# **Chapter 4: Training Model**

Two ways to run Linear Regression:

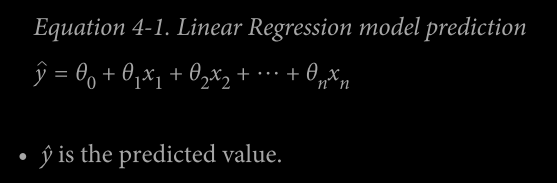
1) Using a direct "closed-form" equation that directly computes the model parameters that best fit the model to the training set - minimizing the cost function in one go.

2) Using an iterative optimization approach, called Gradient Descent (GD), gradually tweaks the model parameters to minimize the cost function over the training set, eventually converging the same set of parameters as the first method.

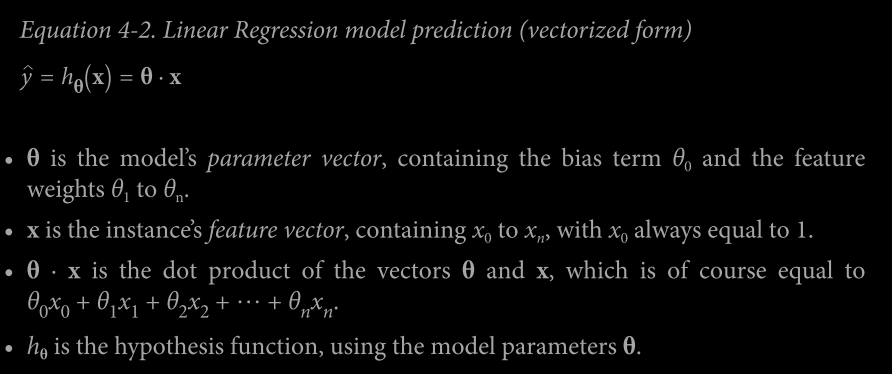
# Linear Regression

# @ <https://towardsdatascience.com/introduction-to-machine-learning-algorithms-linear-regression-14c4e325882a>

The linear model makes prediction by simply computing the weighted sum of the inputs plus a *bias* (aka *intercept term*).



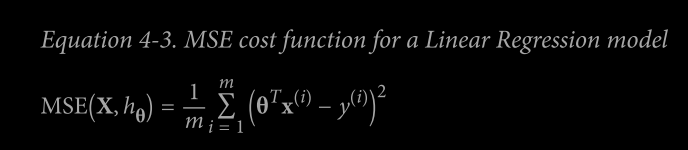
This can be written in vectorized form



## Training Linear Regression

Training a model means setting the parameters to describe best the data it is trained on while maintaining its generality. For this, we need a measure of how poorly the model is performing - RMSE (as learned in previous chapters).

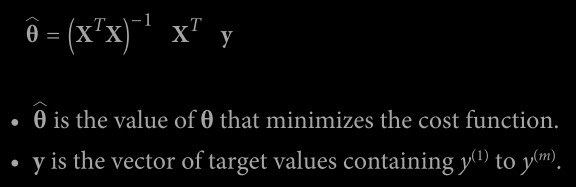
In practice, it is easy to minimize MSE rather than RMSE, and we get the same result.



## The Normal Equation

### @ <https://machinelearningmedium.com/2017/08/28/normal-equation/>

To find the value of theta that minimizes the cost function, there is a *closed-form solution* - in other words, a mathematical equation that gives the result directly. This is called the *Normal Equation*.



# **Feature Scaling is not necessary for the Normal equation method.**

The reason being, the feature scaling was implemented to prevent any skewness in the contour plot of the cost function, which affects the gradient descent, but the analytical solution using a normal equation does not suffer from the same drawback.

### **# Comparison between Gradient Descent and Normal Equation**

### 

### # Generally, the Normal Equation method is used only when number of features is less than 10k.

### # Normal Equation Method doesn’t work when the matrices are non-invertible, these matrices are known as singular or degenerate.

### Reason for non-invertibility are:

### 1) Linearly Dependent features ( i.e. Redundant features),

### 2) Too many features i.e. m<n, then the number of features or use regularization.

### # Calculating **Psuedo-inverse instead of inverse** can also solve the issue of non-invertibilty.

### # The **LinearRegression** class is based on the **scipy.linalg.lstsq() function** (the name stands for “least squares”). This function computes where X+ is the pseudoinverse of X (specifically the Moore Penrose inverse).

### 

### 

### **The Gradient Decent**

### **@** [**https://towardsdatascience.com/gradient-descent-algorithm-a-deep-dive-cf04e8115f21#:~:text=Gradient%20descent%20(GD)%20is%20an,e.g.%20in%20a%20linear%20regression**](https://towardsdatascience.com/gradient-descent-algorithm-a-deep-dive-cf04e8115f21#:~:text=Gradient%20descent%20(GD)%20is%20an,e.g.%20in%20a%20linear%20regression)**).**

### The general idea of gradient decent is to tweak parameters iteratively in order to minimize a cost function.

### 1) Starts with random values of theta ( known as Random Initialization )

### 2) Measures the local gradient of the error function

### 3) Takes small step at a time, each step attempting to decrease the cost function, until the algorithm converges to a minimum.

### # **Learning Rate** Hyperparameter (i.e size of steps )

### If the Learning Rate is small , it takes long time to converge.

### 

### If the learning rate is too large, it diverges the algorithm.

### 

### 

### # MSE cost function for a Linear Regression model is a **convex function,** which means the line join any point on the curve will never cross the curve and this implies that there is no global minimum. It is also a continuous function with slope that never changes abruptly ( Lipschitz Continuous ) .

### # **Feature Scaling** :

### 1) When using Gradient Decent, all the features must have similar scales ( Scikit-Learn’s StandardScaler ) or else it will take a long time time to converge.

### 2) In the 2nd figure, features are not scaled due to which feature 1 will take longer time to converge as compared to feature 2.

### 

### **Partial Derivative of Cost Function**

### 

### 

### @ <https://towardsdatascience.com/batch-mini-batch-stochastic-gradient-descent-7a62ecba642a>

### @ <https://www.analyticsvidhya.com/blog/2021/03/variants-of-gradient-descent-algorithm/>

### **Batch Gradient Descent :** In this formula, full training set X is used at each Gradient Descent step and this becomes very slow on very large training data sets.

### where η is learning rate

### # η decides the size and direction (+ve/-ve) of the step, as it multiplied by the gradient vector.

### 

### **Stochastic Gradient Descent**

### This method picks the random instance in the training set at every step and computes the gradient descent. This algorithm is less regular than Batch Gradient Descent : instead of gently decreasing unit it reaches the minimum, the cost function will bounce up and down, decreasing only on average.

### 

### This method is comparatively faster and good at skipping the local minima. In order to prevent the skipping of global minima, it uses **Simulated Annealing algorithm,** in which the learning rate gradually decreases as it approaches the global minimum. The function that determines the learning rate at each iteration is called the **learning schedule.**

### By convention we iterate by rounds of m iterations; each round is called an epoch. When using Stochastic Gradient, the training instances must be independent and identically, to ensure that the parameters get pulled towards the global optimum, on average. In the order to prevent the same, we need to shuffle the instances during training (pick each instance randomly or shuffle the training set at the beginning of each epoch).

### **Mini-Batch Gradient Descent**

### In this method, at each step, Instead of computing the gradients based on the full training set) as in the batch GD) or based on just one instance (as in Stochastic GD), Mini-Batch GD computes the gradients on small random sets of instances called mini batches.

### 

### 

**Polynomial Regression**

In Polynomial Regression, the algorithm fits any polynomial function to a non-linear data. We add the power of each feature as new features, then train a linear model on the extended set of features.

### # PolynomialFeatures(degree=d) transforms an array containing n features into an array containing features.

### **Learning Curve:** These are the plots of model’s performance on the training set and validation set as a function of the training set size (or the training iteration). To generate the plots, simply train the model several times on different sized subsets of the training data.

### 

### 1) Learning curves of the plain Linear Regression model (a straight line):

### 

### 1.1 The curve starts at zero, as there are only one or two instances in the training set, the model can fit them perfectly. It shows high error in validation set and is incapable of generalizing properly.

### 1.2 As the training set size increases, straight line is not able to fit both training and validation set, and the becomes almost same which means adding more data points is not going to help.

### These learning curves are typical of an underfitting model. Both curves have reached a plateau; they are close and fairly high.

### 2) Learning Curves of 10th degree polynomial model on the same data:

### 

### 2.1 The error on the training set is lower than with the Linear regression.

### 2.2 There is gap between the two curve which indicates the overfitting of model ( error in training set is less than validation set ).

### **The Bias/Variance Tradeoff**

### **@** [**https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229**](https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229)

### **@** [**https://www.analyticsvidhya.com/blog/2020/08/bias-and-variance-tradeoff-machine-learning/**](https://www.analyticsvidhya.com/blog/2020/08/bias-and-variance-tradeoff-machine-learning/)

### **@** [**https://medium.com/@itbodhi/bias-and-variance-trade-off-542b57ac7ff4**](https://medium.com/@itbodhi/bias-and-variance-trade-off-542b57ac7ff4)

### An important theoretical result of statistics and Machine Learning is the fact that a model’s generalization error can be expressed as the sum of three very different errors:

### 1) Bias : This part of the generalization error is due to wrong assumptions, such as assuming that the data is linear when it is actually quadratic. A high-bias model is most likely to underfit the training data.10

### 2) Variance : This part is due to the model’s excessive sensitivity to small variations in the training data. A model with many degrees of freedom (such as a high-degree polynomial model) is likely to have high variance, and thus to overfit the training data.

### 3) Irreducible error : This part is due to the noisiness of the data itself. The only way to reduce this part of the error is to clean up the data (e.g., fix the data sources, such as broken sensors, or detect and remove outliers).

### Increasing a model’s complexity will typically increase its variance and reduce its bias. Conversely, reducing a model’s complexity increases its bias and reduces its variance. This is why it is called a tradeoff.

### **Regularized Linear Model**

### **@** [**https://analyticsindiamag.com/hands-on-tutorial-on-elasticnet-regression/**](https://analyticsindiamag.com/hands-on-tutorial-on-elasticnet-regression/)

### @ <https://www.analyticssteps.com/blogs/l2-and-l1-regularization-machine-learning?fbclid=IwAR1pkL-RKR0xkKkdXtC1Tjcqm7CRX-FTD64U-nwNZYm_qnP2HEhhUdx0wW8>

### It is method to reduce the overfitting the data by regularizing model (i.e to constraint it)

### **Ridge Regression**

### Ridge Regression (also called Tikhonov regularization) is a regularized version of Linear Regression: a regularization term equal alpha sum from i equals 1 to n of theta subscript i superscript 2 is added to cost function.

### # The hyperparameter α controls the amount of regularization needed.

### 1) If α is small, then Ridge Regression is Linear Regression

### 2) If α is large, then all weights( θi ) becomes zero and result is horizontal line passing through the mean.

### **#** Regularization term is added to the cost function only during training and once the model is trained, model’s performance is evaluated using the unregularized performance measure.

### 

### # Ridge Regression needs scaled data (e.g., Standard Scaler)

### # As the α increases, curve becomes flatter (i.e., less extreme, more reasonable) predictions, this reduces the model’s variance but increases its bias.

### 

### Where A is (n+1)\*(n+1) identity matrix except with 0 in top-left cell, corresponding to bias term.

### **Lasso Regression (Least Absolute Shrinkage and Selection Operator Regression)**

### 

### **#** Lasso Regression completely eliminates the weights of least impoertnant features (i.e., set them to zero). In other words, Lasso Regression automatically performs feature selection and outputs a sparse model (i.ie, with few non-zero feature weights).

### **#** The Lasso cost function is not differentiable at θi = 0 (for i=1,2,3,. . . . ,n) but Gradient Descent still works fine if you use a subgradient vector g instead when any θi

### 

### **Elastic Net**

### The Elastic Net is a simple mix of both Ridge and Lasso’s Regularization .

### 

### 1) r=0 ; Elastic Net = Ridge Regression

### 2) r=1 ; Elastic Net = Lasso Regression

### # It is always good to have some regularization (instead of just using linear regularization).

### # Use Lasso or Elastic Regularization when only few features are useful (it tends to reduce useless features and weights down to zero)

### # Elastic Net is preferred when the no. of features > no. of training instance OR when the several features are strongly correlated.

### **Early Stopping**

### The method to regularize iterative learning algorithms such as Gradient Descent by stop training as soon as the validation error reaches a minimum is called Early Stopping.

### #Figure shows graph plotted between RSME vs Epoch for high-degree Polynomial Regression model being trained using the Batch Gradient Descent.

### The both validation error and training error decreases as Epochs increases. After a while validation error stops decreasing and actually starts to go back up and this indicates that the model has started to overfit the training data.

### Early Stopping stops training as soon as the validation error reaches the minimum.

### 

**Questions/Answers:**

**@** [**https://www.analyticsvidhya.com/blog/2021/06/25-questions-to-test-your-skills-on-linear-regression-algorithm/**](https://www.analyticsvidhya.com/blog/2021/06/25-questions-to-test-your-skills-on-linear-regression-algorithm/)

**@** [**https://www.analyticsvidhya.com/blog/2017/07/30-questions-to-test-a-data-scientist-on-linear-regression/**](https://www.analyticsvidhya.com/blog/2017/07/30-questions-to-test-a-data-scientist-on-linear-regression/)

**@** [**https://blog.uwgb.edu/bansalg/statistics-data-analytics/linear-regression/**](https://blog.uwgb.edu/bansalg/statistics-data-analytics/linear-regression/)

**@** [**https://ashutoshtripathi.com/2021/09/13/frequently-asked-machine-learning-interview-questions-from-linear-regression/**](https://ashutoshtripathi.com/2021/09/13/frequently-asked-machine-learning-interview-questions-from-linear-regression/)

**@** [**https://medium.com/analytics-vidhya/20-interview-questions-on-linear-regression-and-logistic-regression-ef4d341d2805**](https://medium.com/analytics-vidhya/20-interview-questions-on-linear-regression-and-logistic-regression-ef4d341d2805)

**@** [**https://kawsar34.medium.com/machine-learning-quiz-02-ridge-lasso-72d92900ba1e**](https://kawsar34.medium.com/machine-learning-quiz-02-ridge-lasso-72d92900ba1e)

### **Logistic Regression**

### **@** [**https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc**](https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc)

### **@** [**https://www.analyticsvidhya.com/blog/2021/08/conceptual-understanding-of-logistic-regression-for-data-science-beginners/**](https://www.analyticsvidhya.com/blog/2021/08/conceptual-understanding-of-logistic-regression-for-data-science-beginners/)

### Logistic Regression is used for classification by estimating the probability of a particular instance belongs to that class.

### 

### 



Where p = hθ (x) is probability that instance x belongs to the positive class, make its predictions y.

# Notice that σ(t) < 0.5 when t < 0,

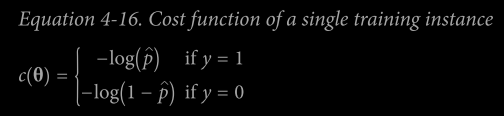
and σ(t) ≥ 0.5 when t ≥ 0,

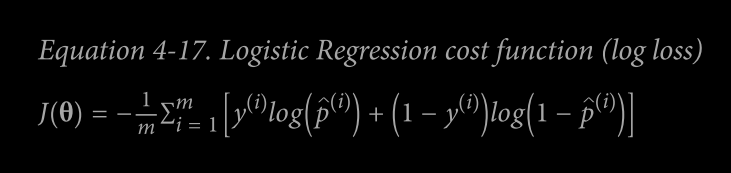
so a Logistic Regression model predicts 1 if xTθ is positive, and 0 if it is negative.

# **Logit** (**t**) is inverse of Logistic function

logit(p) =

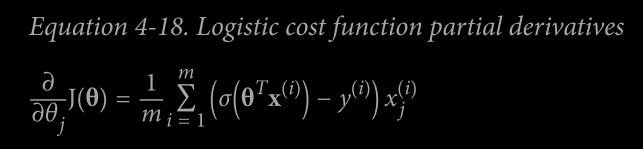
It is also called **log-odds** , as it is the log of the ratio between the estimates probability for positive and negative class.





# There is no closed-form equation to compute the value of θ that minimizes this cost function

# Logistic Regression’s cost function is convex, any optimization algorithm (like Gradient Descent) guaranteed to find the global minimum.



# Decision Boundary: it is an imaginary curve at which the both probability of prediction 1 and of 0 is same (i.e., 50%).

The model predicts 1 if probability lies above is curve and 0 if it lies below the curve.

**SoftMax Regression**

**@** [**https://towardsdatascience.com/multiclass-classification-with-softmax-regression-explained-ea320518ea5d**](https://towardsdatascience.com/multiclass-classification-with-softmax-regression-explained-ea320518ea5d)

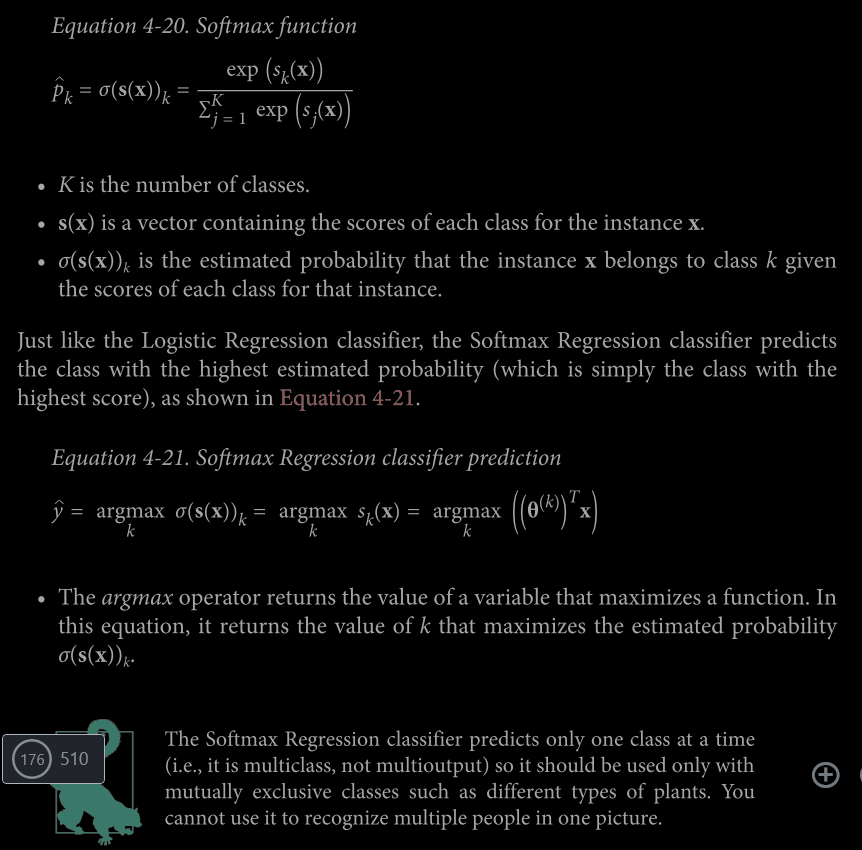
It is a generalized Logistic Regression model which can support multiple classes directly, without having to train and combine multiple binary classifier.

Step 1: Computes a score **sk(x)** for each class k.



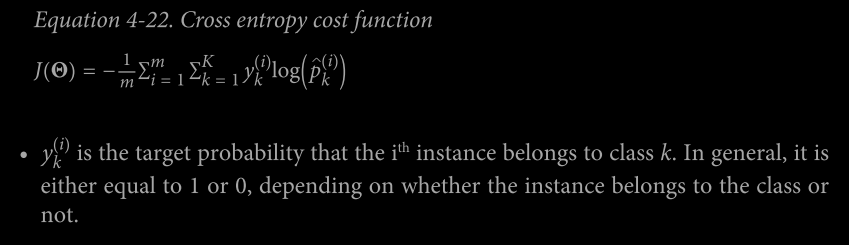
Each class has its own parameter vector ( **θ(k)** ) and all these vectors are stored as row in **parameter matrix**.

Step 2: Estimate the probability of each class by applying the **softmax function** (also called **normalized exponential**) to the score.



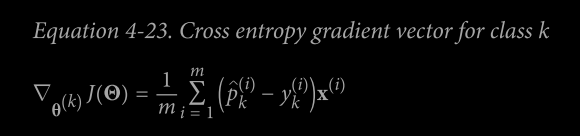
# **Cross Entropy** is frequently used to measure how well a set of estimates class probabilities match the target classes.

@ <https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e>



# For Two Classes (k=2), this cost function is equivalent to logistic regression’s cost function( log loss ).

# The cross entropy between two probability distributions p and q is defined as (at least when the distributions are discrete).



# Compute the gradient vector using above equation for every class, then use Gradient descent ( or any other optimization algorithm) to find parameter matrix Θ that minimizes the cost function.

Question And Answers:

### **@** [**https://www.analyticsvidhya.com/blog/2021/05/20-questions-to-test-your-skills-on-logistic-regression/**](https://www.analyticsvidhya.com/blog/2021/05/20-questions-to-test-your-skills-on-logistic-regression/)

### **@** [**https://www.analyticsvidhya.com/blog/2017/08/skilltest-logistic-regression/**](https://www.analyticsvidhya.com/blog/2017/08/skilltest-logistic-regression/)

### **@** [**https://medium.com/analytics-vidhya/interview-questions-on-logistic-regression-1ebd1666bbbd**](https://medium.com/analytics-vidhya/interview-questions-on-logistic-regression-1ebd1666bbbd)

**Chapter 5: Support Vector Machines**

**@** [**https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/#h2\_9**](https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/#h2_9)

**@** [**https://www.analyticsvidhya.com/blog/2021/06/support-vector-machine-better-understanding/**](https://www.analyticsvidhya.com/blog/2021/06/support-vector-machine-better-understanding/)

**@** [**https://towardsdatascience.com/hyperparameter-tuning-for-support-vector-machines-c-and-gamma-parameters-6a5097416167**](https://towardsdatascience.com/hyperparameter-tuning-for-support-vector-machines-c-and-gamma-parameters-6a5097416167)

**@** [**https://towardsdatascience.com/svm-hyperparameters-explained-with-visualizations-143e48cb701b**](https://towardsdatascience.com/svm-hyperparameters-explained-with-visualizations-143e48cb701b)

**Kernels:**

**@** [**https://medium.com/geekculture/kernel-methods-in-support-vector-machines-bb9409342c49**](https://medium.com/geekculture/kernel-methods-in-support-vector-machines-bb9409342c49)

**@** [**https://towardsdatascience.com/svm-and-kernel-svm-fed02bef1200**](https://towardsdatascience.com/svm-and-kernel-svm-fed02bef1200)

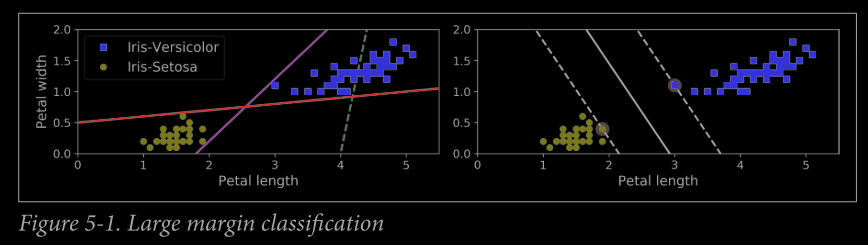
**@** [**https://www.slideshare.net/ankitksharma/svm-37753690**](https://www.slideshare.net/ankitksharma/svm-37753690)

# SVM is very Powerful and versatile ML model capable of performing linear or non-linear classification, regression and even outlier detection.

# It is well suited for classification of complex but small or medium-sized datasets.

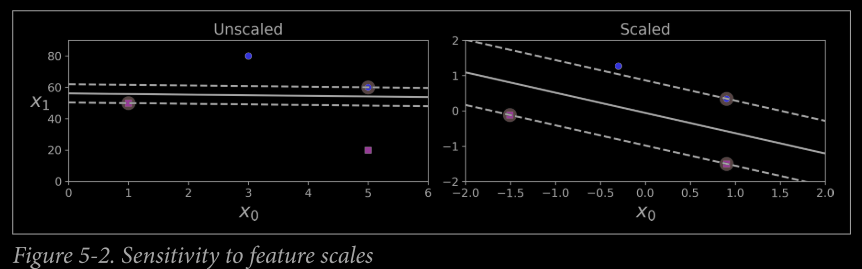
**Linear SVM Classification :**  In this classification, model separates the two classes with straight line (they are **linearly separable**).

# When the SVM classifier fits a line with widest possible street between the classes, then it is called **large margin classification**.



# The instances which are located at the end of edge fully determines (or supports) the decision boundary, these instances are called support vectors. Adding more ‘off the street’ training instances doesn’t affect the boundary.

# SVMs are sensitive to the features scales and after scaling (e.g., using Scikit-Learn’s StandardScaler), the decision boundary looks better (on the right plot).

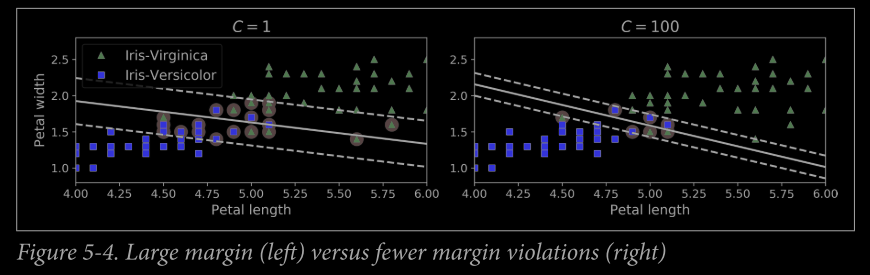


**Hard Margin Classification :** In this we strict keep all the instance to off the street and on the right side.

Two main issues of hard margin classification : 1) Data must be linearly separable 2) It is quite sensitive to outliers.

**Soft Margin Classification :** Soft Margin Classification is more flexible model. It keep the street as large as possible and limits the margin violations (i.e., instances that end up in the middle of the street or even on the wrong side).

# C hyperparameter : It controls the balance and both Width of street and Margin violations is **inversely proportion** to C value. **If your SVM model is overfitting, you can try regularizing it by reducing C.**



# Unlike Logistic Regression, SVM Classifier do not output the probabilities for each class.

# **LinearSVC Class**

1) It regularizes the bias term so the training set must be scaled.

2) Make sure loss hyperparameter to ‘hinge’.

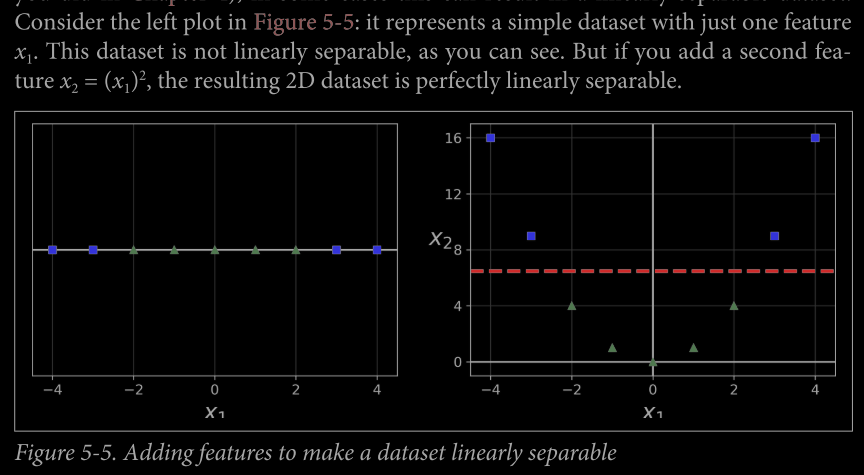
3) For better performance you should set the dual hyperparameter to False, unless there are more features than training instances.

# **SVC class** is generally not recommended as it is too slow, especially with large trainings sets.

# **SGDClassifier class** applies regular Stochastic Gradient Descent to train a linear SVM classifier. It does not converges as fast as SVM class, but it is useful to handle huge data sets or online classification tasks.

**Nonlinear SVM Classification**

When the data set is not linearly separable, we add more features, such as Polynomial Features.



# Disadvantage of adding features : At a low polynomial degree it cannot deal with very complex datasets, and with high polynomial degree it creates a huge no. of features, making the model slow.

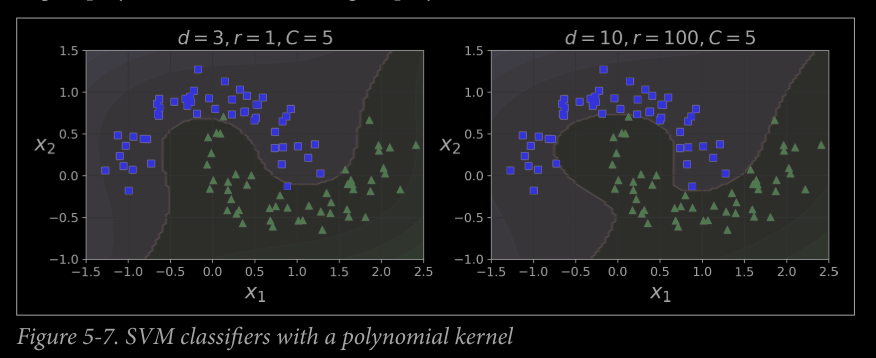
**Polynomial Kernel**

It uses the **Kernel Trick** to perform the Non-linear SVM classification, instead of adding more polynomial features.

SVM Classifier Code : ("svm\_clf", SVC(kernel="poly", degree=3, coef0=1, C=5))

# The hyperparameter coef0 controls how much the model is influenced by highdegree polynomials versus low-degree polynomials.

# If model is overfitting, you might want to reduce the polynomial degree. Conversely, if it is underfitting, you can try increasing it.



**# Finding Right Hyperparameter:** Use Grid Search; It is often faster to first do a very coarse grid search, then a finer grid search around the best values found.

**Adding Similarity Features**

This method is also used for Non-linear SVM Classification. It adds features computed using **similarity function** that measures how much each instance resembles a particular **landmark**.

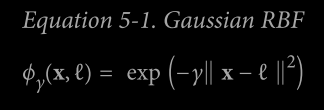
# One of the Similarity Function is **Gaussian Radial Basis Function (RBF).**

It is a bell-shaped function varying from 0 (very far away from the landmark) to 1 (at the landmark) and this convert the datasets into linearly separable data.

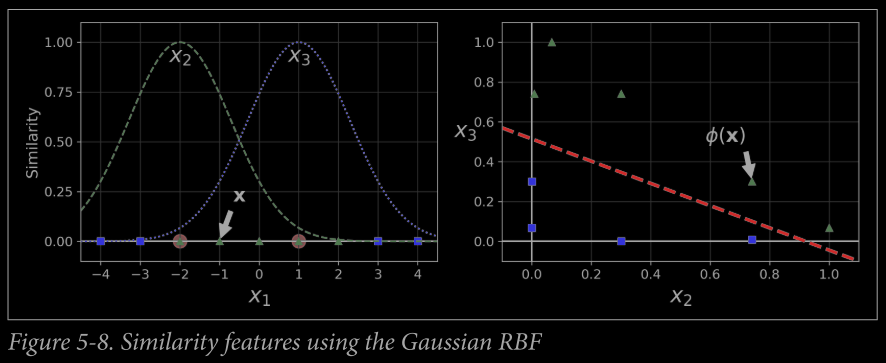
# Selecting Landmarks: Create a landmark at the location of each and every instance in the dataset, this creates many dimensions and linearly separability of transform training data.

# Training set with m instances and n features gets transformed into a training set with m instances and m features.

# If your training set is very large, you end up with an equally large number of features.

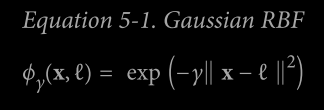


**Code :** ("svm\_clf", SVC(kernel="**poly**", degree=3, coef0=1, C=5))



# Polynomial Feature and Similarity Function Method is computationally expensive as it adds additional features especially on large training sets.

**Gaussian RBF Kernel**



**Code:** ("svm\_clf", SVC(kernel="**rbf**", gamma=5, C=0.001))

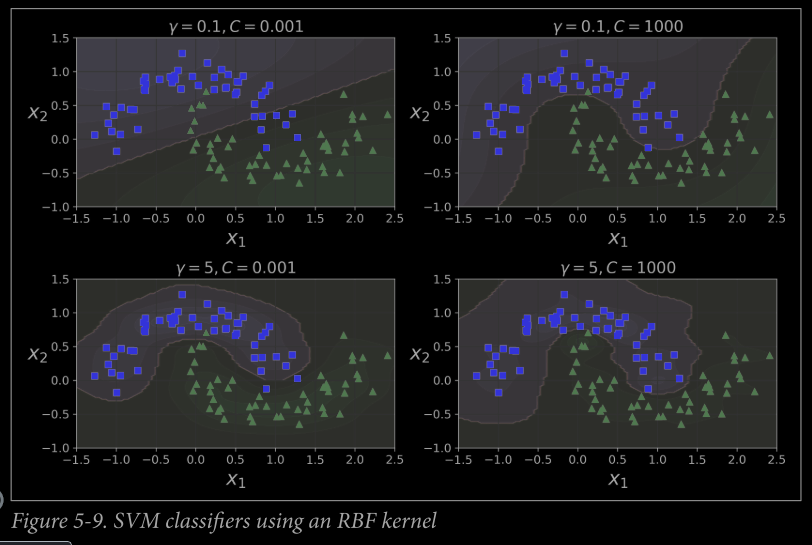
# As γ increases, curve becomes more narrow and as a result each instance’s influence is smaller and decision boundary becomes irregular.

# For small γ, bell-shaped curve are wider, so the instance’s influence increases and decision boundary becomes smoother.

# γ acts as Regularization Hyperparameter:

1) If model is overfitting, then reduce γ

2) If model is underfitting, then increase γ



**# Choice of Kernel**

1) First try Linear Kernel ( LinearSVC is much faster SVC(kernel=’linear ), especially if training set is very large.

2) If training is not too large, then try Gaussian RBF kernel

3) Try few other kernel using cross-validation and grid search, especially if there are kernels specialized for the training set’s data structure.

**Computational Complexity**

# LinearSVC class

1) Based on liblinear library.

2) It implements optimized algorithm for linear SVMs.

3) It doesn’t support the kernel trick, but it scales almost linearly with the no. of training instances and no. of features.

4) Training time complexity is roughly O(m x n).

5) It is controlled by the tolerance hyperparameter ϵ (called tol in Scikit-Learn).

# SVC class

1) Based on the libsvm library.

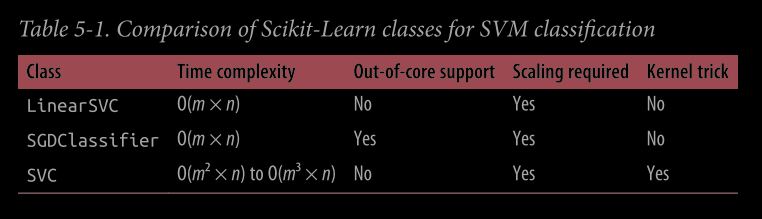
2) Implements an algorithm that supports kernel trick.

3) Training time complexity is usually between O(m2 × n) and O(m3 × n).

4) When training set is large, it becomes very slow

5) This algorithm is perfect for complex but small or medium training sets.

6) It scales well with the number of features, especially with sparse features (i.e., when each instance has few nonzero features).

7) The algorithm scales roughly with the average number of nonzero features per instance. 

**SVM Regression**

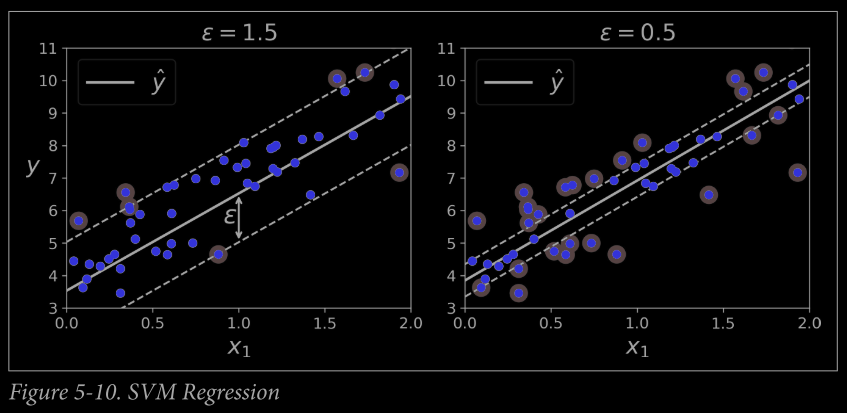
**@** [**https://towardsdatascience.com/unlocking-the-true-power-of-support-vector-regression-847fd123a4a0**](https://towardsdatascience.com/unlocking-the-true-power-of-support-vector-regression-847fd123a4a0)

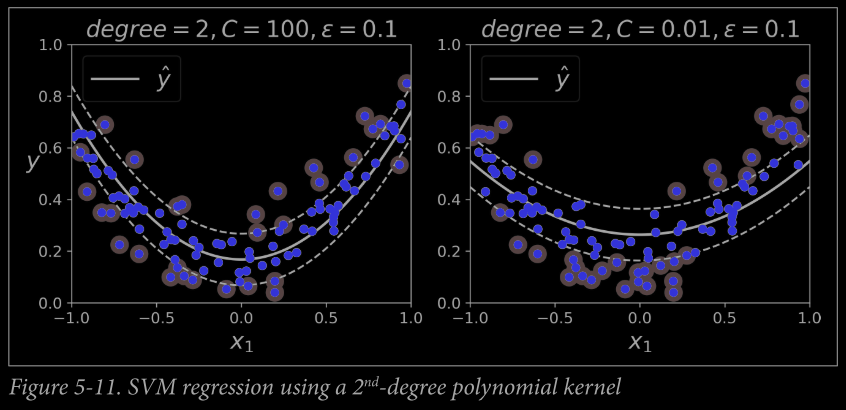
**@** [**https://www.analyticsvidhya.com/blog/2020/03/support-vector-regression-tutorial-for-machine-learning/**](https://www.analyticsvidhya.com/blog/2020/03/support-vector-regression-tutorial-for-machine-learning/)

SVM is versatile algorithm that supports both linear , non-linear classification as well as regression.

SVM Regression tries to fit as many instances as possible on the street while limiting margin violations (i.e., instances off the strret).

# The width of the street is controlled by hyperparameter ϵ. Adding more instances within the margin does not affect the model’s predictons and model is said to be ϵ-insensitive.





# Large C value means little regularization(left) and small C means more regularization.

# The SVR class is the regression equivalent of the SVC class, and the LinearSVR class is the regression equivalent of the LinearSVC class.

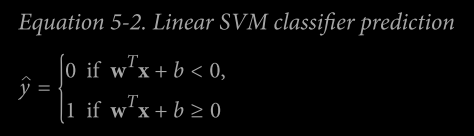
# The LinearSVR class scales linearly with the size of the training set (just like the LinearSVC class), while the SVR class gets much too slow when the training set grows large (just like the SVC class).

**“Under The Hood”**

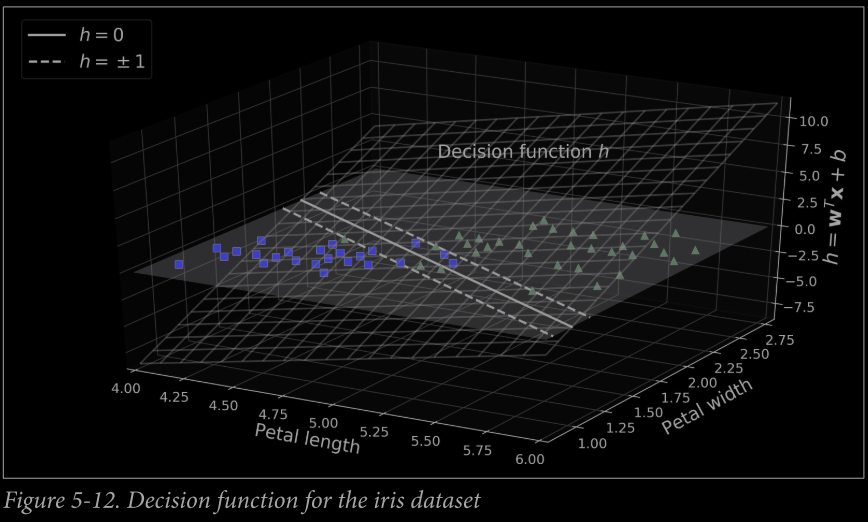
For studying SVMs, the bias term will be called b and the feature wieghts vector will be called w. No bias feature will be added to the input feature vectors.

**Decision Function and Predictions**

The **Prediction Clsss ŷ** is given by :



Where **Decision Function** is : wTx + b = w1x1 + w2x2 + w3x3 + . . . . . . . . . . . . . . . . . . . wnxn + b ;



# The **Decision Boundary**  is the set of point for which the decision function is eqaul to **0**. It is a straight line formed by the intersection of two planes as shown in the figure.

# The dashed lines represent the points where the decision function is equal to 1 to -1. They parallel and at equal distance to the decision boundary, forming the margin around it.

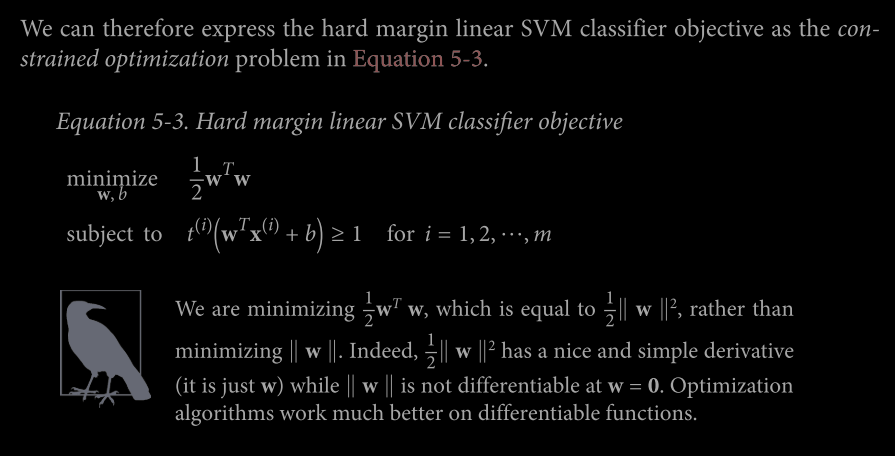
# Training a linear SVM classifier means finding the value of w and b that this margin as wide as possible while avoiding the margin voliations(hard margin) or limiting them (soft margin).

**Training Objective**

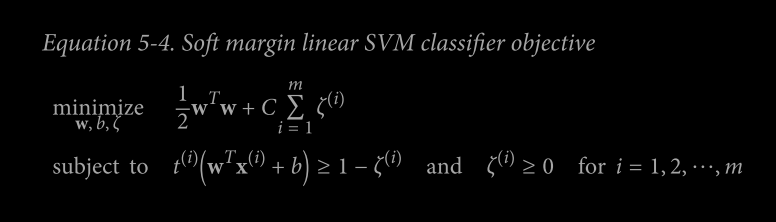
# Slope of decision function : ||w|| ( norm of vector w ).

# Margin is inversely proportional to weight vector w.

# In general, when the decision function is n-dimensional hyperplane then decision boundary is n-1 dimensional hyperplane.



where t(i) = -1 if y(i) = 0 (for negative training instance) and t(i) = 1 if y(i) = 1 (for positive training instance)



Slack Variable ( ζ(i) ): It measures how much the ith instance is allowed to violate the margin, and it is positive or zero for each instance.

C hyperparameter allows us to define the trade off between these two objectives.

Some Part of Chap5 SVM is left.

**# Logistic Regression vs. SVMs**

**@** [**https://medium.com/axum-labs/logistic-regression-vs-support-vector-machines-svm-c335610a3d16#:~:text=Difference%20between%20SVM%20and%20Logistic%20Regression&text=SVM%20works%20well%20with%20unstructured,is%20based%20on%20statistical%20approaches**](https://medium.com/axum-labs/logistic-regression-vs-support-vector-machines-svm-c335610a3d16#:~:text=Difference%20between%20SVM%20and%20Logistic%20Regression&text=SVM%20works%20well%20with%20unstructured,is%20based%20on%20statistical%20approaches)**.**

n = number of features (x € Rn+1) , m = number of training examples

1) If n is large relative to m ( e.g., n=10,000 , m = 10 – 1000 ) : Use logistic regression, or SVM without a kernel(‘linear kernel’)

2) If n is small, m is intermediate ( e.g., n = 1 – 1000 , m = 10 – 10,000 ) : Use SVM with Gaussian Kernel

3) If n is small, m is large (e.g., n=1 – 1000, m = 50,000+) : Create/Add more features, then use logistic regression or SVM without a kernel.

# Neural Networl is likely to work well in most of these settings, but maybe slower to train.

Questions and Answers:

@ <https://www.analyticsvidhya.com/blog/2021/05/top-15-questions-to-test-your-data-science-skills-on-svm/>

@ <https://towardsdatascience.com/support-vector-machine-svm-719e530a725f>

@ <https://medium.datadriveninvestor.com/support-vector-machines-important-questions-a47224692495>

@ <https://alekhyo.medium.com/interview-questions-on-svm-bf13e5fbcca8>

@ <https://kawsar34.medium.com/machine-learning-quiz-03-support-vector-machine-c40cc80279a5>

**Chapter 6 : Decision Trees**

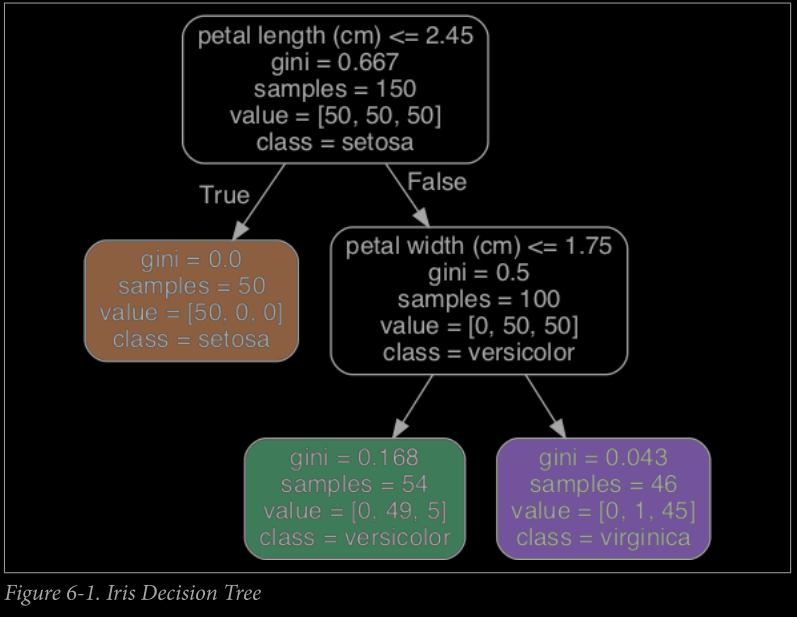
**@** [**https://www.analyticsvidhya.com/blog/2021/08/decision-tree-algorithm/#h2\_5**](https://www.analyticsvidhya.com/blog/2021/08/decision-tree-algorithm/#h2_5)

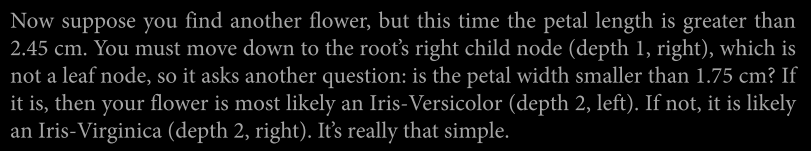
**@** [**https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052**](https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052)

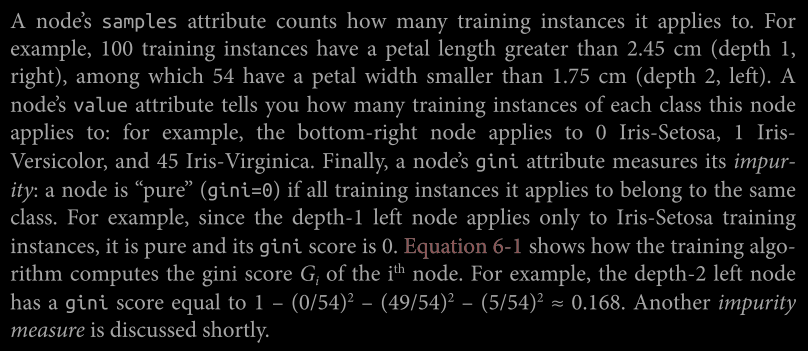
Decision Trees are versatile Machine Learning algorithms that can perform both classification and regression tasks, and even multioutput tasks. They are very powerful algorithms, capable of fitting complex datasets.

# Decision Trees require very little data preparation : No Need of Scaling or Centering at all.

# Data be continous or discrete, they works well for any kind of data.



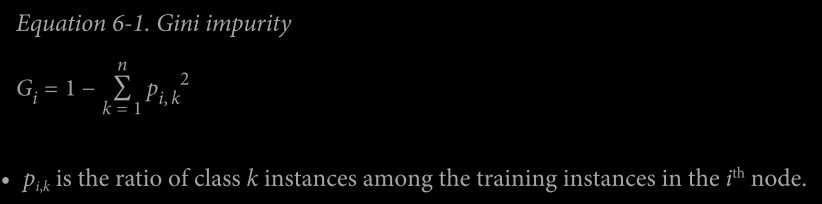




# The ‘**impurity’** of any node is measured by **gini** attribute (known as nodes’gini)

@ <https://towardsdatascience.com/gini-impurity-measure-dbd3878ead33>

# Node is ‘pure’ if gini=0



# Scikit-Learn uses the CART algorithm : It Produces only ‘binary trees’, nonleaf nodes always have two children (i.e., questions only have yes/no answers).

# Decision Trees can istimate the probabilty that an instance belongs to a particular class k.

**The CART Training Algorithm**

CART : Classification and Regression Tree

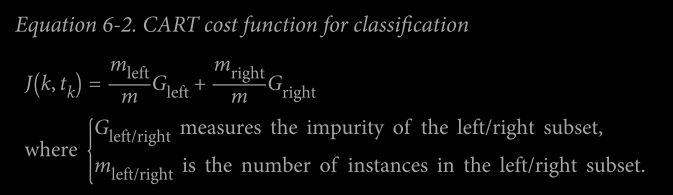
@ <https://towardsdatascience.com/cart-classification-and-regression-trees-for-clean-but-powerful-models-cc89e60b7a85>

Steps that algorithm performs:

1) Spilt the training set in two subsets using a single feature k and threshold tk.

2) It search for pair (k, tk,) that produces pureest subsets (wieghted by their size)

3) It minizes the cost function J(k,tk,) :



4) Once spliting the training set in two, it splits subsets using the same logic and then sub-subset and so on.

5) It stops recursing once it reaches the maximum depth.

**Computational Complexity**

The overall prediction complexity is O(log2(m)), independent of no. of features.

# Predictions are very fast, even with large training set.

# If max\_features is set then training complexity is O(n × m log(m)).

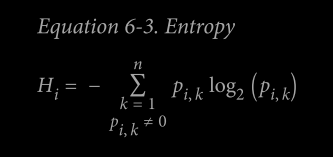
**Entropy**

**@** [**https://www.analyticsvidhya.com/blog/2020/11/entropy-a-key-concept-for-all-data-science-beginners/**](https://www.analyticsvidhya.com/blog/2020/11/entropy-a-key-concept-for-all-data-science-beginners/)

**@** [**https://towardsdatascience.com/entropy-and-information-gain-in-decision-trees-c7db67a3a293**](https://towardsdatascience.com/entropy-and-information-gain-in-decision-trees-c7db67a3a293)

# In ML, Entropy refers to the mixing of instances of different classes.

Entropy is zero when sample contains instances of one class only.



# Both Gini Impurity and Entropy produce almost same results and Gini is little faster.

# Gini Impurity tends to osolate the most frequant class in its own branch of tree, while entropy tends to produce slightly more balanced tress.

**Regularization Hyperparameters**

@ <https://towardsdatascience.com/hyperparameters-of-decision-trees-explained-with-visualizations-1a6ef2f67edf>

# Reducing ‘max\_depth’ hyperparameter will regularize the model and thus the risk overfitting

# Other parameters that reduce overfitting :

1) min\_samples\_split (the minimum number of samples a node must have before it can be split)

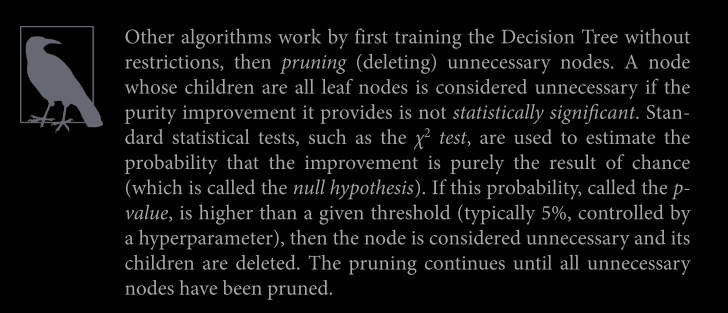
2) min\_samples\_leaf (the minimum number of samples a leaf node must have)

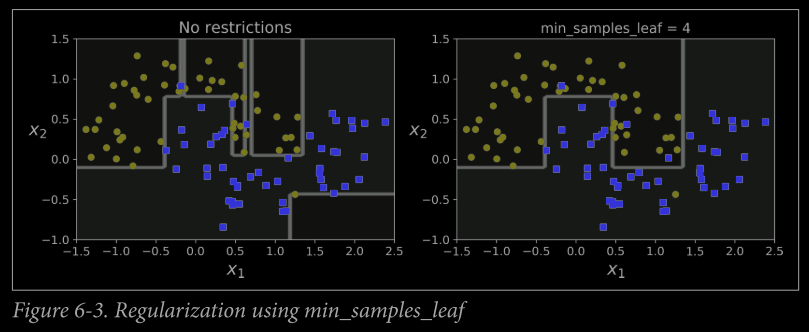
3) min\_weight\_fraction\_leaf (same as min\_samples\_leaf but expressed as a fraction of the total number of weighted instances),

4) max\_leaf\_nodes (maximum number of leaf nodes)

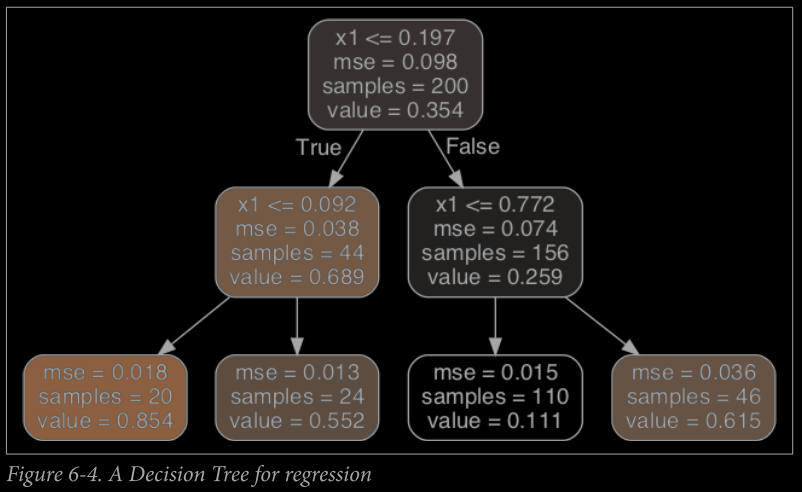
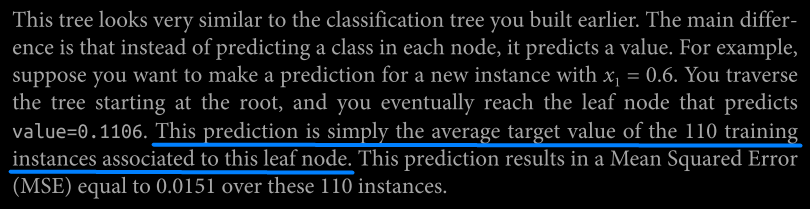
5) max\_features (maximum number of features that are evaluated for splitting at each node).

6) Increasing min\_\* hyperparameters or reducing max\_\* hyperparameters will regularize the model.





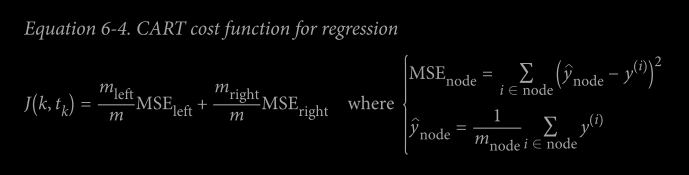
**Regression**

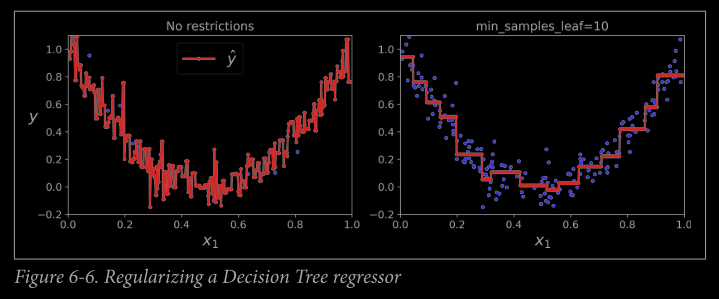
# This algorithm splits each region in a way that makes most training instances as close as possible to that predicted value.

# The CART algorithm works mostly in same way

# It tries minimize the MSE (instead of minimizing the impurity)



# Decision Trees for Regreesion are also prone to overfitting the training set and same hyperparameters available in DecisionTreesRegression class with which we can regularize the model.

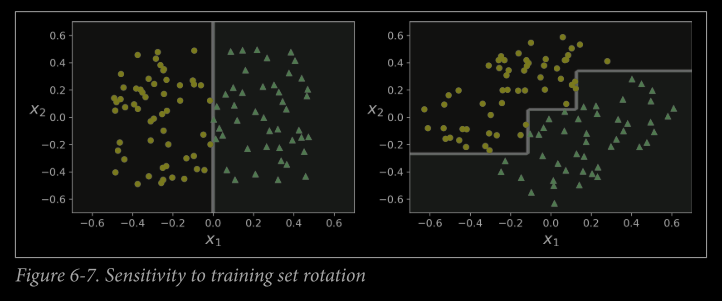


**Instability ( Limitations )**

# Sensitivity to training set rotation : On rotating the data set, it produces decision boundary which are unnecessarily convoluted.

In general, Decision Trees model always produces orthogonal decision boundaries.

# **To tackle this we use PCA**



# Decision Trees are very sensitive to small in any variations in training set.

# Scikit-Learn is stochastic, it randomly selects the set of features evaluate at each node and it produces very different model on the same training data (unless you set the random\_state hyperparameter).

Question and Answers:

@ <https://www.analyticsvidhya.com/blog/2021/05/25-questions-to-test-your-skills-on-decision-trees/>

@ <https://360digitmg.com/decision-tree-interview-questions-answers>

@ <https://kawsar34.medium.com/machine-learning-quiz-05-decision-tree-part-1-3ea71fa312e5>

@ <https://kawsar34.medium.com/machine-learning-quiz-06-decision-tree-part-2-1115393a2668>