# INTRODUCTION

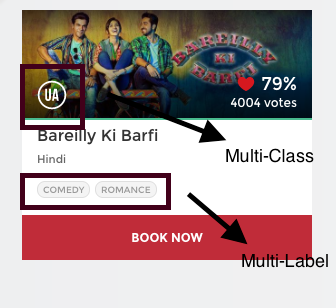
Classification can be performed on structured or unstructured data. Classification is a technique where we categorize data into a given number of classes. The main goal of a classification problem is to identify the category/class to which a new data will fall under.

Few of the terminologies encountered in machine learning – classification:

* **Classifier:** An algorithm that maps the input data to a specific category.
* **Classification model:** A classification model tries to draw some conclusion from the input values given for training. It will predict the class labels/categories for the new data.
* **Feature:** A feature is an individual measurable property of a phenomenon being observed.
* **Binary Classification:** Classification task with two possible outcomes. Example Gender classification (Male / Female)
* **Multi class classification:** Classification with more than two classes. In multi class classification each sample is assigned to one and only one target label. Example An animal can be cat or dog but not both at the same time. It is also known as multi-nominal classification.
* **Multi label classification:** Classification task where each sample is mapped to a set of target labels (more than one class). Example A news article can be about sports, a person, and location at the same time.

## MULTI-CLASS vs MULTI-LABEL

Consider an example to understand the difference between these two.



This movie has been rated as ‘U/A’ certificate. There are other types of certificates classes like ‘A’ or ‘U’ but it is sure that each movie can only be categorized with only one out of those three type of certificates. In short, there are multiple categories but each instance is assigned only one, therefore such problems are known as **multi-class classification problem**.

Again, if you look back at the image, this movie has been categorized into comedy and romance genre. But there is a difference that this time each movie could fall into one or more different sets of categories. Therefore, each instance can be assigned with multiple categories, so these types of problems are known as **multi-label classification problem**, where we have a set of target labels.

## CLASSIFICATION TYPES

Based on the number of classes (categories) to predict, classification task can be classified as:

1. **BINOMIAL**

Target variable can have only 2 possible types: “0” or “1” which may represent “win” vs “loss”, “pass” vs “fails”, “dead” vs “alive” etc.

1. **MULTINOMIAL**

Target variable can have 3 or more possible types which are not ordered (i.e. types have no quantitative significance) like “disease A” vs “disease B” vs “disease C”.

1. **ORDINAL**

It deals with target variables with ordered categories. For example, a test score can be categorized as: “very poor”, “poor”, “good”, “very good”. Here, each category can be given a score like 0, 1, 2, 3.

1. **POISSON**

Poisson regression is used when your response is a count of an incident.

Multinomial and Poisson are very different. Multinomial regression should be used when your response is categorical with more than 2 categories. Poisson regression is used when your response is a count of an incident.

Classic Poisson regression: response is # of accidents at an intersection in a day. This can be modeled using Poisson regression since there is no obvious upper bound to the # of accidents. Also note that the numbers actually mean something and have quantitative value.

Classic example of multinomial regression: response is dietary habits of an individual, Non-vegetarian, Vegetarian or Vegan. Note that the variable in qualitative and not necessarily quantitative. And there are restrictions to what values the response can take (1 of three).

## CLASSIFICATION ALGORITHMS

We can use one of the below algorithms for solving classification problem:

* Logistic Regression (Binary Classification)
* One vs Rest Logistic Regression (Multi Class Classification)
* Naïve Bayes
* K-Nearest Neighbors (K-NN)
* Decision Tree
* Random Forest
* Support Vector Machine (SVM)
* Stochastic Gradient Descent
* Multi Label classification

## CLASSIFICATION LEARNERS

There are two types of learners in classification as lazy learners and eager learners.

### LAZY LEARNERS

Lazy learners simply store the training data and wait until a testing data appear. When it does, classification is conducted based on the most related data in the stored training data. Compared to eager learners, lazy learners have less training time but more time in predicting. Example K-NN.

### EAGER LEARNERS

Eager learners construct a classification model based on the given training data before receiving data for classification. It must be able to commit to a single hypothesis that covers the entire instance space. Due to the model construction, eager learners take a long time for train and less time to predict. Example Decision Tree, Random Forest, Naive Bayes, Artificial Neural Networks.

An eager learner has a model fitting or training step. A lazy learner does not have a training phase.

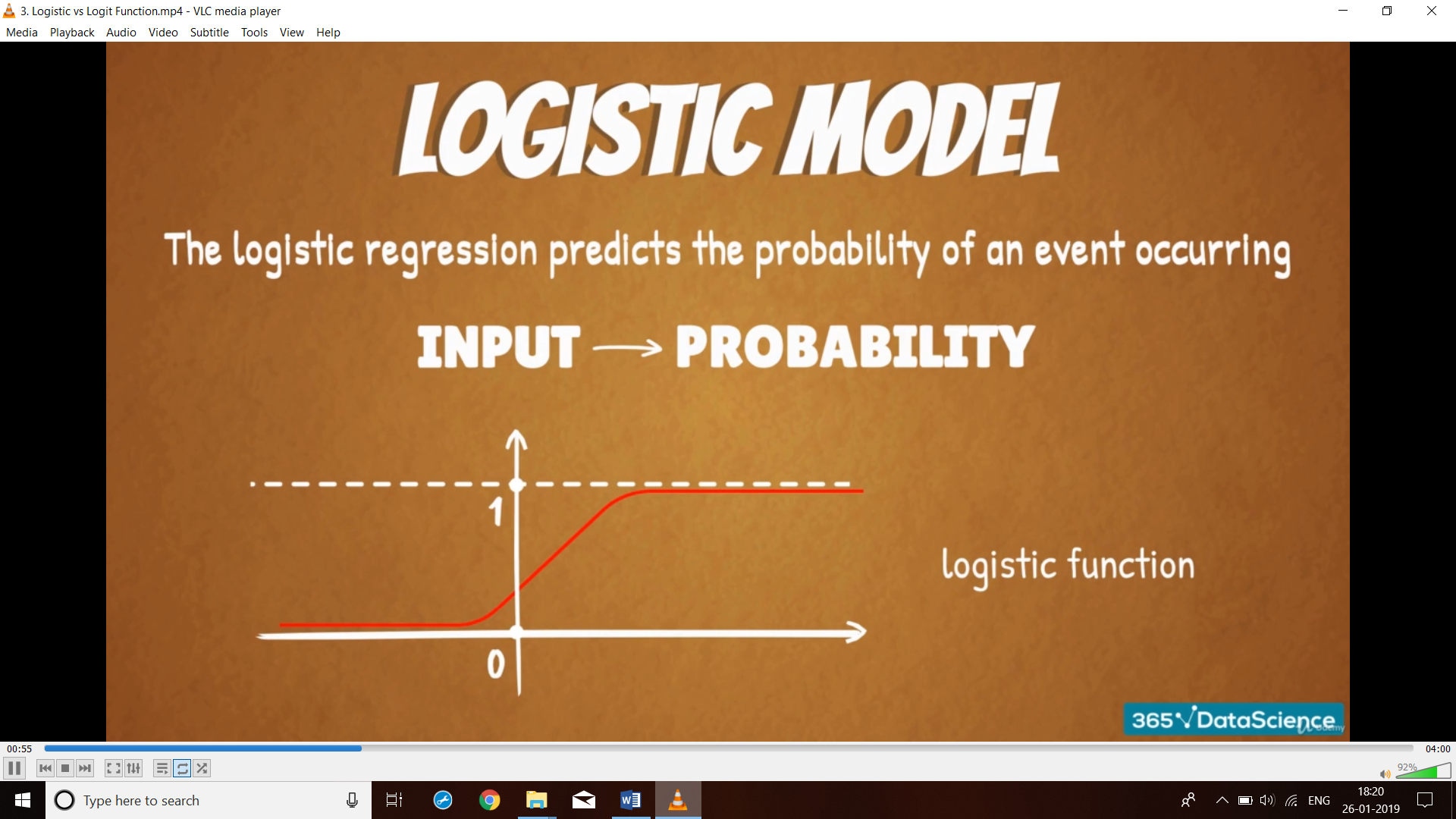
# LOGISTIC REGRESSION

Logistic regression is the most famous machine learning algorithm after linear regression. Logistic regression is basically a supervised classification algorithm. In a classification problem, the target variable (y) can take only discrete values for given set of features (X). Logistic regression is non-linear by definition.

The goal of the algorithm is to model the probability of a random variable being 0 or 1 given experimental data.

The model itself simply models probability of output in terms of input, and does not perform statistical classification (it is not a classifier), though it can be used to make a classifier, for instance by choosing a cutoff value and classifying inputs with probability greater than the cutoff as one class, below the cutoff as the other this is a common way to make a binary classifier. The coefficients are generally not computed by a closed-form expression, unlike linear least squares.

We take our likelihood function assuming that all the observations in the sample are independently Bernoulli distributed. This leads to the intuition that by maximizing the log-likelihood of a model, you are minimizing the KL divergence of your model.



Some important points about Logistic regression to consider:

1. Does NOT assume a linear relationship between the dependent variable and the independent variables, but it does assume linear relationship between the logit of the explanatory variables and the response.
2. Independent variables can be even the power terms or some other nonlinear transformations of the original independent variables.
3. The dependent variable does NOT need to be normally distributed, but it typically assumes a distribution from an exponential family (e.g. binomial, Poisson, multinomial, normal). Binary logistic regression assumes binomial distribution of the response.
4. The homogeneity of variance does NOT need to be satisfied.
5. Errors need to be independent but NOT normally distributed.
6. It uses maximum likelihood estimation (MLE) rather than ordinary least squares (OLS) to estimate the parameters, and thus relies on large-sample approximations.

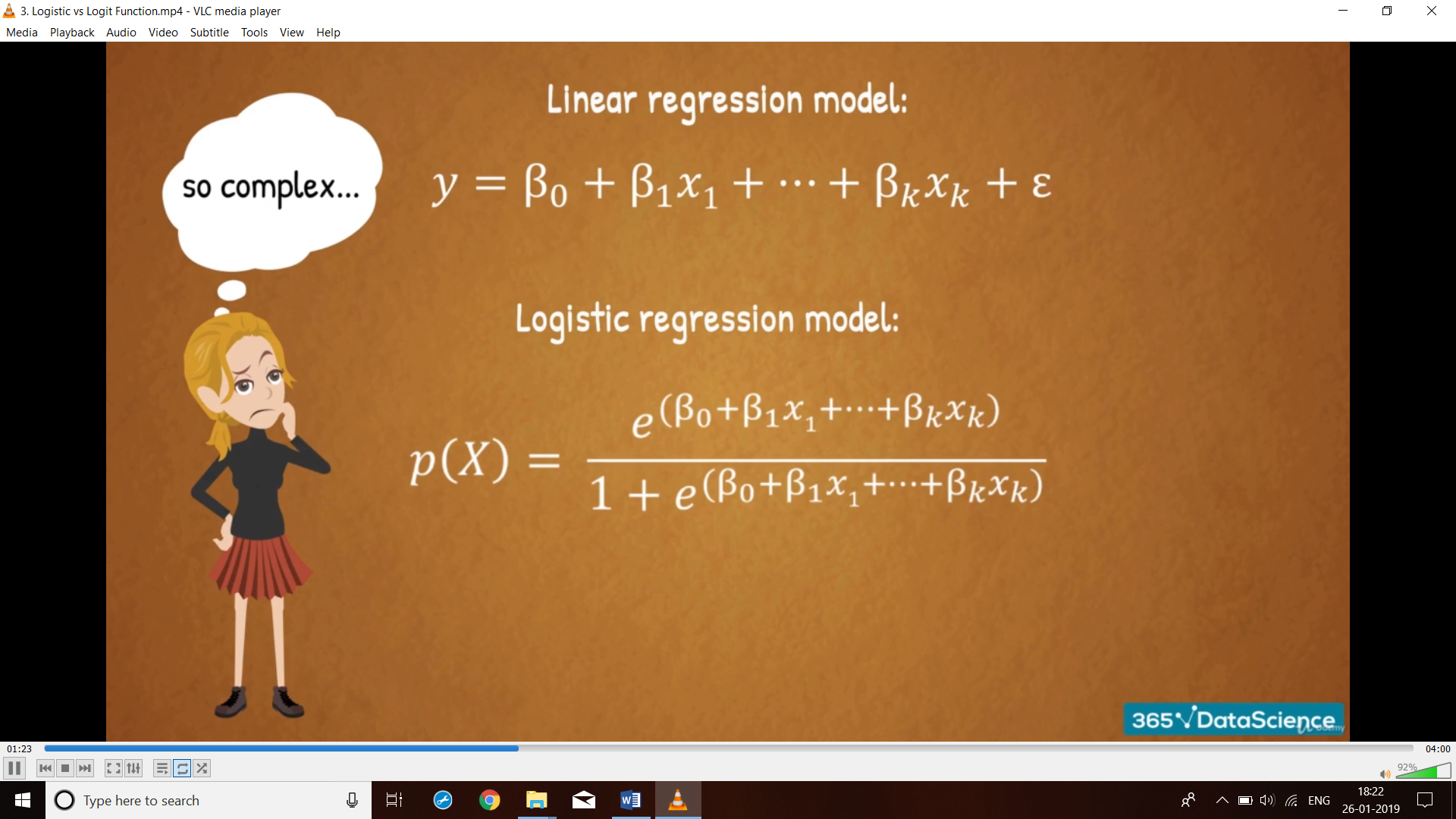
## COMPARISON TO LINEAR REGRESSION

Given data on time spent studying and exam scores. Linear Regression and logistic regression can predict different things:

**Linear Regression** could help us predict the student’s test score on a scale of 0 - 100. Linear regression predictions are continuous (numbers in a range).

**Logistic Regression** could help use predict whether the student passed or failed. Logistic regression predictions are discrete (only specific values or categories are allowed). We can also view probability scores underlying the model’s classifications.

Unlike linear regression, the prediction for the output is transformed using a non-linear function called the logistic function (aka sigmoid function or logit function).

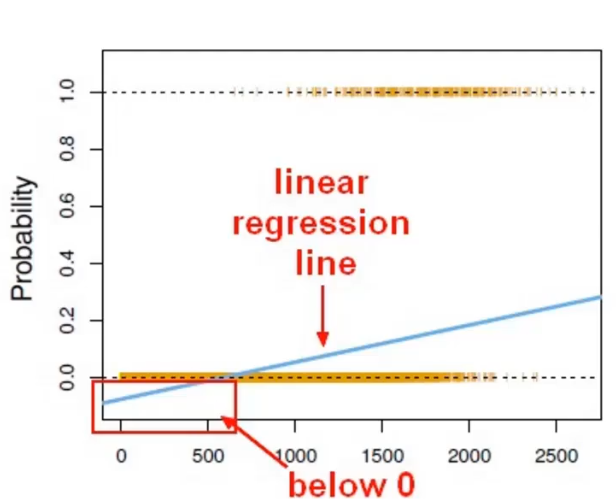


## PROBLEM WITH LINEAR REGRESSION FOR CLASSIFICATION

The linear regression model can work well for regression, but fails for classification. A linear model does not output probabilities, but it treats the classes as numbers (0 and 1) and fits the best line / hyperplane that minimizes the distances between the points and the hyperplane. So, it simply interpolates between the points and you cannot interpret it as probabilities.

A linear model also gives output values below zero and above one. This is a good sign that there might be a smarter approach to classification.

Since the predicted outcome is not a probability, but a linear interpolation between points, there is no meaningful threshold at which you can distinguish one class from the other.

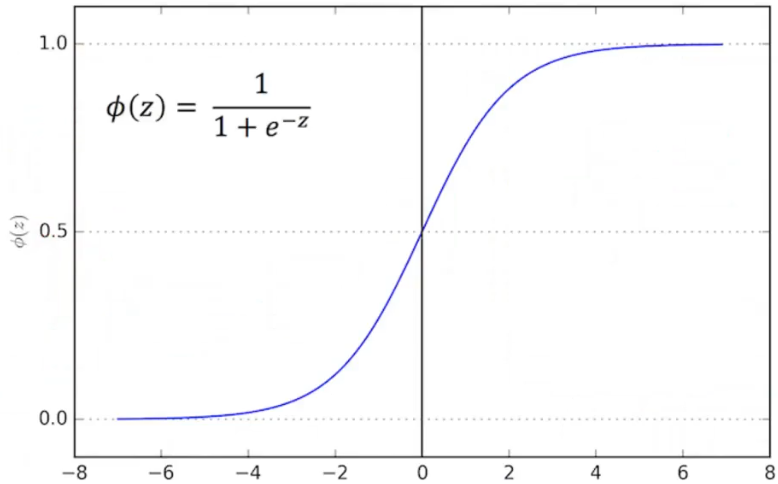




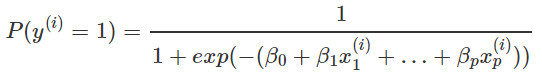
As it can be seen above, we cannot use a normal linear model on binary groups as it leads to a bad fit and we will end up predicting probabilities below 0% which does not make sense. Hence, we will transform our linear regression to logistic regression curve using a sigmoid function which always take values between 0 and 1 as shown above in the figure.

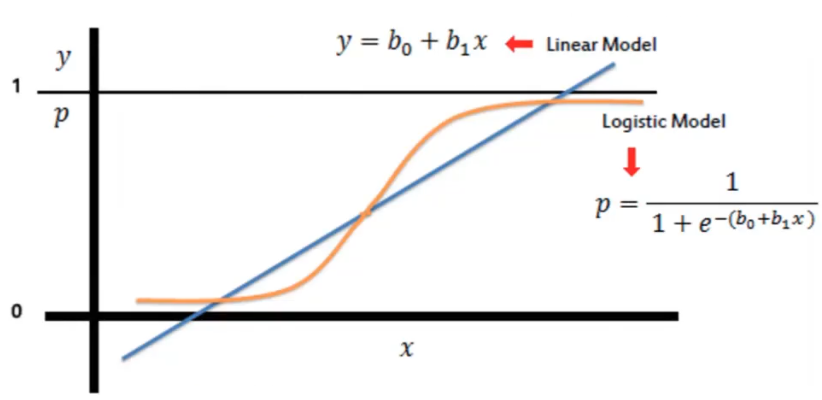
## LOGISTIC FUNCTION

Logistic function (aka Sigmoid function) takes any real value as input and outputs a value between 0 and 1.



For classification, we prefer probabilities between 0 and 1, so we wrap the right side of the equation into the logistic function. This forces the output to assume only values between 0 and 1.





This results in a probability from 0 to 1 of belonging in the one class. We can also set a cut-off point (threshold) say at 0.5

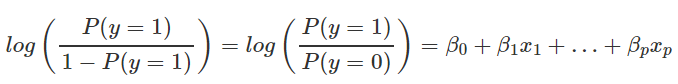
* Anything below it results in class 0.
* Anything above it results in class 1.

This is a non-linear form of logistic regression equation and interpretation of the coefficients (weights) is different from linear regression as the output now is probability instead of a continuous number. Therefore, we need to reformulate the equation for the interpretation so that only the linear term is on the right side of the formula. This is done by calculating odds ratio.

## LOG ODDS (LOGIT)

Odds ratio is defined as the ratio between the likelihood of an event occurring to likelihood of the event not occurring.

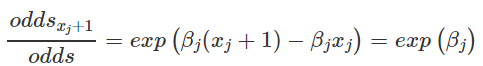
By doing some shuffling of the terms we have changed the Logistic equation to have linear term on the right side which is as below:



We call the term in the log () function “**odds**” (probability of event divided by probability of no event) and wrapped in the logarithm it is called **log odds (logit)**.

We can further simplify the equation to have below format:

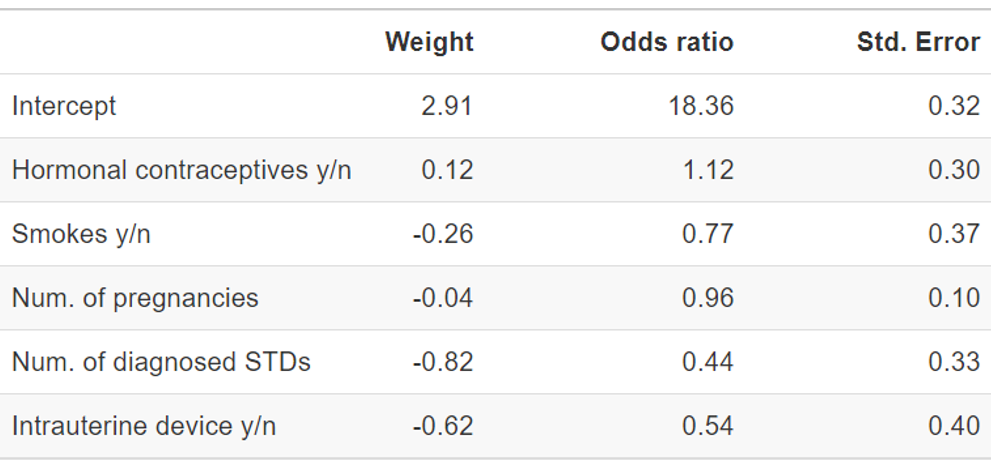
Now it will be easy to interpret the coefficients of classification.



For example, if you have odds of 2, it means that the probability for y=1 is twice as high as y=0.

## COEFFICIENT INTERPRETATION

For Logistic classification the interpretation of the weights is multiplicative and not additive like linear regression. Consider below example:



**Interpretation of a numerical feature** (“Num. of diagnosed STDs”): An increase in the number of diagnosed STDs changes (decreases) the odds of cancer vs. no cancer by a factor of 0.44 when all other features remain the same.

**Interpretation of a categorical feature** (“Hormonal contraceptives y/n”): For women using hormonal contraceptives, the odds for cancer vs. no cancer are by a factor of 1.12 higher, compared to women without hormonal contraceptives, given all other features stay the same.

Like in the linear model, the interpretations always come with the clause that ‘all other features stay the same’.

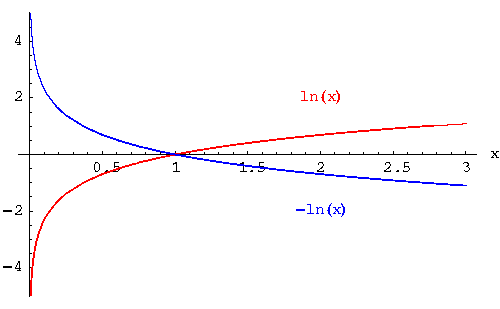
## LOGARITHMIC LOSS

Log Loss quantifies the accuracy of a classifier by penalising false classifications. Minimizing the Log Loss is basically equivalent to maximizing the accuracy of the classifier.

It's hard to interpret raw log-loss values, but log-loss is still a good metric for comparing models. For any given problem, a lower log-loss value means better predictions.

Log Loss heavily penalises classifiers for doing a wrong prediction. For example, if for a particular observation, the classifier assigns a very small probability to the correct class then the corresponding contribution to the Log Loss will be very large indeed.

Log Loss is a slight twist on something called the Likelihood Function. In fact, Log Loss is -1 \* the log of the likelihood function. So, we will start by understanding the likelihood function.



### LIKELIHOOD FUNCTION

A model predicts probabilities of [0.8, 0.4, 0.1] for three houses. The first two houses were sold, and the last one was not sold. So, the actual outcomes could be represented numerically as [1, 1, 0].

Let's step through these predictions one at a time to iteratively calculate the likelihood function. The first house sold, and the model said that was 80% likely. So, the likelihood function after looking at one prediction is 0.8.

The second house sold, and the model said that was 40% likely. There is a rule of probability that the probability of multiple independent events is the product of their individual probabilities. So, we get the combined likelihood from the first two predictions by multiplying their associated probabilities. That is 0.8 \* 0.4, which happens to be 0.32.

Now we get to our third prediction. That home did not sell. The model said it was 10% likely to sell. That means it was 90% likely to not sell. So, the observed outcome of not selling was 90% likely according to the model. So, we multiply the previous result of 0.32 by 0.9.

We could step through all of our predictions. Each time we'd find the probability associated with the outcome that actually occurred, and we'd multiply that by the previous result. That's the likelihood.

### FROM LIKELIHOOD TO LOG LOSS

Each prediction is between 0 and 1. If you multiply enough numbers in this range, the result gets so small that computers can't keep track of it. So, as a clever computational trick, we instead keep track of the log of the Likelihood. This is in a range that's easy to keep track of. We multiply this by negative 1 to maintain a common convention that lower loss scores are better.

## MAXIMUM LIKELIHOOD FUNCTION

Likelihood function is a function which estimates how likely it is that the model at hand describes the real underlying relationship of the variables.

The bigger the likelihood function, the higher the probability that our model is correct. MLE tried to maximize the likelihood function.

MLE goes through different values until it finds a model for which the likelihood is highest. When it can no longer improve it, it will just stop the optimization.

The regression coefficients are usually estimated using maximum likelihood estimation. For example, Newton's method.

The process begins with a tentative solution, revises it slightly to see if it can be improved, and repeats this revision until no more improvement is made, at which point the process is said to have converged.

## LIMITATIONS & ADVANTAGES

Logistic regression can suffer from complete separation. If there is a feature that would perfectly separate the two classes, the logistic regression model can no longer be trained. This is because the weight for that feature would not converge, because the optimal weight would be infinite.

This is really a bit unfortunate, because such a feature is really useful. But you do not need machine learning if you have a simple rule that separates both classes. The problem of complete separation can be solved by introducing penalization of the weights or defining a prior probability distribution of weights.

Another disadvantage of the logistic regression model is that the interpretation is more difficult because the interpretation of the weights is multiplicative and not additive.

## SOLVER

It’s the algorithm to use in the optimization problem for logistic regression. Options available are:

* newton-cg
* lbfgs
* liblinear
* sag
* saga

Important points:

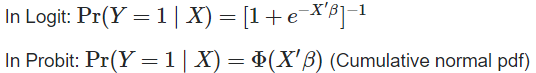
* For small datasets liblinear is a good choice, whereas sag and saga are faster for large ones.
* For multiclass problems only newton-cg, sag, saga and lbfgs handle multinomial loss. liblinear is limited to one-versus-rest schemes.
* newton-cg, lbfgs and sag only handle L2 penalty, whereas liblinear and saga handle L1 penalty.

Note:

* sag and saga fast convergence is only guaranteed on features with approximately the same scale. You can pre-process the data with a scaler from sklearn.preprocessing.
* New in version 0.17: Stochastic Average Gradient descent solver.
* New in version 0.19: SAGA solver.
* Changed in version 0.20: Default will change from ‘liblinear’ to ‘lbfgs’ in 0.22.

# LOGIT vs PROBIT

Both logit and probit models provide statistical models that give the probability that a dependent response variable would be 0 or 1. The difference mainly in the link function.



In other way, logistic has slightly flatter tails. i.e. probit curve approaches the axes more quickly than the curve.

Logit has better interpretation than probit. Logistic regression can be interpreted as modeling log odds. Usually people start the modeling with logit. You could use likelihood value to decide logit or probit.

They are very similar and often given practically identical results, but because they use different functions to calculate the probabilities, their results are sometimes slightly different.

* In most scenarios, the logit and probit models fit the data equally well, with the following two exceptions.
* Logit is definitely better in the case of "extreme independent variables". These are independent variables where one particularly large or small value will overwhelmingly often determine whether the dependent variable is a 0 or a 1, overriding the effects of most other variables
* Probit is better in the case of "random effects models" with moderate or large sample sizes (it is equal to logit for small sample sizes). For fixed effects models, probit and logit are equally good.

# ONE VS ONE CLASSIFIER

Say we have a classification problem and there are N distinct classes. In this case, we’ll have to train a multi-class classifier instead of a binary one.

Here we pick 2 classes at a time, and train a two-class-classifier using samples from the selected two classes only (other samples are ignored in this step). We repeat this for all the two class combinations. So, we end up with N(N-1)/2 classifiers. And while testing we choose the class based on voting among these classifiers.

# ONE VS REST CLASSIFIER

Here we pick one class and train a two-class-classifier with the samples of the selected class on one side and all the other samples on the other side. Thus, you end up with N classifiers. While testing we simply classify the sample as belonging to the class with maximum score among the N classifiers.

This approach is fairly reasonable when the total number of classes is small, but becomes increasingly inefficient as the number of classes rises.

# CLASSIFICATION THRESHOLDING

Logistic regression returns a probability. You can use the returned probability "as is" (for example, the probability that the user will click on this ad is 0.00023) or convert the returned probability to a binary value (for example, this email is spam).

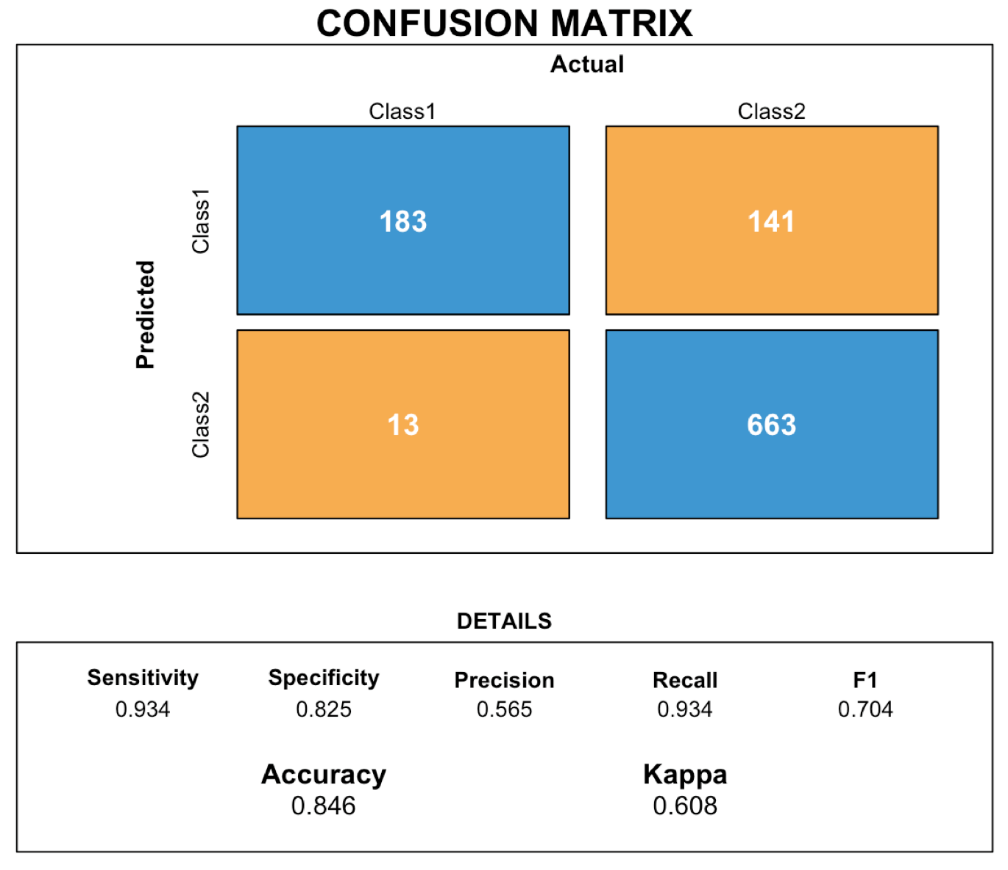
A logistic regression model that returns 0.9995 for a particular email message is predicting that it is very likely to be spam. Conversely, another email message with a prediction score of 0.0003 on that same logistic regression model is very likely not spam. However, what about an email message with a prediction score of 0.6?

In order to map a logistic regression value to a binary category, you must define a classification threshold (also called the decision threshold). A value above that threshold indicates "spam"; a value below indicates "not spam." It is tempting to assume that the classification threshold should always be 0.5, but thresholds are problem-dependent, and are therefore values that you must tune.

"Tuning" a threshold for logistic regression is different from tuning hyperparameters such as learning rate. Part of choosing a threshold is assessing how much you'll suffer for making a mistake. For example, mistakenly labeling a non-spam message as spam is very bad. However, mistakenly labeling a spam message as non-spam is unpleasant, but hardly the end of your job.

# CONFUSION MATRIX

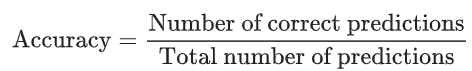
A confusion matrix is an N X N matrix, where N is the number of classes being predicted. For example, problem we have N=2 (class 1 and 0) and hence we get a 2 X 2 matrix.



On the good side, the logistic regression model is not only a classification model, but also gives you probabilities. This is a big advantage over models that can only provide the final classification. Knowing that an instance has a 99% probability for a class compared to 51% makes a big difference.

# CLASSIFICATION ACCURACY

Accuracy is one metric for evaluating classification models. Accuracy is the fraction of predictions our model got right. Accuracy has the following definition:



For binary classification, accuracy can also be calculated in terms of positives and negatives as follows:

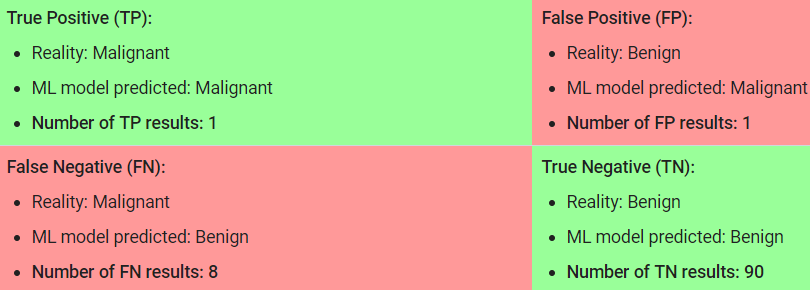


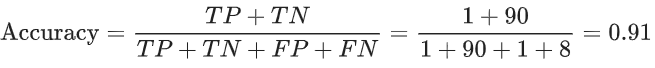
Where TP = True Positives, TN = True Negatives, FP = False Positives, and FN = False Negatives.

A **true positive** is an outcome where the model correctly predicts the positive class. Similarly, a **true negative** is an outcome where the model correctly predicts the negative class.

A **false positive** is an outcome where the model incorrectly predicts the positive class. And a **false negative** is an outcome where the model incorrectly predicts the negative class.

Let's try calculating accuracy for the following model that classified 100 tumors as malignant (the positive class) or benign (the negative class):





Accuracy comes out to 0.91, or 91% (91 correct predictions out of 100 total examples). That means our tumor classifier is doing a great job of identifying malignancies, right?

Actually, let's do a closer analysis of positives and negatives to gain more insight into our model's performance. Of the 100 tumor examples, 91 are benign (90 TNs and 1 FP) and 9 are malignant (1 TP and 8 FNs).

Of the 91 benign tumors, the model correctly identifies 90 as benign. That's good. However, of the 9 malignant tumors, the model only correctly identifies 1 as malignant—a terrible outcome, as 8 out of 9 malignancies go undiagnosed!

Accuracy alone doesn't tell the full story when you're working with a class-imbalanced dataset where there is a significant disparity between the number of positive and negative labels.

# PRECISION SENSITIVITY AND SPECIFICITY

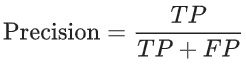
Since accuracy is not a good measure of accuracy for class imbalance imbalanced datasets, we can evaluate the model using precision and recall

## PRECISION

Precision attempts to answer the following question

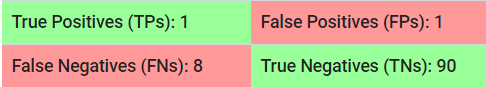


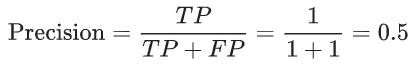
Precision is defined as follows:



Example, percentage of sick people that were correctly identified as having the condition.

Let's calculate precision for our ML model from the previous example:





Our model has a precision of 0.5—in other words, when it predicts a tumor is malignant, it is correct 50% of the time.

## RECALL

Recall is also known as Sensitivity (true positive rate). Recall attempts to answer the following question:

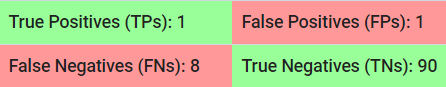


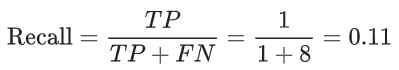
Recall is defined as:



Example, percentage of actual sick people who are correctly identified as having the condition.

Let's calculate recall for our tumor classifier:

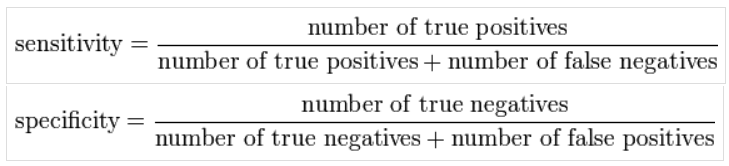




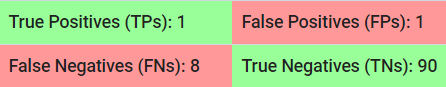
Our model has a recall of 0.11—in other words, it correctly identifies 11% of all malignant tumors.

## SPECIFICITY

Specificity (the true negative rate) measures the proportion of actual negatives that are correctly identified as such. Example the percentage of healthy people who are correctly identified as not having the condition.



Let's calculate specificity for our tumor classifier:



Specificity = 1 – Sensitivity = 1- 0.11 = 0.89

Our model has a Specificity (TNR) of 0.89 —in other words, it correctly identifies 89% of all benign tumors.

## PRACTICAL EXAMPLE

In airport security tests of passengers is carried out for potential threats to safety. In this case our positive class (class 1) will be highly dangerous objects and class 0 will be other metal objects.

Scanners may be set to trigger alarms on low-risk items like belt buckles and keys (low specificity) in order to increase the probability of identifying dangerous objects and minimize the risk of missing objects that do pose a threat (high sensitivity).

This trade-off can be represented graphically using a receiver operating characteristic curve. A perfect predictor would be described as 100% sensitive, meaning all sick individuals are correctly identified as sick, and 100% specific, meaning healthy individuals are correctly identified as healthy.

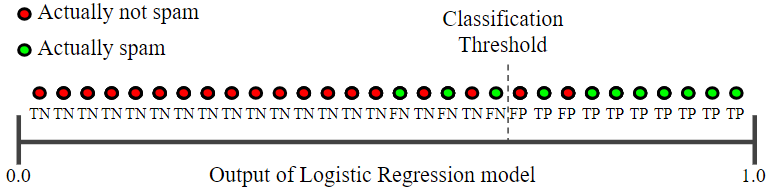
# TRADE OFF BETWEEN PRECISION AND RECALL

To fully evaluate the effectiveness of a model, you must examine both precision and recall. Unfortunately, precision and recall are often in tension. That is, improving precision typically reduces recall and vice versa.

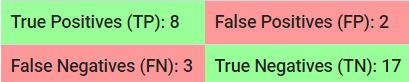
## DEFAUT THRESHOLD

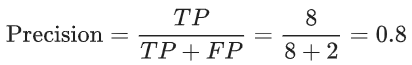
Following figure shows 30 predictions made by an email classification model. Here class 1 is spam and class 0 is not spam.

Those to the right of the classification threshold are classified as "spam", while those to the left are classified as "not spam."



Precision measures the percentage of emails flagged as spam that were correctly classified—that is, the percentage of dots to the right of the threshold line that are green in Figure 1:





Since precision of the model is 0.8, we can say that when our model classifies an email as spam its correct 80% of the times.

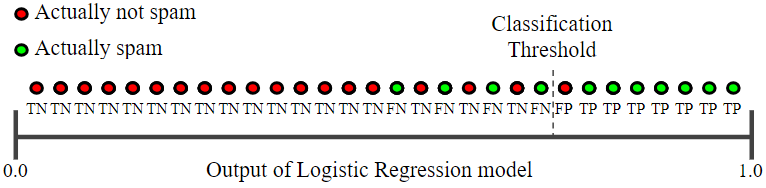
Recall measures the percentage of actual spam emails that were correctly classified—that is, the percentage of green dots that are to the right of the threshold line in Figure 1:

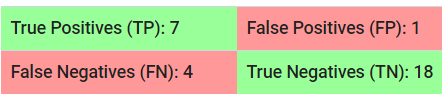


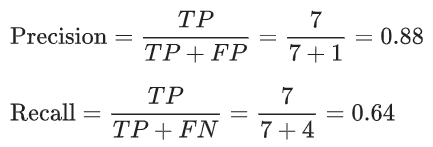
Since Recall of the model is 0.73, we can say that our model has correctly identified 73% of spam emails.

## INCREASING CLASSIFICATION THRESHOLD

The number of false positives decreases, but false negatives increase. As a result, precision increases, while recall decreases:



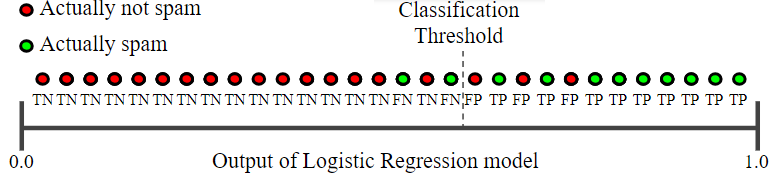




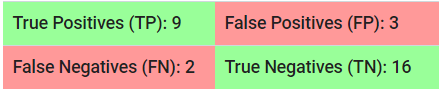
Now since Precision is 0.88, we can say that when the model classifies an email as spam its correct 88% pf the times. And since Recall is 0.64, we can say that our model has correctly identified 64% of the spam emails.

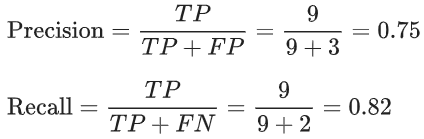
## DECREASING CLASSIFICATION THRESHOLD.

Below figure shows the effect of decreasing the classification threshold:



False positives increase, and false negatives decrease. As a result, this time, precision decreases and recall increases:





Now since Precision is 0.75, we can say that when the model classifies an email as spam its correct 75% pf the times. And since Recall is 0.82, we can say that our model has correctly identified 82% of the spam emails. Various metrics have been developed that rely on both precision and recall, one of them is F1 score.

## CONCLUSION

In general, a model that outperforms another model on both precision and recall is likely the better model. Obviously, we'll need to make sure that comparison is being done at a precision / recall point that is useful in practice for this to be meaningful.

For example, suppose our spam detection model needs to have at least 90% precision to be useful and avoid unnecessary false alarms. In this case, comparing one model at {20% precision, 99% recall} to another at {15% precision, 98% recall} is not particularly instructive, as neither model meets the 90% precision requirement. But with that caveat in mind, this is a good way to think about comparing models when using precision and recall.

# RECEIVER OPERATING CHARACTERISTIC CURVE

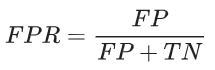
A ROC curve is a graph showing the performance of a classification model at various classification thresholds. This curve plots two parameters:

* True Positive Rate
* False Positive Rate

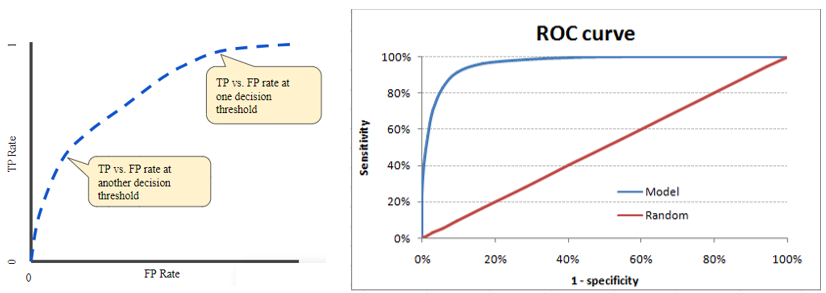
True Positive Rate (TPR) is a synonym for recall and is therefore defined as follows:



False Positive Rate (FPR) is defined as follows:



An ROC curve plots TPR vs. FPR at different classification thresholds. Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives. The following figure shows a typical ROC curve.

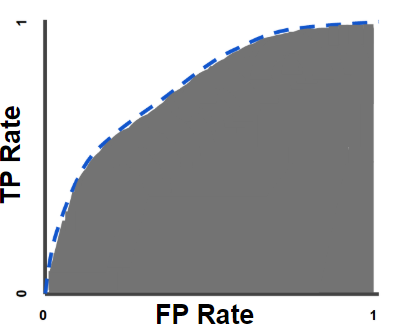


The biggest advantage of using ROC curve is that it is independent of the change in percentage of responders. A ROC curve plots the performance of a binary classifier under various threshold settings.

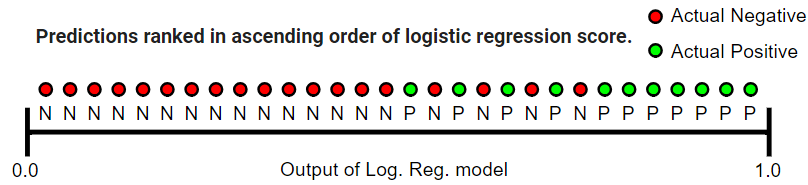
* A perfect classifier has a 100% true positive rate and 0% false positive rate, so its ROC curve passes through the upper left corner of the square.
* A completely random classifier (predicting “true” with probability p and “false” with probability 1-p for all inputs) will by random chance correctly classify proportion p of the actual true values and incorrectly classify proportion p of the false values, so it’s true and false positive rates are both p. Therefore, a completely random classifier’s ROC curve is a straight line through the diagonal of the plot.

## AUC

Area under ROC Curve (or AUC for short) is a performance metric for binary classification problems. If I randomly pick one positive example and one negative example from the distribution, what is the probability that our model will correctly assign a higher score to the positive than it does to the negative. This probability values is given by AUC score.



For example, given the following examples, which are arranged from left to right in ascending order of logistic regression predictions:



AUC represents the probability that a random positive (green) example is positioned to the right of a random negative (red) example.

AUC is desirable for the following two reasons:

* AUC is scale-independent. It measures how well predictions are ranked, rather than their absolute values.
* AUC is classification-threshold-independent. It measures the quality of the model's predictions irrespective of what classification threshold is chosen.

Note that the area of entire square is 1\*1 = 1. Hence AUC itself is the ratio under the curve and the total area. Following are a few thumb rules:

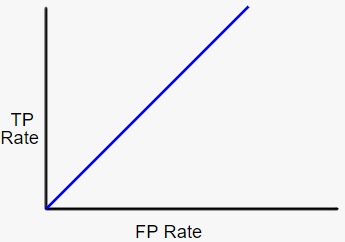
* 0.9 - 1 = excellent (A)
* 0.8 - 0.9 = good (B)
* 0.7 - 0.8 = fair (C)
* 0.6 - 0.7 = poor (D)
* 0.5 - 0.6 = fail (F)

AUC ranges in value from 0 to 1. A model whose predictions are 100% wrong has an AUC of 0.0; one whose predictions are 100% correct has an AUC of 1.0. An area of 0.5 represents a model as good as random.

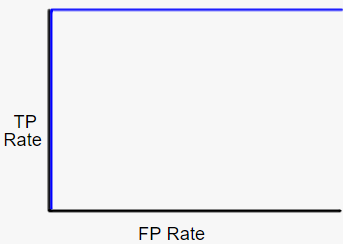
AUC is based on the relative predictions, so any transformation of the predictions that preserves the relative ranking has no effect on AUC. This is clearly not the case for other metrics such as squared error, log loss, or prediction bias.

## EXMAPLE OF AUC CURVES

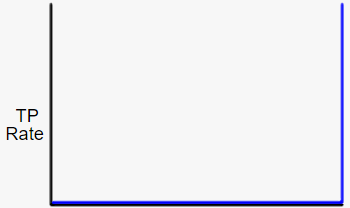
This ROC curve has an AUC of 0.5, meaning it ranks a random positive example higher than a random negative example 50% of the time. As such, the corresponding classification model is basically worthless, as its predictive ability is no better than random guessing.



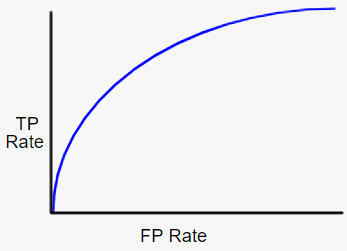
This is the best possible ROC curve, as it ranks all positives above all negatives. It has an AUC of 1.0.



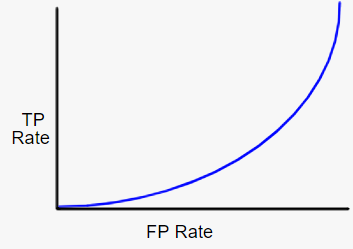
This is the worst possible ROC curve; it ranks all negatives above all positives, and has an AUC of 0.0. If you were to reverse every prediction (flip negatives to positives and positives to negatives), you'd actually have a perfect classifier!



This ROC curve has an AUC between 0.5 and 1.0, meaning it ranks a random positive example higher than a random negative example more than 50% of the time. Real-world binary classification AUC values generally fall into this range.



This ROC curve has an AUC between 0 and 0.5, meaning it ranks a random positive example higher than a random negative example less than 50% of the time. The corresponding model actually performs worse than random guessing! If you see an ROC curve like this, it likely indicates there's a bug in your data.



# GINI COEFFICIENT

Gini is impurity measure. Gini coefficient can be straight away derived from the AUC ROC number. Gini is the ratio between area between the ROC curve and the diagonal line & the area of the above triangle. Gini above 60% is a good model. Following is the formulae used:

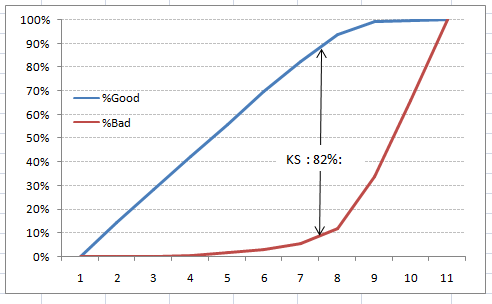
Gini = 2\*AUC – 1

# KOLOMOGOROV SMIRNOV CHART

K-S or Kolmogorov-Smirnov chart measures performance of classification models. More accurately K-S is a measure of the degree of separation between the positive and negative distributions.

The K-S is 100, if the scores partition the population into two separate groups in which one group contains all the positives and the other all the negatives.

On the other hand, If the model cannot differentiate between positives and negatives, then it is as if the model selects cases randomly from the population. The K-S would be 0. In most classification models the K-S will fall between 0 and 100, and that the higher the value the better the model is at separating the positive from negative cases.



# SOMER’S D

Somers’ D is a measure of ordinal association between two possibly dependent random variables X and Y (binary choice). Somers’ D takes values between -1 when all pairs of the variables disagree and 1 when all pairs of the variables agree.

Somers’ D plays a central role in rank statistics and is the parameter behind many nonparametric methods. It is also used as a quality measure of binary choice or ordinal regression (e.g., logistic regressions) and credit scoring models.

## CONCORDANT vs DISCORDANT

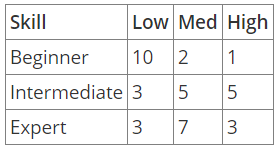
Concordance is just one of the measures to see the goodness of Logistic Regression. It alone cannot tell you much about how good the model is. You need to use it along with other measure to see how good a model is.

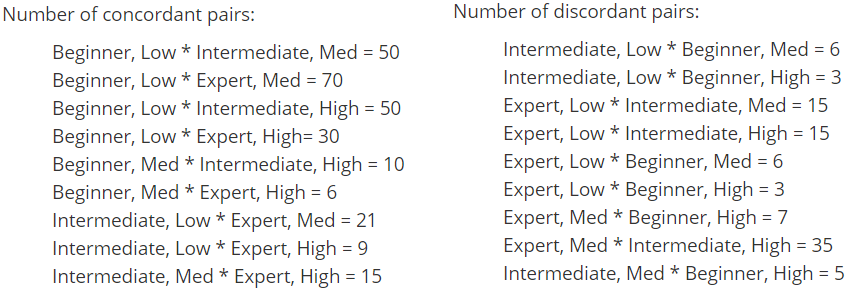
We say that two pairs (Xi, Yi) and (Xj, Yj) are concordant if the ranks of both elements agree i.e. Xi >Xj and Yi >Yj or Xi <Xj and Yi <Yj

We say that two pairs are discordant, if the ranks of both elements disagree, i.e. if Xi >Xj and Yi <Yj or Xi <Xj and Yi >Yj.

If Xi = Xj or Yi = Yj the pair is neither concordant nor discordant.

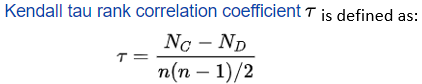
For example: Suppose a group of friends are playing darts. They identify their skill level as beginner, intermediate, or expert, and collect data on their accuracy - low, medium, and high





## KENDALL TAU RANK CORRELATION COEFFICIENT





Where NC is the number of concordant pairs, ND are number of discordant pairs & n is the sample size.

## KENDALL TAU FOR DISTRIBUTION



Again, Somers’ D, which measures ordinal association of random variables X and Y in PXY can be defined through Kendall's tau.

Tau is defined as or the difference between the probabilities of concordance and discordance or as below:



## SOMER’S D FOR SAMPLE

Somers’ D of Y with respect to X is defined as



Note that Kendall's tau is symmetric in X and Y, whereas Somers’ D is asymmetric in X and Y a quantifies the number of pairs with unequal X’s values.

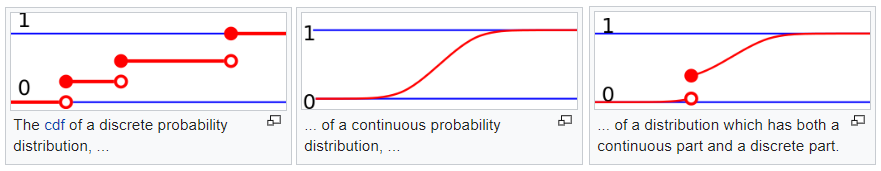
Somers’ D is the difference between the number of concordant and discordant pairs, divided by the number of pairs with X values in the pair being unequal.

## SOMER’S D FOR DISTRIBUTION

Somers’ D of Y with respect to X is defined as



Thus, DYX is the difference between the two corresponding probabilities, conditional on the X values not being equal.



If X has a continuous probability distribution, then tau (X, X) = 1 and Kendall's tau and Somers’ D coincide.

If X and Y are both binary with values 0 and 1, then Somers’ D is the difference between two probabilities:



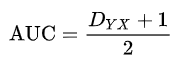
## SOMERS' D FOR BINARY DEPENDENT VARIABLES

Somers' D is most often used when the dependent variable Y is a binary variable, i.e. for binary classification or prediction of binary outcomes. Methods for fitting such models include logistic and probit regression.

Several statistics can be used to quantify the quality of such models:

* Area under the receiver operating characteristic (ROC) curve.
* Goodman and Kruskal's gamma
* Kendall's tau (Tau-a)
* Somers’ D

Somers’ D is probably the most widely used of the available ordinal association statistics. If there are no ties on independent variable, Somers’ D is related to the area under the receiver operating characteristic curve (AUC)



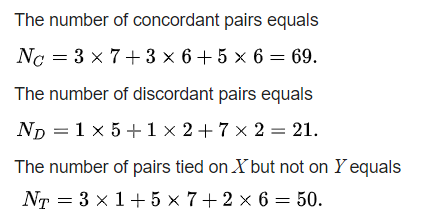
In the case where the independent (predictor) variable Y is discrete and the dependent (outcome) variable X is binary, Somers’ D equals

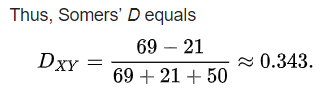




Example: Suppose that the predictor variable X takes three values, 0.25, 0.5, or 0.75, and outcome variable Y takes two values, 0 or 1. The table below contains observed combinations of X and Y:







# PREDICTION BIAS

Logistic regression predictions should be unbiased. That is:



Prediction bias is a quantity that measures how far apart those two averages are. That is:



A significant nonzero prediction bias tells you there is a bug somewhere in your model, as it indicates that the model is wrong about how frequently positive labels occur.

For example, let's say we know that on average, 1% of all emails are spam. If we don't know anything at all about a given email, we should predict that it's 1% likely to be spam. Similarly, a good spam model should predict on average that emails are 1% likely to be spam. (In other words, if we average the predicted likelihoods of each individual email being spam, the result should be 1%.) If instead, the model's average prediction is 20% likelihood of being spam, we can conclude that it exhibits prediction bias.

Possible root causes of prediction bias are:

* Incomplete feature set
* Noisy data set
* Buggy pipeline
* Biased training sample
* Overly strong regularization

A good model will usually have near-zero bias. That said, a low prediction bias does not prove that your model is good. A really terrible model could have a zero-prediction bias. For example, a model that just predicts the mean value for all examples would be a bad model, despite having zero bias.

# CLASSIFICATION REGULARIZATION

Regularization is extremely important in logistic regression modeling. Without regularization, the asymptotic nature of logistic regression would keep driving loss towards 0 in high dimensions. Consequently, most logistic regression models use one of the following two strategies to reduce model complexity:

* L1 regularization tends to reduce the number of features. In other words, L1 regularization often reduces the model size.
* L2 regularization rarely reduces the number of features. In other words, L2 regularization rarely reduces the model size.
* Early stopping, that is, limiting the number of training steps or the learning rate.

# CONVERGENCE

Machine learning algorithms tend to work by examining huge amounts of data to generate a rule on how to judge certain things. Most often these are used as classifiers

For example, you could give a machine learning algorithm a large set of data of patients and tell it which patients had heart disease and which didn't. The algorithm will then look for patterns in the data, generating rules on those patterns. If you're such and such a height and weight and experience this symptom then you have heart disease.

It will then apply these patterns to unknown data and classify it for you. You trained the algorithm with the batch of known data, then give it a new example. Based on the rules it created it will tell you whether the new example is a heart disease candidate or not.

To generate these rules the algorithm tends to do many things at the same time with the data. Most machine learning algorithms are actually SEVERAL algorithms. Let's say the algorithm has 10 variant algorithms inside. Each of those 10 algorithms looks for rules. Any rule that is discovered by 3 or more algorithms is considered "Good". Any rule that is discovered by only 1 or 2 algorithms isn't considered strong enough for use.

This is called **CONVERGENCE**. Multiple algorithms are converging on the same solution. Poor convergence means the machine learning algorithm takes a long time to train, and needs more data to do it.

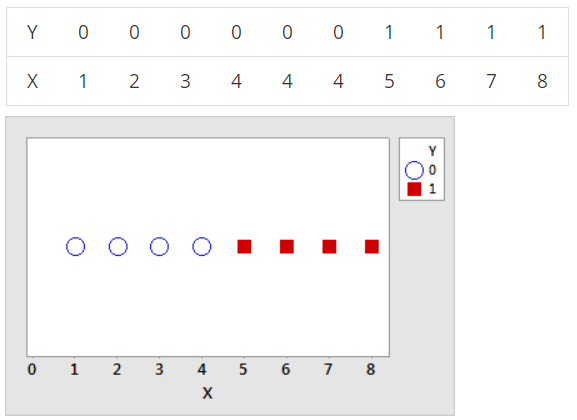
A model is guaranteed to converge if there is no way the error will increase after an iteration. If you can prove that, you have proved that the algorithm will converge. If you can't prove that, it doesn't mean the algorithm won't converge, but it will be much more difficult to show that it will.

In some instances, the model may not reach convergence. Nonconvergence of a model indicates that the coefficients are not meaningful because the iterative process was unable to find appropriate solutions. A failure to converge may occur for a number of reasons:

* Complete separation
* Having a large ratio of predictors to cases
* Multicollinearity
* Sparseness
* Very small sample set for training

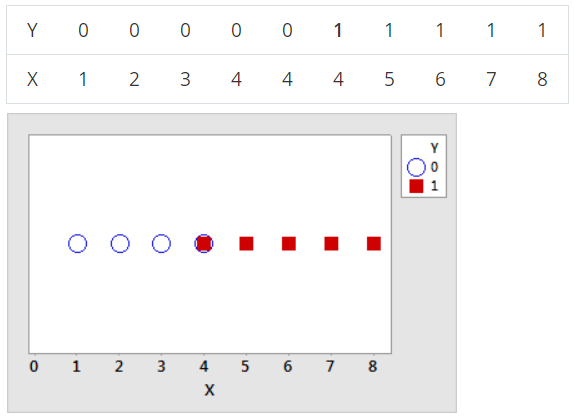
## COMPLETE SEPARATION

Complete separation occurs when a linear combination of the predictors yields a perfect prediction of the response variable. For example, in the following data set if X ≤ 4 then Y = 0. If X > 4 then Y = 1.



## QUASI-COMPLETE SEPARATION

Quasi-complete separation is similar to complete separation. The predictors yield a perfect prediction of the response variable for most values of the predictors, but not all. For example, in the previous data set, for one of the values where X = 4, let Y = 1 instead of 0. Now, if X < 4 then Y = 0, if X > 4 then Y = 1, but if X = 4 then Y could be 0 or 1. This overlap in the middle range of the data makes the separation quasi-complete.



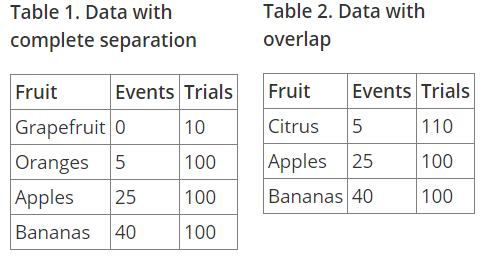
## CAUSES AND REMEDIATION

Often, separation occurs when the data set is too small to observe events with low probabilities. The more predictors are in the model, the more likely separation is to occur because the individual groups in the data have smaller sample sizes. In Minitab, the model can also fail to converge for very large or very small probabilities that are not strictly 0 or 1, such as less than 1 out of 1 trillion.

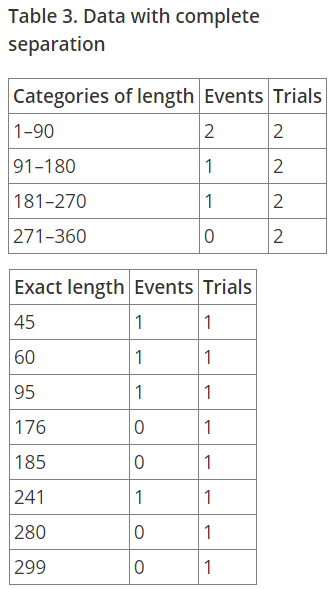
Although Minitab prints a warning when it detects separation, the more predictors are in the model the more difficult the identification of the cause of the separation is. The inclusion of interaction terms in the model makes the difficulty even greater.

When the maximum likelihood estimates fail to converge because of separation, consider the following 5 strategies:

1. Increase the amount of data. Separation often occurs when there is a category or range of a predictor with only one value of the response. A larger sample size increases the probability of different values for the response.
2. Consider what the separation means. While complete separation and quasi-complete separation can indicate that the sample size is too small, they can also indicate important relationships. If the true probability of an event at a particular level or combination of levels is close to 0 or 1, this information is important.
3. Consider an alternative model. The more terms are in the model, the more likely that separation occurs for at least one variable. When you select terms for the model, you can check whether the exclusion of a term allows the maximum likelihood estimates to converge. If a useful model exists that does not use the term, you can continue the analysis with the new model.
4. Check to see whether you can combine categories in problematic variables. If there are categories that are sensible to combine, the separation can disappear from the data set. For example, suppose “Fruit” is a variable in the model. “Grapefruit” has no events because of the small number of trials. Combining “Grapefruit” and “Oranges” into the category “Citrus” eliminates the separation.



1. Check to see whether a problematic categorical variable is an aggregated variable. If the relationship of the unaggregated variable to the response does not show complete separation, the substitution of the numeric data can eliminate the separation. For example, suppose “Length of employment” is an aggregated variable in the model. When the data are in 30-day increments, the lowest level has all events and the highest level has no events, which creates complete separation. The substitution of the number of days into the model eliminates the separation.



1. One cause can be when using several categorical variables whose categories are coded by indicators. For example, if one is studying an age-related disease (present/absent) and age is one of the predictors, there may be subgroups (e.g., women over 55) all of whom have the disease.
2. Complete separation also may occur if there is a coding error or you mistakenly included another version of the outcome as a predictor. For example, we might have classified a continuous variable X into a binary variable Y. We then wanted to study the relationship between Y and some predictor variables. If we would include X as a predictor variable, we would run into the problem of perfect prediction, since by definition, Y separates X completely.
3. In the case of complete separation, make sure that the outcome variable is not a dichotomous version of a variable in the model.
4. If it is quasi-complete separation, the easiest strategy is the "Do nothing" strategy. This is because that the maximum likelihood for other predictor variables are still valid. The drawback is that we don’t get any reasonable estimate for the variable X that actually predicts the outcome variable effectively. This strategy does not work well for the situation of complete separation.
5. Firth logistic regression is another good strategy. It uses a penalized likelihood estimation method.

# WEIGHT OF EVIDENCE

We use weight of evidence in logistic regression.

# LDA

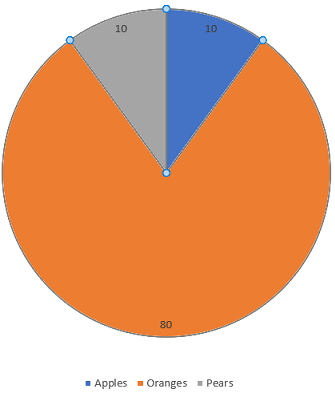
Logistic regression has below drawbacks:

* When classes are well separated, parameters estimate from logistic regression tend to be unstable
* When the data set is small, logistic regression is also unstable
* Not the best to predict more than two classes.

That’s where linear discriminant analysis (LDA) comes in handy. It is more stable than logistic regression and widely used to predict more than two classes.

# UNBALANCED CLASS DATASET

Imbalanced data typically refers to a problem with classification problems where the classes are not represented equally. For example, you may have a 3-class classification problem of set of fruits to classify as oranges, apples or pears with total 100 instances. A total of 80 instances are labeled with Class-1 (Oranges), 10 instances with Class-2 (Apples) and the remaining 10 instances are labeled with Class-3 (Pears).



Most classification data sets do not have exactly equal number of instances in each class, but a small difference often does not matter.

There are problems where a class imbalance is not just common, it is expected. For example, in datasets like those that characterize fraudulent transactions are imbalanced. The vast majority of the transactions will be in the “Not-Fraud” class and a very small minority will be in the “Fraud” class.

There are various techniques involved in improving the performance of imbalanced datasets.

## RE-SAMPLING DATASET

To make our dataset balanced there are two ways to do so:

* Under-sampling: Remove samples from over-represented classes; use this if you have huge dataset
* Over-sampling: Add more samples from under-represented classes; use this if you have small dataset

SMOTE is an over-sampling method. It creates synthetic samples of the minority class. We use imblearn python package to over-sample the minority classes.

## CLASS\_WEIGHT

Since classes are imbalanced, what about providing some bias to minority classes? We can estimate class weights in scikit by using compute\_class\_weight and use the parameter ‘class weight’, while training the model. This can help to provide some bias towards the minority classes while training the model and thus help in improving performance of the model while classifying various classes.

<https://datascience.stackexchange.com/questions/16342/unbalanced-multiclass-data-with-xgboost>



## PRECISION-RECALL CURVES

Precision-Recall is a useful measure of success of prediction when the classes are very imbalanced. Precision is a measure of the ability of a classification model to identify only the relevant data points, while recall is a measure of the ability of a model to find all the relevant cases within a dataset.

The precision-recall curve shows the trade-off between precision and recall for different threshold. A high area under the curve represents both high recall and high precision, where high precision relates to a low false positive rate, and high recall relates to a low false negative rate.

## USE PENALIZED MODELS

Like penalized-SVM and penalized-LDA. They put additional cost on the model for making classification mistakes on the minority class during training. These penalties can bias the model towards paying attention to minority class.

## ANOMALY DETECTION

Try Anomaly Detection techniques and models often used there. Although that would probably be necessary if your data was even more Imbalanced.

# MODEL EVALUATION

Classification problems are perhaps the most common type of machine learning problem and as such there are a myriad of metrics that can be used to evaluate predictions for these problems.

After training the model the most important part is to evaluate the classifier to verify its applicability. We can use below metrics:

1. Classification report
2. Logarithmic loss
3. Confusion matrix (Binary MODEL)
4. Accuracy (Binary Model)
5. Precision and Recall
6. Receiver Operating Characteristics (ROC) (Binary Classification)
7. AUC (Binary Classification)
8. GINNI score (Binary Classification)
9. Kolmogorov Smirnov
10. Concordant ratio
11. Somer’s D

# STACKING CLASSIFIERS

<http://rasbt.github.io/mlxtend/user_guide/classifier/StackingClassifier/>

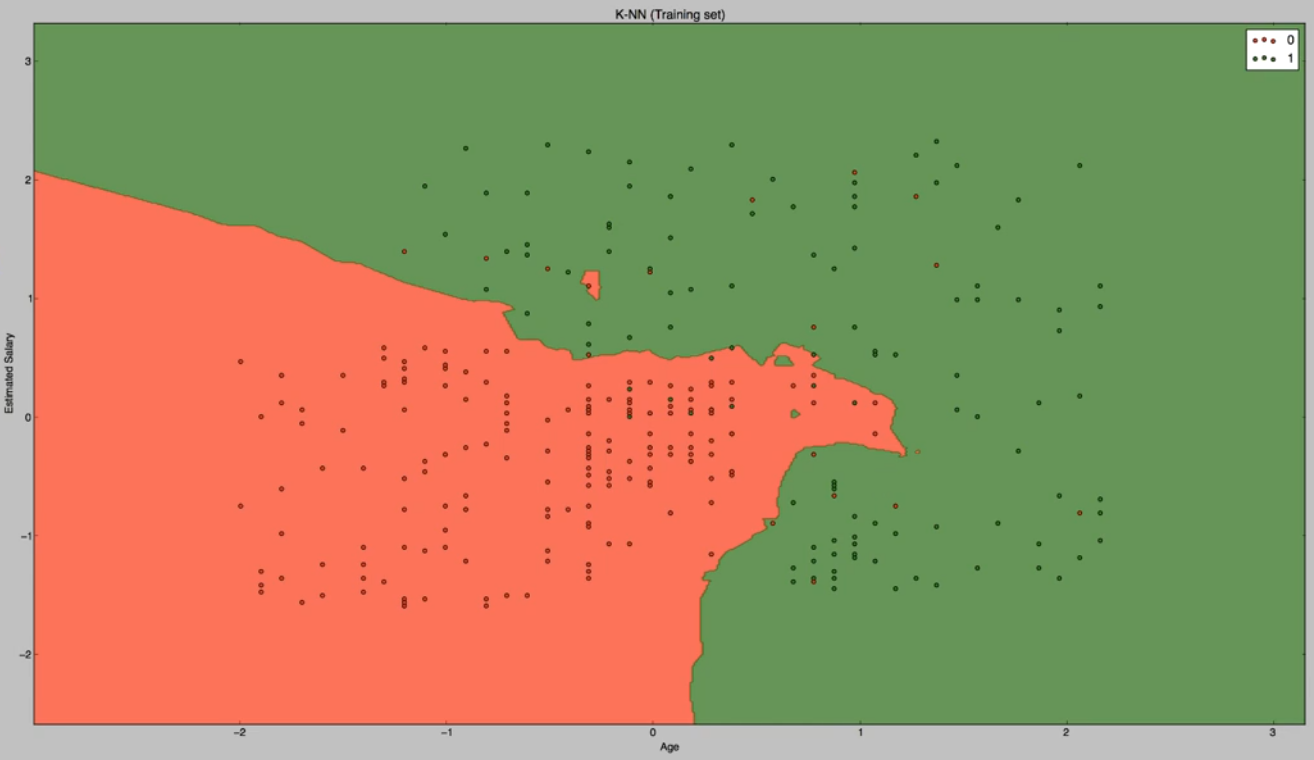
# MODEL PREDICTION BOUNDARIES

## LOGISTIC REGRESSION CLASSIFICATION

Since a linear regression is used, a straight line used to create the decision boundary. It’s not effected by outliers.

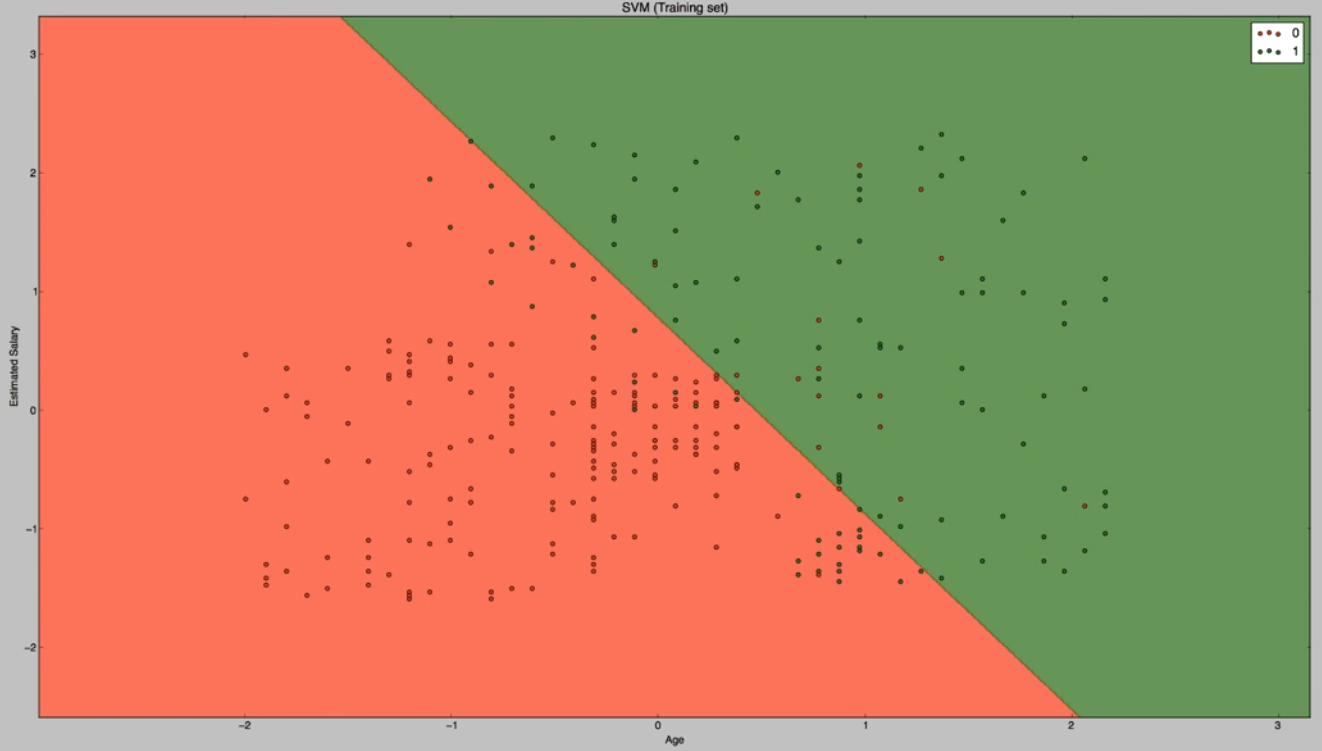


## K NEAREST NEIGHBORS (K-NN) CLASSIFICATION

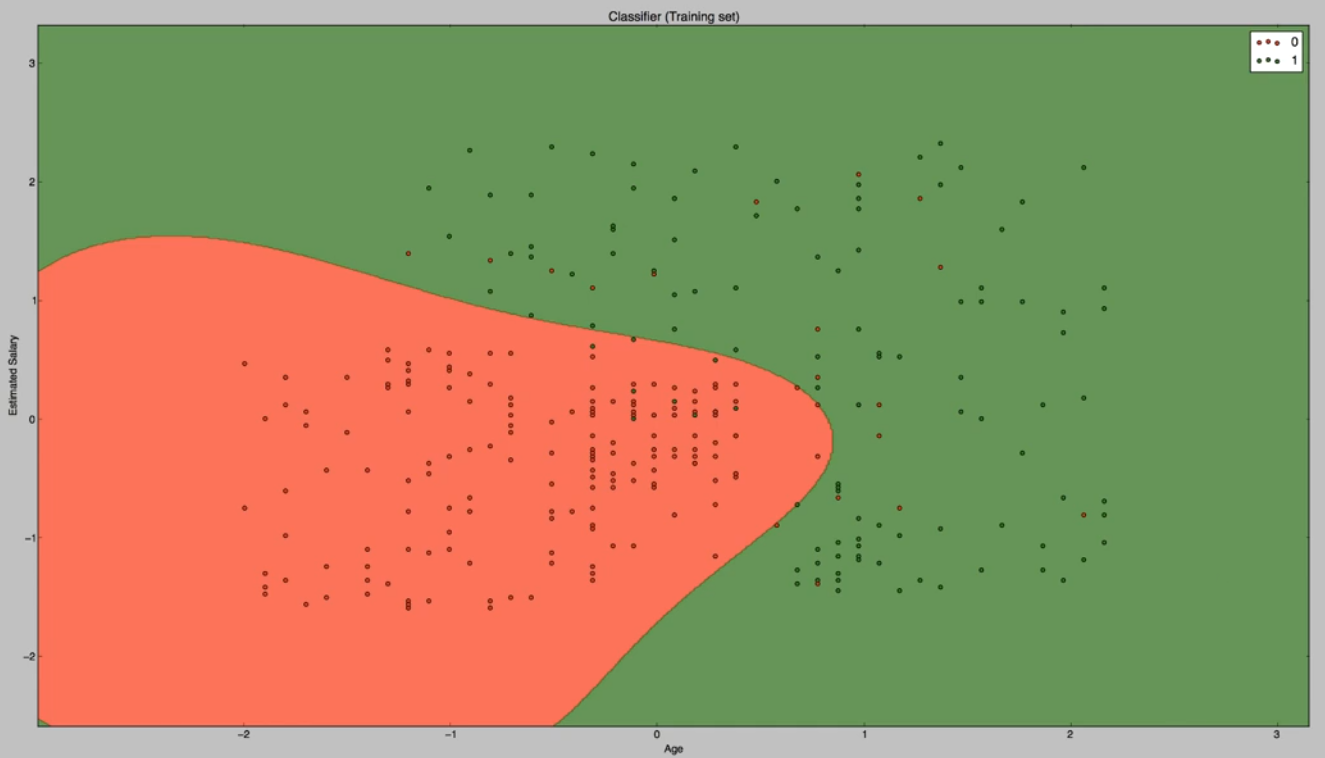


## SUPPORT VECTOR MACHINE CLASSIFICATION

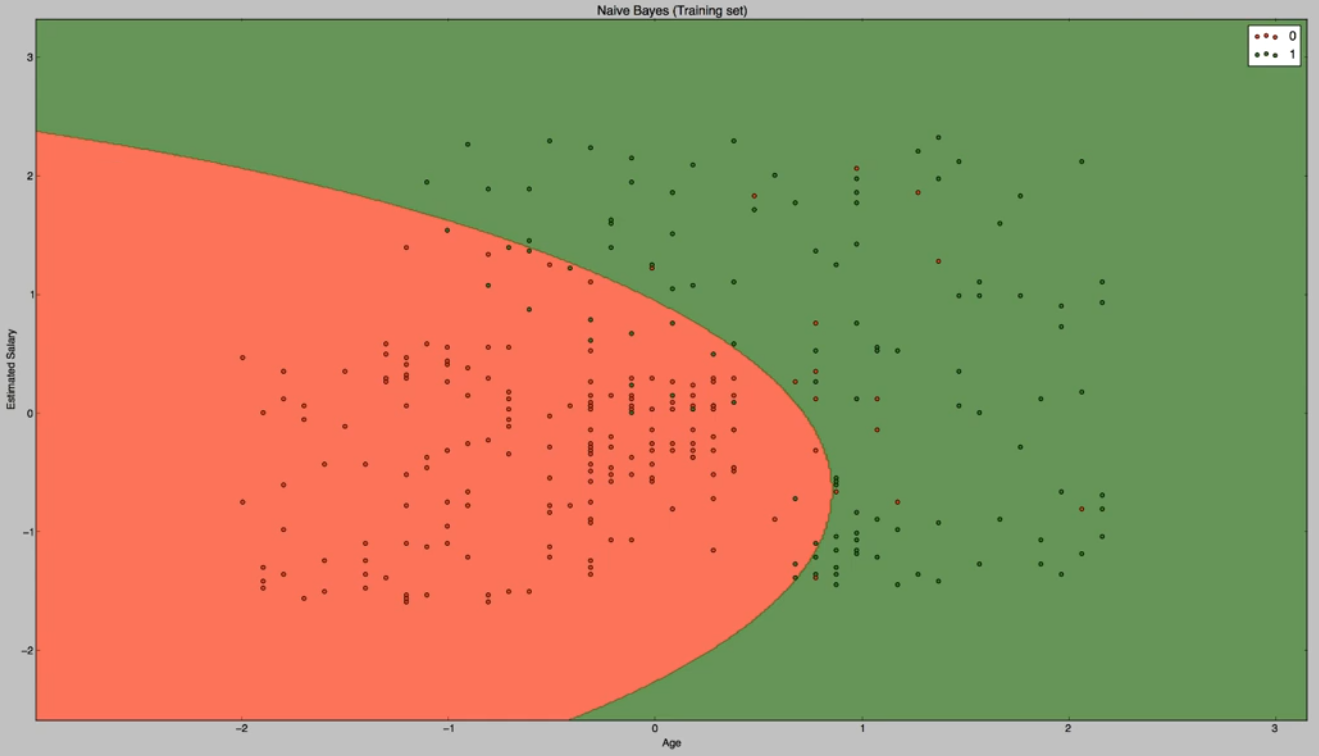
Since we have used kernel = ‘linear’ we are getting a straight line as decision boundary.



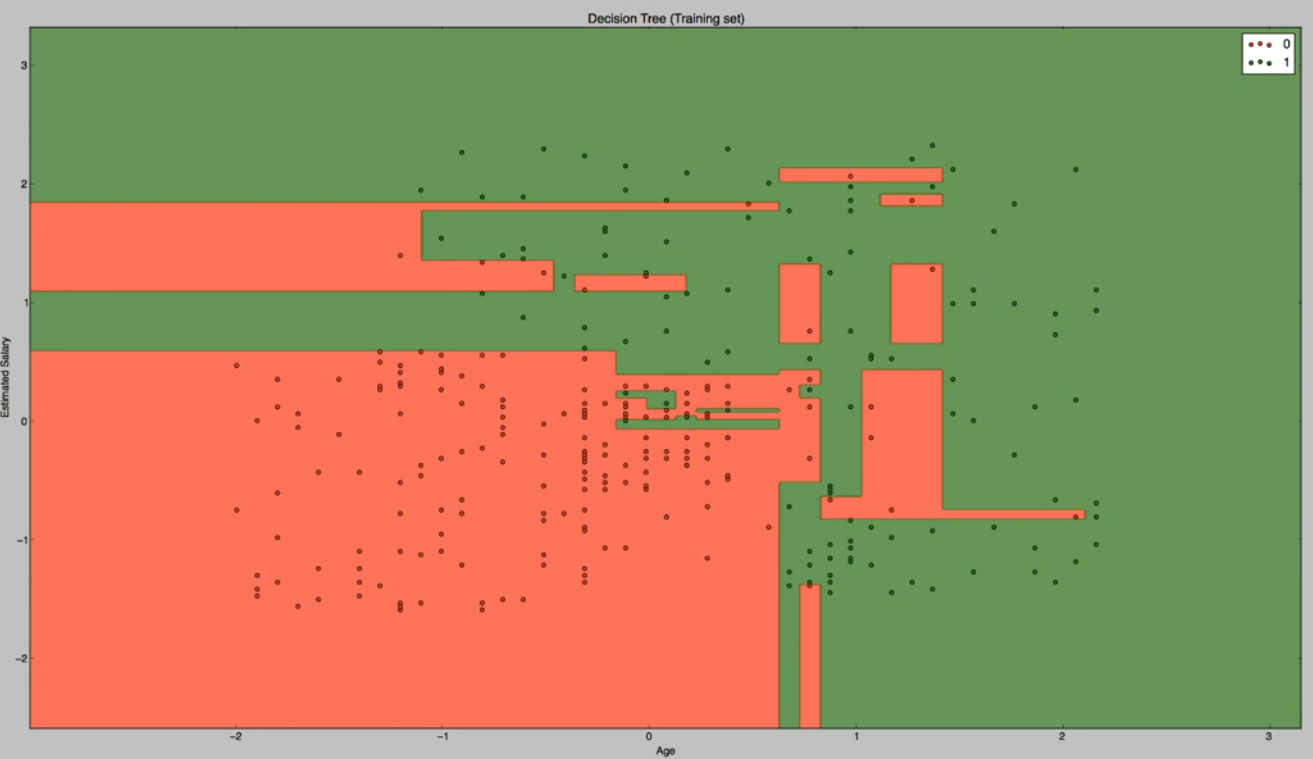
## SVM CLASSIFICATION WITH KERNEL = RBF



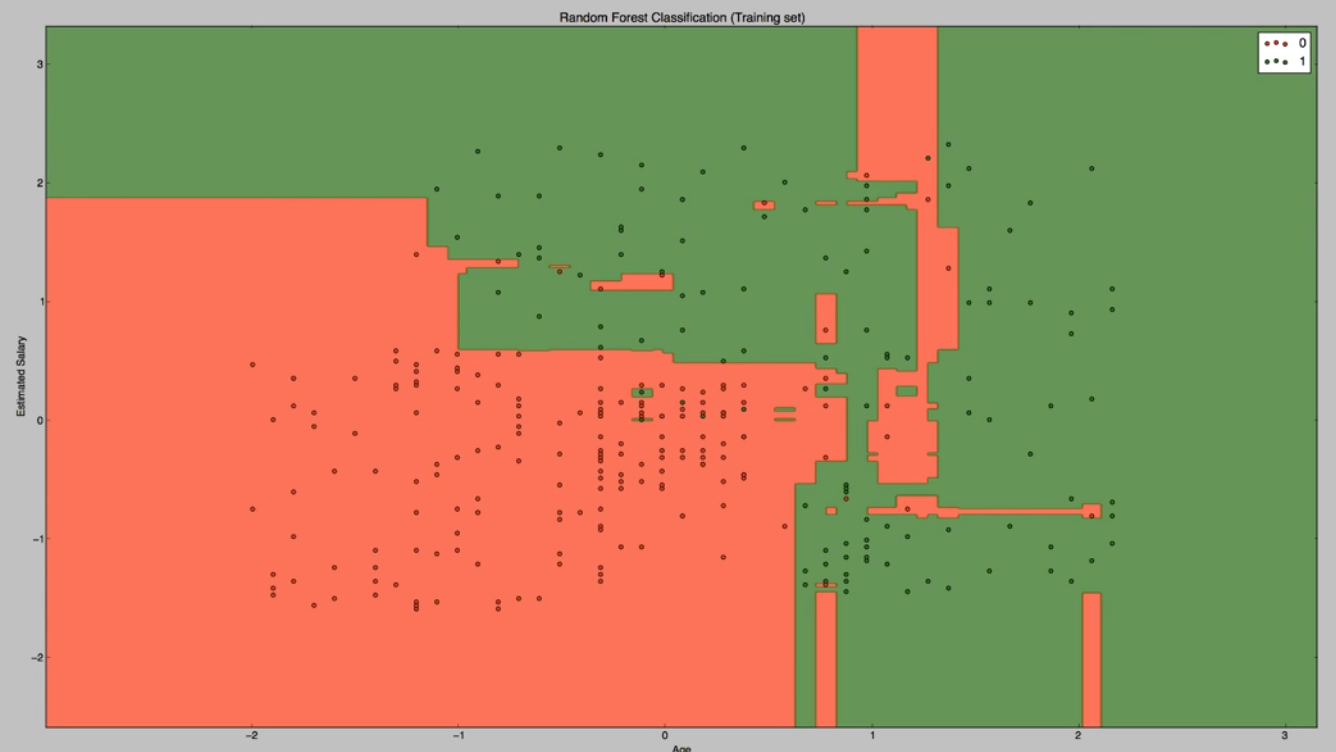
## NAÏVE BAYES CLASSIFICATION



## DECISION TREE CLASSIFICATION



## RANDOM FOREST



# REFERENCES

Cross validation example

<https://www.ritchieng.com/machine-learning-cross-validation/>

Model evaluation using CV

<https://scikit-learn.org/stable/modules/model_evaluation.html>

Multi Label Classification

<https://www.analyticsvidhya.com/blog/2017/08/introduction-to-multi-label-classification/>

Unbalanced class dataset

<https://www.analyticsvidhya.com/blog/2017/03/imbalanced-classification-problem/>

Class Imbalance Techniques

<https://elitedatascience.com/imbalanced-classes>

<https://www.kaggle.com/rafjaa/resampling-strategies-for-imbalanced-datasets>

<http://glitterpointcorp.com/a-solution-for-high-dimensional-imbalanced-dataset-challenges-in-ml-using-catboost/>