. Thus, the lowest P-values represent the most likely predictors to be impacting the response.

1.5 ASSUMPTIONS FOR LINEAR REGRESSION

1. Regression is linear in parameters and correctly specified
2. The error terms are normally distributed and zero population mean
3. The error term has constant variance for every data point i.
4. Errors are uncorrelated across observations, which means covariance $COV(\epsilon_i, \epsilon_j)$ errors is 0.
5. All variables(features) are independent.

1.6 REGULARIZATION

Regularization is a technique used in an attempt to solve the overfitting problem in statistical models. Regularization penalizes large weights in the regression model because large weights result in complex model and might cause overfitting.

Cost Function of Linear Regression (MSE):

$$J(\epsilon_0, \epsilon_1) = \frac{1}{2m} \sum_{i}^{m} ((\beta_0 + \beta_1 x^{(i)}) - y^{(i)})^2$$

L2 Regularization (Ridge Regression): $J(\epsilon_0, \epsilon_1) + \lambda \sum {\beta_j}^2$

L1 Regularization (Lasso Regression): $J(\epsilon_0, \epsilon_1) + \lambda \sum |\beta_j|$

Elastic Net (L1+L2): $J(\epsilon_0, \epsilon_1) + \lambda_1 \sum {\beta_j}^2 + \lambda_2 \sum |\beta_j|$

L1 norm regularization and sparsity
Assume

$$\left\|\vec{x}\right\|_1 = 1 + \epsilon$$

and

$$\left\|\vec{x}\right\|_2{}^2 = 1 + \epsilon^2$$

For an $l2$ penalty, regularizing the larger term x_1(x_1 is a feature) results in a much greater reduction in norm than doing so to the smaller term $x2 \approx 0$. For the $l1$ penalty, however, the reduction is the same. Thus, when penalizing a model using the $l2$ norm, it is highly unlikely that anything will ever be set to zero, since the reduction in $l2$ norm going from ϵ to 0 is almost nonexistent when ϵ is small. On the other hand, the reduction in $l1$ norm is always equal to δ, regardless of the quantity being penalized.
Intuitively, in order to decrease the same amount of loss, $l1$ will have to decrease the coefficients by a larger magnitude comparing to $l2$.

Diagnostics to Detect Under/Overfitting:

1. Underfitting: High + Low
2. Just Right: Low + Low
3. Overfitting: Low + High

1.7 CROSS-VALIDATION

Cross validation is a model evaluation method that is better than residuals. The problem with residual evaluations is that they do not give an indication of how well the learner will do when it is asked to make new predictions for data it has not already seen. One way to overcome this problem is to not use the entire data set when training a learner. Some of the data is removed before training begins. Then when training is done, the data that was removed can be used to test the performance of the learned model on 'new' data. This is the basic idea for a whole class of model evaluation methods called cross validation.

Holdout Method
The data set is separated into two sets, called the training set and the testing set.

K-fold cross validation
The data set is divided into k subsets, and the holdout method is repeated k times. Each time, one of the k subsets is used as the test set and the other k-1 subsets are put together to form a training set. Then the average error across all k trials is computed. The advantage of this method is that it matters less how the data gets divided.

2 LOGISTIC REGRESSION

Logistic regression is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome(categorical variable). Logistic regression models the probability that a target variable Y belongs in particular category.

2.1 SIGMOID(LOGISTIC) FUNCTION

We must model $p(\mathbf{X})$ using a function that gives outputs between 0 and 1 for all values of $\mathbf{X}$. Many functions meet this description. In logistic regression, we use the sigmoid function,

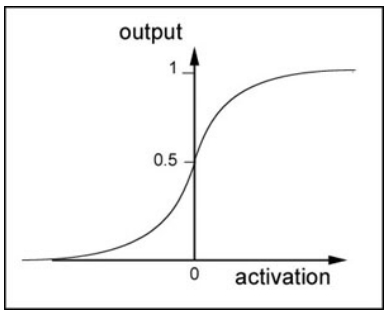
$$p(\mathbf{X}) = \frac{\exp(\beta_0 + \beta_1 \mathbf{X})}{1 + \exp(\beta_0 + \beta_1 \mathbf{X})}$$

, which can also be written as:

$$p(\mathbf{X}) = \frac{1}{1 + \exp(-(\beta_0 + \beta_1 \mathbf{X}))}$$

Ff β_1 is positive then increasing $\mathbf{X}$ will be associated with increasing $p(\mathbf{X})$, and if β_1 is negative then increasing $\mathbf{X}$ will be associated with decreasing $p(\mathbf{X})$.

Figure 2.1: Sigmoid Function



2.2 ODDS

The quantity of $\frac{p(\mathbf{X})}{1-p(\mathbf{X})}$ is called the odds. Values of the odds close to 0 and ∞ indicate very low and very high probabilities of default, respectively.

$$\frac{p(\mathbf{X})}{1-p(\mathbf{X})} = \exp(\beta_0 + \beta_1 \mathbf{X})$$

2.3 LOG-ODDS(LOGIT)

Logit is defined as:

$$\log(\frac{p(\mathbf{X})}{1-p(\mathbf{X})}) = \beta_0 + \beta_1 \mathbf{X}$$

We see that the logistic regression model has a logit that is linear in X.

3 SUPPORT VECTOR MACHINE (SVM)

A Support Vector Machine (SVM) is a discriminative classifier formally defined by one or more separating hyperplanes. In other words, given labeled training data (supervised learning), the algorithm outputs one or more optimal hyperplanes which categorize test examples.

3.1 HYPERPLANE

In a n-dimensional space, a hyperplane is a flat affine subspace of dimension $n-1$. Mathematically, a hyperplane in 2 dimension is defined by the

equation

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$$

for parameters β_0, β_1 and β_2.
If **X** satisfies that

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 > 0$$

, **X** lies on one side of the hyperplane.
If **X** satisfies that

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 < 0$$

, **X** lies on the other side of the hyperplane.

3.2 THE MAXIMAL MARGIN CLASSIFIER

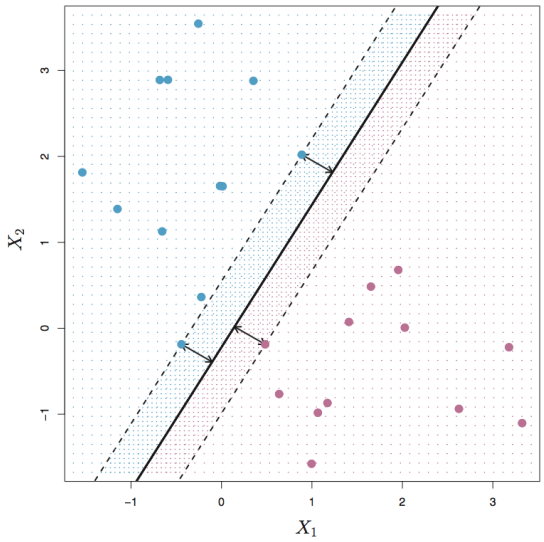
In general, if our data can be perfectly separated using a hyperplane, then there will in fact exist an infinite number of such hyperplanes. This is because a given separating hyperplane can usually be shifted a tiny bit up or down, or rotated. But which hyperplane is the best?

A natural choice is the **maximal margin hyperplane** (also known as the optimal separating hyperplane), which is the separating hyperplane that is farthest from the training observations. That is, we can compute the (perpendicular) distance from each training observation to a given separating hyperplane; the smallest such distance is the minimal distance from the observations to the hyperplane, and is known as the **margin**. The maximal margin hyperplane is the separating hyperplane for which the margin is largest-that is, it is the hyperplane that has the farthest minimum distance to the training observations.

3.3 SUPPORT VECTOR

In figure 3.1, we see that three training observations on the dashed lines are equidistant from the maximal margin hyperplane. The margin is the distance from the solid line to either of the dashed lines. These three observations are known as **Support Vector** since they "support" the maximal margin hyperplane in the sense that if these points were moved slightly then the maximal margin hyperplane would move as well.

Figure 3.1: Maximal Margin Hyperplane



3.4 HOW TO MAXIMIZE THE MARGIN?

The construction of the maximal margin classifier is a little complicated. Further details can be found here. But in general, the problem of maximizing the margin is just a convex optimization problem.

Given a training example $(\mathbf{x}^{(i)}, y^{(i)})$ and a hyperplane $w^T\mathbf{x}+b$, we define the distance from i^{th}training point to the hyperplane as:

$$d^{(i)} = y^{(i)}(w^T\mathbf{x}+b)$$

Note that if $y^{(i)} = 1$, then for the functional margin to be large, we need $w^T\mathbf{x}+b$ to be a large positive number. Conversely, if $y^{(i)} = -1$ then for the functional margin to be large, we need $w^T\mathbf{x}+b$ to be a large negative number.

Recall that we define the margin, denoted by γ, to be the minimal distance from the observations to the hyperplane.

$$\gamma = \min_{i=1,\ldots n} d^{(i)}$$

The goal here is to maximize the margin γ with a constraint that makes sure the margin is th minimal distance. This statement can be represented as the following optimization problem.

$$\begin{aligned} &\underset{\gamma,w,b}{\text{maximize}} && \gamma \\ &\text{subject to} && y^{(i)}(w^T\mathbf{x}+b) \geq \gamma,\ i=1,\ldots,m. \end{aligned}$$

The constraint guarantees that the distance from the training observation to the hyperplane must be greater than or equal to the margin.
Through derivation, the above optimization problem can be rewritten as:

$$\begin{aligned} &\underset{\gamma,w,b}{\text{minimize}} && \frac{1}{2}||w||^2 \\ &\text{subject to} && y^{(i)}(w^T\mathbf{x}+b) \geq 1,\ i=1,\ldots,m. \end{aligned}$$

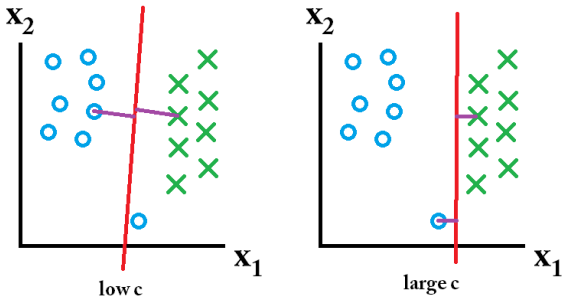
3.5 SOFT MARGIN SVM

If our data is not linearly separable, we use the soft margin SVM. In soft margin SVM, we reformulate our optimization by introducing a slack variable ζ.(L1 regularization)

$$\begin{aligned} &\underset{\gamma,w,b}{\text{minimize}} && \frac{1}{2}||w||^2 + C\sum_{i=1}^{m}\zeta_i \\ &\text{subject to} && y^{(i)}(w^T\mathbf{x}+b) \geq 1-\zeta_i,\quad \zeta_i \geq 0,\ i=1,\ldots,m. \end{aligned}$$

The C parameter tells the SVM optimization how much you want to avoid misclassifying each training example. For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Thus, you will not neglect the outliers and will end up with a much smaller margin. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points. However, this requires that we neglect the outliers that we have failed to classify correct.

Figure 3.2: The effect of C parameter



3.6 KERNEL

In linear regression, we sometimes consider to perform regression using features ${x_1}^2$, ${x_3}^3$, x_1x_2 and so on. The same analogy applies here in SVM. By using the new features, we are able to generate non-linear boundaries. Instead of calculating the mapping function $\phi(x)$, which costs $O(n^2)$ time, we can just calculate the inner product of the mapping function, such that $K(x,z) = \phi(x)^T\phi(z)$, which only takes $O(n)$ time.

Further details can be found here on page 13.

3.7 PROS AND CONS

Pros:
1. Guaranteed Optimality: Due to the nature of Convex Optimization, the solution is guaranteed to be the global minimum not a local minimum.
2. Can deal with high dimensional data.
3. It is useful for both Linearly Seperable(hard margin) and Non-linearly Seperable(soft margin) data. The only thing to do is to come up with the optimal penalty variable C (the one that multiplies slack variables)
4. Conformity with Semi-Supervised Learning: It may be used in a dataset where some of the data are labeled and some are not. You only add an additional condition to the minimization problem and it is called Transductive SVM.
5. Feature Mapping might have been a burden on the computational complexity of the overall training performance; however, thanks to the 'Kernel

Tric' the feature mapping is implicitly carried out via simple dot products.

Cons:
1. Require both positive and negative examples.
2. Need to select a good kernel function.
3. Require lots of memory and CPU time.
4. There are some numerical stability problems in solving the constrained QP.

4 DECISION TREE

To build a tree, we take atop-down,greedy approach that is known as recursive binary splitting. The approach is top-down because it begins at the top of the tree(at which point all observations belong to a single region) and then successively splits the predictor space; each split is indicated via two new branches further down on the tree. It is greedy because at each step of the tree-building process, the best split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step.

4.1 PROS AND CONS

Advantage:
The decisions are easy to understand and interpret.
Both numerical and categorical features can be used naturally.
Natural multiclass classifier.

Disadvantage:
Can overfit to training data with complex trees.
Small changes in input data can result in totally different trees.
Can make mistake with unbalanced classes.
No confidence intervals(regression).

4.2 REGRESSION TREE

Any combination of feature X_j and cutpoing s splits the data into 2 regions:

$$R_1(j,s) = [X|X_j < s] \;\; and \;\; R_2(j,s) = [X|X_j > s]$$

To build a regression tree, we consider all the remaining features $X_1, ..., X_p$ and all possible values of the cutpoint s for each of the features, and then choose the feature and cutpoint such that the resulting tree has the lowest RSS, which is defined as:

$$\sum_{i:x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2,$$

where $\hat{y}_{R_1}$ is the mean response for the training observations in $R_1(j,s)$ and $\hat{y}_{R_2}$ is the mean response for the training observations in $R_2(j,s)$.

4.3 CLASSIFICATION TREE

The same analogy from regression tree applies here as well. However, instead of minimizing RSS, we use cross-entropy to evaluate the quality of a particular split. You can think of cross-entropy as a measurement of the purity of a node. The cross-entropy will take on small value if the node is pure and is defined as:

$$D = -\sum_i p_i \log p_i$$

where p_i is the proportion of training observations that are from the ith class.

4.4 REGULARIZATION

1. Minimum number of points per cell:
require that each cell (i.e., each leaf node) covers a given minimum number of training points.
2. Maximum number of cells:
limit the maximum number of cells of the partition (i.e., leaf nodes).
3. Maximum depth:
limit the maximum depth of the tree.

4.5 TREE PRUNING

A smaller tree with fewer splits (that is, fewer regions R_1,...,R_J) might lead to lower variance and better interpretation at the cost of a little bias.
Therefore, a better strategy is to grow a very large tree T_0, and then prune it back in order to obtain a subtree.
For each value of α there corresponds a subtree $T \subset T_0$ such that

$$\sum_{m=1}^{|T|} \sum_{x_i \subset R_m} (y_i - \hat{y}_{R_M})^2 + \alpha |T|$$

is as small as possible. Here $|T|$ indicates the number of terminal nodes of the tree T, R_M is the rectangle (i.e. the subset of predictor space) corresponding to the m^{th} terminal node,and y^{R_m} is the predicted response associated with R_m - that is, the mean of the training observations in R_m.
As α increases, there is a price to pay for having a tree with many terminal nodes, and so the cost function above will tend to be minimized for a smaller subtree.
We can select a value of α using a validation set or using cross-validation. We then return to the full data set and obtain the subtree corresponding to α.

5 RANDOM FOREST

5.1 ENSEMBLE METHOD: BAGGING

Bagging, or bootstrap aggregation, is a general-purpose procedure for reducing the variance of a statistical learning method.
The general idea is that we can take repeated samples from the training data set(bootstrap), build a separate prediction model using each sample, and average(regression) or take a majority vote(classification) on the resulting predictions.
Overall, bagging improves prediction accuracy at the expense of interpretability.

5.2 RANDOM FOREST

Random forests provide an improvement over bagged trees by way of a small tweak that decorrelates the trees. As in bagging, we build a number of decision trees on bootstrapped training samples. But when building these decision trees, each time a split in a tree is considered, a random sample of m($m \approx \sqrt{p}$) features is chosen as split candidates from the full set of p predictors.
In other words, in building a random forest, at each split in the tree, the algorithm is not even allowed to consider a majority of the available features. This may sound crazy, but it has a clever rationale. Suppose that there is one very strong feature in the data set, along with a number of other moderately strong features. Then in the collection of bagged trees, most or all of the trees will use this strong feature in the top split. Consequently, all of the bagged trees will look quite similar to each other. Hence the predictions from the bagged trees will be highly correlated. Unfortunately, averaging many highly correlated quantities does not lead to as large of a reduction in variance as averaging many uncorrelated quantities. In particular, this means that bagging will not lead to a substantial reduction in variance over a single tree in this setting.
Random forests overcome this problem by forcing each split to consider only a subset of the predictors. We can think of this process as decorrelating the trees, thereby making the average of the resulting trees less variable and hence more reliable.

5.3 VARIABLE IMPORTANCE MEASURES

One can obtain an overall summary of the importance of each predictor using the RSS (for bagging regression trees) or the Gini index (for bagging classification trees). In the case of bagging regression trees, we can record the total amount that the RSS is decreased due to splits over a given predictor, averaged over all B trees.

6 GRADIENT TREE BOOSTING

6.1 BOOSTING

Unlike fitting a single large decision tree to the data, which amounts to fitting the data hard and potentially overfitting, the boosting approach instead learns slowly. Given the current model, we fit a decision tree to the residuals from the model. That is, we fit a tree using the current residuals, rather than the outcome Y , as the response. We then add this new decision tree into the fitted function in order to update the residuals. Each of these trees can be rather small, with just a few terminal nodes, determined by the parameter d in the algorithm. By fitting small trees to the residuals, we slowly improve $\hat{f}$ in areas where it does not perform well. The shrinkage parameter λ slows the process down even further, allowing more and different shaped trees to attack the residuals. In general, statistical learning approaches that learn slowly tend to perform well. Note that in boosting, unlike in bagging, the construction of each tree depends strongly on the trees that have already been grown.
In boosting, each tree is grown using information from previously grown trees. Boosting does not involve bootstrap sampling; instead each tree is fit on a modified version of the original data set.
Boosting has **three** tuning parameters:

1. The number of trees B. Unlike bagging and random forests, boosting can overfit if B is too large, although this overfitting tends to occur slowly if at all. We use cross-validation to select B.

2. The shrinkage parameter λ,a small positive number.This controls the rate at which boosting learns. Typical values are 0.01 or 0.001, and the right choice can depend on the problem. Very small λ can require using a very large value of B in order to achieve good performance.

3. The number d of splits in each tree, which controls the complexity of the boosted ensemble. Often d = 1 works well, in which case each tree is a stump, consisting of a single split. In this case, the boosted ensemble is fitting an additive model, since each term involves only a single variable. More generally d is the interaction depth, and controls the interaction or-

der of the boosted model, since d splits can involve at most d variables.

7 NAIVE BAYES

The Naive Bayesian classifier is based on Bayes theorem with independence assumptions between predictors. A Naive Bayesian model is easy to build, with no complicated iterative parameter estimation which makes it particularly useful for very large datasets.

7.1 BAYES RULE

$$P_{y|\mathbf{x}}(i|\mathbf{X}) = \frac{P_{\mathbf{x}|y}(\mathbf{X}|i)P_y(i)}{P_{\mathbf{X}}(\mathbf{X})}$$

where i stands for the i^{th} class.
The class with the highest posterior probability $P_{y|\mathbf{x}}(i|\mathbf{X})$ is the outcome of prediction.

7.2 WHAT IS THE NAIVE PART?

The naive part of this model comes from the assumption that all the features are independent. Given the likelihood function $P_{\mathbf{x}|y}(x_1, x_2...x_n|i)$, the assumption is saying:

$$P_{\mathbf{x}|y}(x_1, x_2...x_n|i) = P_{\mathbf{x}|y}(x_1|i)P_{\mathbf{x}|y}(x_2|i)...P_{\mathbf{x}|y}(x_n|i)$$

7.3 NUMERIC FEATURES

For numeric features, we usually assume the features have certain probabilistic distribution given a class. That is saying the likelihood function $P_{\mathbf{x}|y}(\mathbf{X}|i)$ has certain probabilistic distribution.

8 CLUSTERING

8.1 K-MEANS

k-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean.

Complexity: $O(n)$

8.2 HIERARCHICAL CLUSTERING

Complexity: $O(n^2)$

9 SVD

We can perform matrix decomposition on a $m \times n$ matrix X with S.V.D such that $X = USV^T$, where U is a $m \times m$ real or complex unitary matrix, S is a $m \times n$rectangular diagonal matrix with non-negative real numbers on the diagonal and V is a $n \times n$ real or complex unitary matrix. The diagonal entries s_i of S are known as the singular values of X.

10 PCA

What principle component analysis does is to find directions of maximum variance of the data and perform dimension reduction.
Assume we have a $n \times p$ matrix X. Its covariance matrix C is defined as

$$\frac{1}{n-1}X^T X$$

The covariance matrix is symmetric and so it can be diagonalized as $C = VLV^T$, where V is the matrix of eigenvectors(each column is an eigenvector) and L is a diagonal matrix with eigenvalue λ_i in decreasing order on the diagonal.
The eigenvectors are called principle axes or principle directions of the data.
Projections of the data (XV)on the principle axes are called principle components.
i.e The j^{th} principle component is the j^{th} column of XV.
From S.V.D, we can decompose $X = USV^T$. Therefore, the covariance matrix

$$C = \frac{1}{n-1}X^T X = \frac{1}{n-1}VSU^T USV^T = V\frac{S^2}{n-1}V^T = VLV^T$$

$$\lambda_i = \frac{{S_i}^2}{n-1}$$

Therefore, the principle components are

$$XV = USV^T V = US$$

To sum up:
1. $X = USV^T$ columns of V are principle directions.
2. columns of US are principle components.
3. singular values of X (S_i)are related to eigenvalues (λ_i) of the covariance matrix C as

$$\lambda_i = \frac{{S_i}^2}{n-1}$$

4. since the principle components are US, to reduce dimensionality of data from p to k, select the first k columns of U and $k \times k$ upper left part of S, such that

$$U_k S_k \rightarrow n \times k$$

11 LDA

Explanation can be found here

12 NMF

A blog post about NMF topic modeling can be found here

13 CONFUSION MATRIX

	P'(Predicted)	N'(Predicted)
P'(Actual)	True Postive	False Negative
N'(Actual)	False Postive	True Negative

13.1 RECALL AND PRECISION

Recall = TP/(TP + FN)
Recall: Out of all the(few) postive cases, how many did I find.
Precision = TP/(TP+FP)
Precision: Out of all the cases that I predicted as postive, how many times am I right?

13.2 F1 SCORE

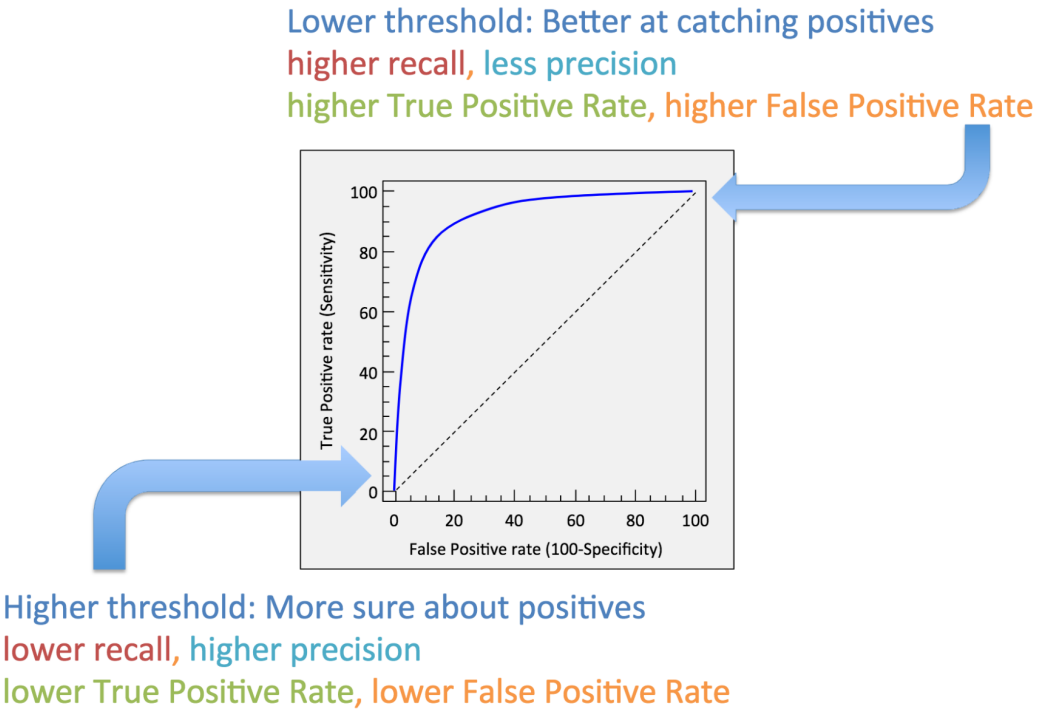
The marmonic mean of recall and precision.

$$F1 = 2 * \frac{precision * call}{precision + recall}$$

13.3 ROC CURVE

ROC curve represents a relation between sensitivity (RECALL, True Postive Rate) and specificity(NOT PRECISION, False Postive Rate) and is commonly used to measure the performance of binary classifiers.

Figure 13.1: ROC Curve



Intuition: In a ROC curve the true positive rate (Sensitivity) is plotted in function of the false positive rate (100-Specificity) for different cut-off points of a parameter. Each point on the ROC curve represents a sensitivity/specificity pair corresponding to a particular decision threshold. The area under the ROC curve (AUC) is a measure of how well a parameter can distinguish between two diagnostic groups (diseased/normal).
As we decrease the threshold, we are more likely to predict positive. And as a result, the recall will increase(more likely to captures all the positive cases in the dataset) but the precision will decrease.(less sure about the positive prediction)

13.4 IS IT BETTER TO HAVE TOO MANY FALSE POSITIVES, OR TOO MANY FALSE NEGATIVES?

It depends on the question as well as on the domain for which we are trying to solve the question.
In medical testing, false negatives may provide a falsely reassuring message to patients and physicians that disease is absent, when it is actually present. This sometimes leads to inappropriate or inadequate treatment of both the patient and their disease. So, it is desired to have too many false positive.
For spam filtering, a false positive occurs when spam filtering or spam blocking techniques wrongly classify a legitimate email message as spam and, as a result, interferes with its delivery. While most anti-spam tactics can block or filter a high percentage of unwanted emails, doing so without creating significant false-positive results is a much more demanding task. So, we prefer too many false negatives over many false positives.