BIT34503 Data Science

CHAPTER 7: SUPERVISED LEARNING

SUPERVISED LEARNING

7.1 Introduction to Prediction

7.2 Sampling, Training and Testing Set

7.1 Introduction to Prediction

- 2 prediction techniques for supervised learning techniques.
  - numerical prediction
  - category prediction
- These machine learning techniques are applied when the target whose value needs to be predicted is known in advance and some sample data is available to train a model.

Numerical Prediction

- **Overview**
  - ***Requirement.*** How can the value of a numerical variable be predicted based on a set of known values of other variables?
  - ***Problem.*** Certain machine learning problems require predicting a numerical value based on some known variables, making it impossible to apply machine learning techniques where the target can only be a categorical variable.
  - ***Solution.*** The relationship between the known variables and the target variable is captured using a regression algorithm, which is then used to predict the unknown value of the target variable based on known values of the other variables.
  - ***Application.*** Regression algorithms, such as linear regression, are applied to the data set by first developing a model based on the known set of values of both the target and other variables, and then making predictions for the unknown value of the target variable based on a known set of values of other variables.

Numerical Prediction

- **Problem**
  - A data set may contain a number of historical observations (rows) amassed over a period of time where the target value is numerical in nature and is known for those observations.
  - An example is the number of ice creams sold and the temperature readings, where the number of ice creams sold is the target variable.
  - To obtain value from this data, a business use case might require a prediction of how much ice cream will be sold if the temperature reading is known in advance from the weather forecast.
  - As the target is numerical in nature, supervised learning techniques that work with categorical targets cannot be applied (Figure 1).

Numerical Prediction

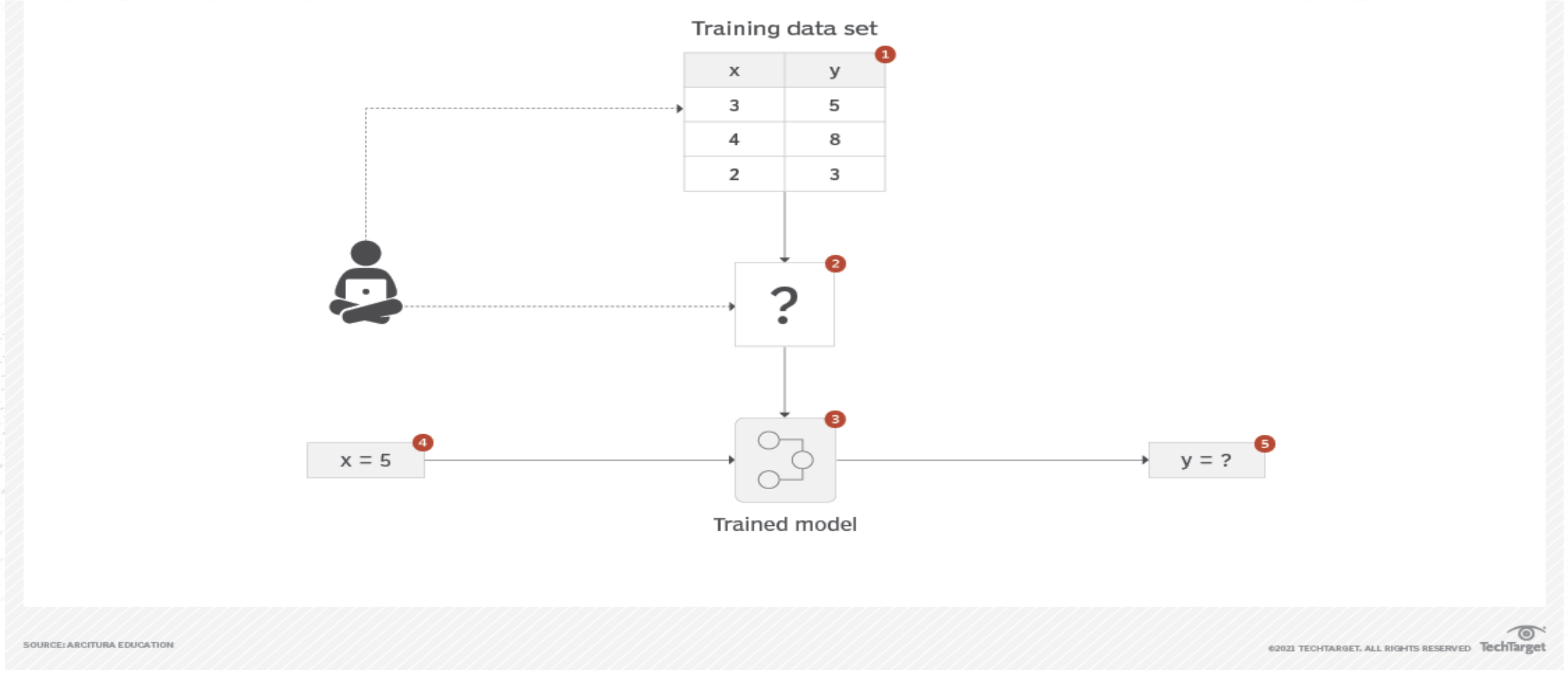


Figure 1. A training data set contains two variables (x and y) that are numerical in nature (1)

- An algorithm needs to be applied to train the model to predict numerical values (2, 3).
- Once the model is trained, a known value of x needs to be fed to the model in order to find the value of y (4, 5).

Numerical Prediction

- **Solution**
- The historical data is capitalized upon by first finding independent variables that influence the target dependent variable and then quantifying this influence in a mathematical equation.
- Once the mathematical equation is complete, the value of the target variable is predicted by inputting the values of the independent values.

Numerical Prediction

- **Application**
- The data set is first scanned to find the best independent variables by applying the associativity computation pattern to find the relationship between the independent variables and the dependent variable.
- Only the independent variables that are highly correlated with the dependent variable are kept. Next, linear regression is applied.

Numerical Prediction

- Linear regression, also known as *least squares regression*, is a statistical technique for predicting the values of a continuous dependent variable based on the values of an independent variable.
- The dependent and independent variables are also known as *response* and *explanatory* variables, respectively.
- As a mathematical relationship between the response variable and the explanatory variables, linear regression assumes that a linear correlation exists between the response and explanatory variables.
- A linear correlation between response and explanatory variables is represented through the line of best fit, also called a *regression line*.
- This is a straight line that passes as closely as possible through all points on the scatter plot (Figure 2).

Numerical Prediction

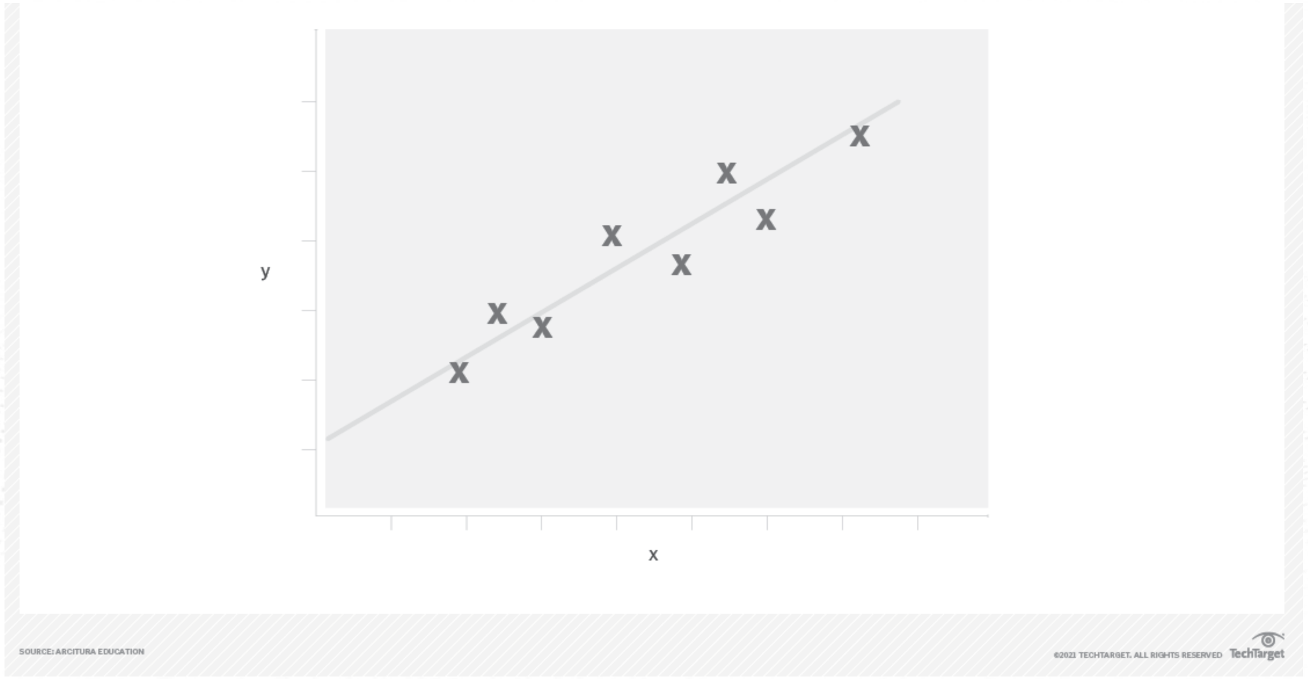


Figure 2. This is an example of linear regression as illustrated by the straight line passing as closely as possible to the scatter plot points.

Category Prediction

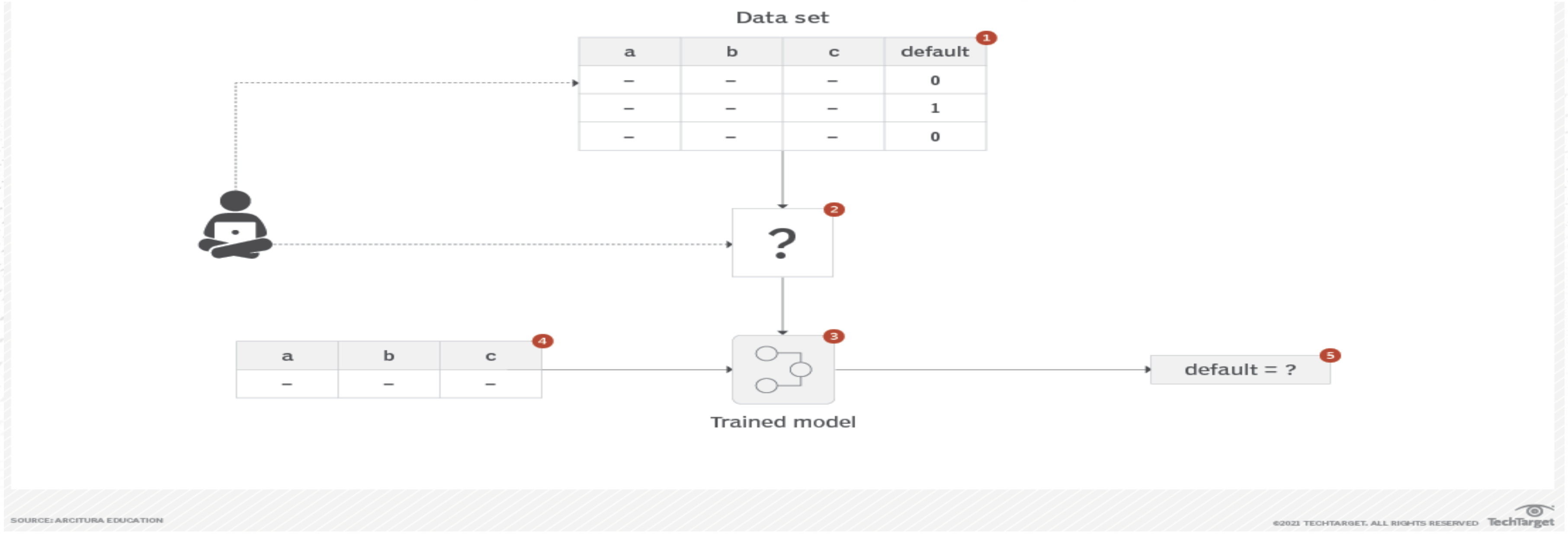
Overview

- ***Requirement.*** How can the category that new data belongs to be predicted based on previous example data given so that the target categories are known in advance?
- ***Problem.*** In certain scenarios, the possible categories to which a data instance belongs to are known in advance along with examples of data instances belonging to these categories. However, what is *not* known is how to categorize new data instance into one of the known categories.
- ***Solution.*** A classification model is trained by using the example data in order to build an understanding of the characteristics of examples that belong to different categories. The model is then used to classify new instances into one of the known categories.
- ***Application.*** Classification algorithms such as KNN, Naive Bayes, and Decision Trees are employed to build the classification model using existing example data and to predict the category of the new data instance.

Category Prediction

- **Problem**
- There are cases where a business problem involves predicting a category - such as whether a customer will default on their loan or whether an image is a cat or a dog - based on historical examples of defaulters and cats and dogs, respectively.
- In this case, the categories (default/not default and cat/dog) are known in advance.
- However, as the target class is categorical in nature, numerical predictive algorithms cannot be applied to train and predict a model for classification purposes (Figure 3).

Category Prediction



Data set
a
b
c
default
–
0
1
?
Trained model
default = ?
1
2
3
4
5
SOURCE: ARCITURA EDUCATION
©2021 TECHTARGET. ALL RIGHTS RESERVED
TechTarget

Figure 3. A financial data set

Category Prediction

- Figure 3 is a financial data set is prepared to display whether customers have defaulted on their loans based on a number of features such as age and outstanding loan amount (whereby 1=yes and 0=no) (1).
- A model needs to be trained for default prediction (2, 3).
- The trained model will be fed known feature values of a customer (not previously seen by the model), based on which the model needs to make a prediction on whether or not the customer will default (4, 5).

Category Prediction

- **Solution**
- Supervised machine learning techniques are applied by selecting a problem-specific machine learning algorithm and developing a classification model.
- This involves first using the known example data to train a model.
- The model is then fed new unseen data to find out the most appropriate category to which the new data instance belongs.

Category Prediction

- **Application**
- Different machine learning algorithms exist for developing classification models.
- For example, Naive Bayes is probabilistic while K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Logistic Regression and Decision Trees are deterministic in nature.
- Generally, in the case of a binary problem - cat or dog - logistic regression is applied.
- If the feature space is n-dimensional (a large number of features) with complex interactions between the features, KNN is applied.
- Naive Bayes is applied when there is not enough training data or fast predictions are required, while decision trees are a good choice when the model needs to be explainable.

7.2 Sampling, Training and Testing Set

- Splitting a data set into *train* and *test* sets is usually one of the first processing steps in a machine learning pipeline.
- After this point, there is just one inviolable rule: set aside the test data split and refer to it again only when your model is ready for its final evaluation.
- **But is there a way to be confident that the split we made is the right one?**
- **How can we be sure that the test set is representative of our population?**
- **We must be sure that the test set resembles closely the intricacies of our population.**
- **The same goes for our validation sets as well.**

Sampling

- We discuss two methods of splitting: random and stratified sampling.
- We consider their advantages and debate over which one to use under specific circumstances.

The Data Set

Random Sampling

- Creating a test set is really easy in theory; just select a few rows at random.
- Typically, the 10% or 20% of the full data set — and keep them in a new, separate dataframe.
- Moreover, for reproducibility, we could set a random_state number so every time we run an experiment, we get back the same test set.
- That would also help compare different approaches directly because every algorithm we choose is evaluated on the same test set.
- Moving forward, we use a convenient method such as train_test_split().
- The method accepts lists, numpy arrays, scipy sparse matrices or pandas dataframes.
- We should also provide the split percentage as it is None by default and a random_state number.

The Data Set

Random Sampling

- We can see that the method returns a tuple, which we can unpack directly into a train and a test set.
- Since we pass a dataframe to the method, the resulting train and test sets are also dataframes.
- Keep in mind that by default the method first shuffles the data set and then splits it.
- To avoid this behaviour we could set the shuffle attribute to False.
- Usually, if the data set is large enough (especially relative to the number of attributes) randomly splitting it would suffice.
- But if that’s not the case, random splitting could introduce significant sampling bias.

The Data Set

Stratified Sampling

- Imagine we are running a survey in Malaysia that records the education level and median income of a specific state.
- If we only question people that live in expensive districts, we would surely introduce significant bias in our sample.
- Thus, we need a way to select people of any economic status roughly approximating the corresponding distribution of the population.
- In this case, we can check if there is a variable that correlates highly with our target variable.
- This is done easily using the pandas.corr() method.

The Data Set

- We sort the results for the dependent variable in descending order and we get that from 2 variables median_income has a fairly strong positive correlation with median_house_value.
- This is surely something to expect.
- It seems that splitting the data set in a stratified fashion, using median_income as the class labels would be a good idea.
- We will examine the ratio for each income category in the full data set, as well as in the test sample.
- We can split into 5 classes for examples and see that there are very close in terms of its ratio.
- The small differences are really insignificant.
- We now have a test set that follows the distribution of the income category in our population!

REFERENCES

- https://www.techtarget.com/searchenterpriseai/post/2-supervised-learning-techniques-that-aid-value-predictions
- https://towardsdatascience.com/how-to-create-a-representative-test-set-f0aa56adaf35

SUPERVISED LEARNING

7.3 Classification Methods:

- 7.3.1 Neural Networks
- 7.3.2 Regression
- 7.3.3 K-Nearest Neighbors
- 7.3.4 Decision Trees
- 7.3.5 Support Vector Machines

7.3.1 Neural Networks

- Neural networks are complex models, which try to mimic the way the human brain develops classification rules.
- A neural net consists of many different layers of neurons, with each layer receiving inputs from previous layers, and passing outputs to further layers.
- The way each layer output becomes the input for the next layer depends on the weight given to that specific link, which depends on the cost function, and the optimizer.
- The neural net iterates for a predetermined number of iterations, called epochs.
- After each epoch, the cost function is analyzed to see where the model could be improved.
- The optimizing function then alters the internal mechanics of the network, such as the weights, and the biases, based on the information provided by the cost function, until the cost function is minimized.

7.3.1 Neural Networks

- *Figure 1* shows a visual representation of such a network.
- The initial input is *x,* which is then passed to the first layer of neurons (the h bubbles in *Figure 1),* where three functions consider the input that they receive, and generate an output.
- That output is then passed to the second layer (the g bubbles in *Figure 1)*.
- There further output is calculated, based on the output from the first layer. That secondary output is then combined to yield a final output of

7.3.1 Neural Networks

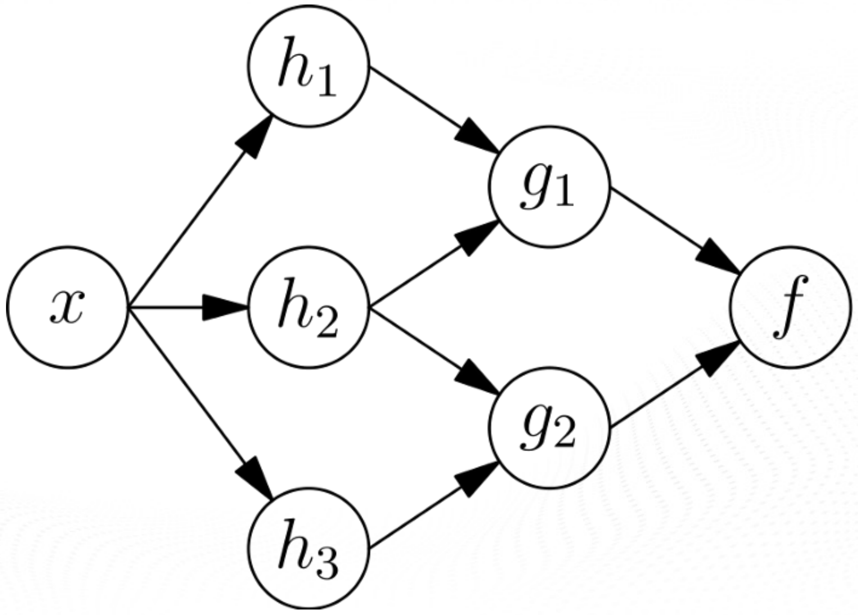

$$\begin{tikzcd}
 & h_1 \arrow[dr] & & \\
x \arrow[ur] \arrow[r] \arrow[dr] & h_2 \arrow[r] \arrow[dr] & g_1 \arrow[r] & f \\
 & h_3 \arrow[r] & g_2 \arrow[ur] &
\end{tikzcd}$$

Figure 1: A Visual Representation of a Simple Neural Net

Image from: https://en.wikipedia.org/wiki/Artificial_neural_network

7.3.1 Neural Networks

- A convolution neural network is a twist of a normal neural network, which attempts to deal with the issue of high dimensionality.
- For examples by reducing the number of pixels in image classification through two separate phases: the convolution phase, and the pooling phase.
- After that it performs much like an ordinary neural network.

7.3.1 Neural Networks

What are Some Applications of Neural Networks?

- This applications focus on classification of images.
- Let’s have a look at an example:
  - Many companies would love to be able to automate a model which would classify articles of clothing.
  - This would allow them to draw insight into fashion trends, buying habits, and differences between cultural, and socio-economic groups.
  - Neural Network is applied, to see if an adequate classification model can be constructed, when given a set of 60,000 images, with labels identifying what type of clothing they were.
  - All of those pictures are made up of pixels, which since we will be doing a simple neural network, and not a convolution neural net, will be passed directly into the network as a vector of pixels.

7.3.1 Neural Networks

- The number of layers within the model need to specify.
- For the sake of simplicity, the model is limited to two layers.
- One must also select the type of activation for each layer.
- Again, to keep it simple, a sigmoid is selected as activation for the first layers, with the final layer having 10 nodes, and set to return 10 probability scores, indicating the probability whether the image belongs to one of the 10 possible articles of clothing.
- To compile the model, a loss function must be defined, which will be how the model evaluates its own performance.
- An optimizer must also be determined, which is how the information from the cost function is used to change the weights and the bias of each node.

7.3.1 Neural Networks

- With those parameters input, a model can be trained.
- Once the model has been trained, it must be evaluated based on the testing data.
- To accomplish that, one must denote the number of epochs which the model will consider.
- Epochs determine how many iterations through the data shall be done.
- More epochs will be more computationally expensive but should allow for a better fit.
- In this model, 5 epochs are selected.
- The accuracy of the model can be seen on the training data improves after each iteration, as the optimizing function alters the weights.
- The real fit of how a model performs is by running the model on a testing set, which was not used to construct the model.

7.3.2 Regression

- Regression analysis is a subfield of supervised machine learning.
- It aims to model the relationship between a certain number of features and a continuous target variable.
- In regression problems we try to come up with a quantitative answer, like predicting the prices of a house or the number of seconds that someone will spend watching a video.

7.3.2 Regression

Simple Linear Regression: Fitting a Line Through Data

- Having a set of points, the regression algorithm will model the relationship between a single feature (explanatory variable x) and a continuous valued response (target variable y). See Figure 2.
- The model will model this relationship by setting an arbitrarily line and computing the distance from this line to the data points.
- This distance, the vertical lines, are the residuals or prediction's errors.
- The regression algorithm will keep moving the line through each iteration, trying to find the best-fitting line, in other words, the line with the minimum error.
- There are several techniques to perform this task and the ultimate goal is to get the line closer to the maximum number of points.

7.3.2 Regression

Moving The Line

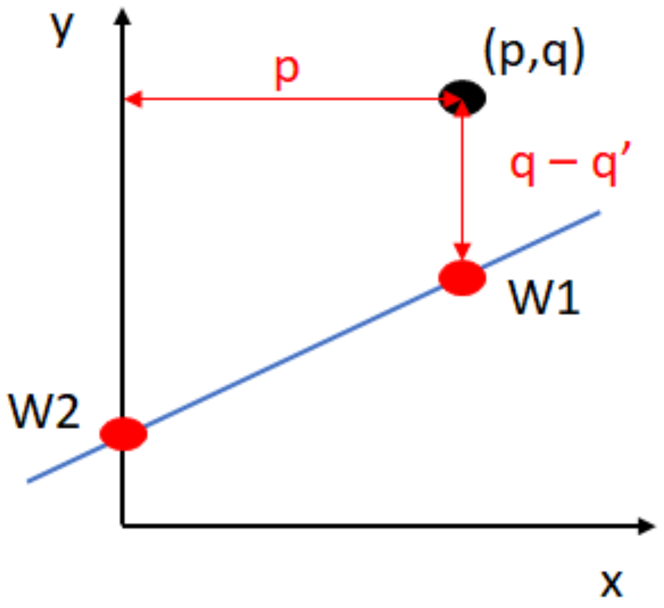


Figure 2: Regression Line

$$\boxed{y = W1x + W2}$$

W1 = Slope

W2 = Intercept

7.3.2 Regression

Absolute Trick

- When having a point and a line, the goal is to get the line closer to this point.
- For achieving this task, the algorithm will use a parameter called “learning rate”.
- This learning rate is the number that will be multiplied to the function parameters in order to make small steps when approximating the line to the point.
- In other words, the learning rate will determine the length of the distance covered in each iteration that will get the line closer to the point.
- It is commonly represented as the α symbol.

$$\boxed{y = (W_1 + p\alpha)x + W_2 + \alpha}$$

7.3.2 Regression

Square Trick

- It is based on the following premise: If there is a point closer to a line, and the distance is small, the line is moved a little distance.
- If it is far, the line will be moved a lot more.

$$\boxed{y = (W_1 + p(q - q')\alpha)x + W_2 + (q - q')\alpha}$$

7.3.2 Regression

Multiple Linear Regression

- Independent variables are also known as predictors, which are variables we look at to make predictions about other variables.
- The Variables that are trying to predict are known as dependent variables.
- When the outcome we are trying to predict depends on more than one variable, a more complicated model can be made that takes this higher dimensionality into account.
- As long as they are relevant to the problem faced, using more predictor variables can help us to get a better prediction.
- As seen before, the following image shows a simple linear regression:

7.3.2 Regression

Simple Regression → $\boxed{y = mx + b}$

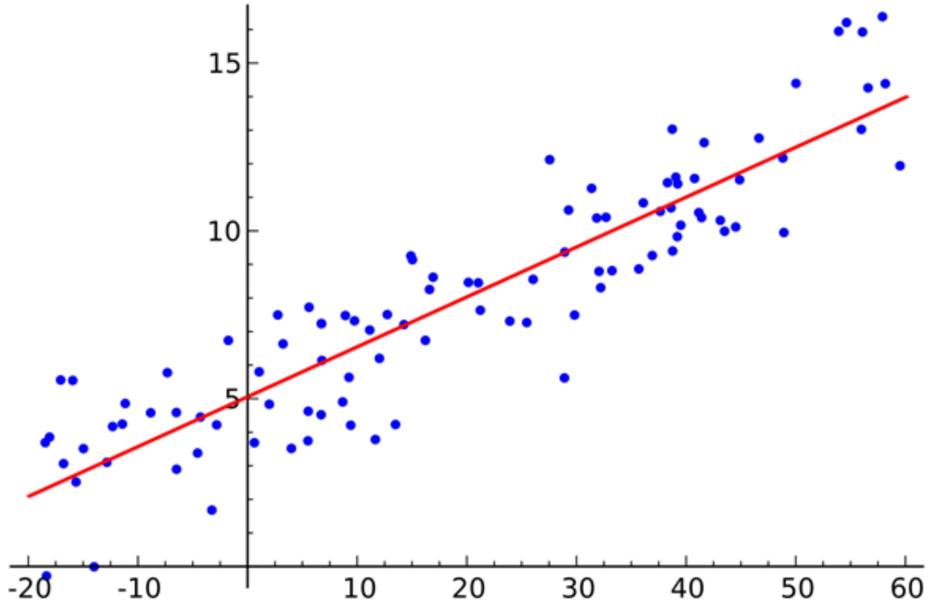


Figure 3: Simple Linear Regression

7.3.2 Regression

- Figure 4 shows a fitted hyperplane of multiple linear regression with two features.

Multiple Regression → $\boxed{y = m_1x_1 + m_2x_2 + b}$

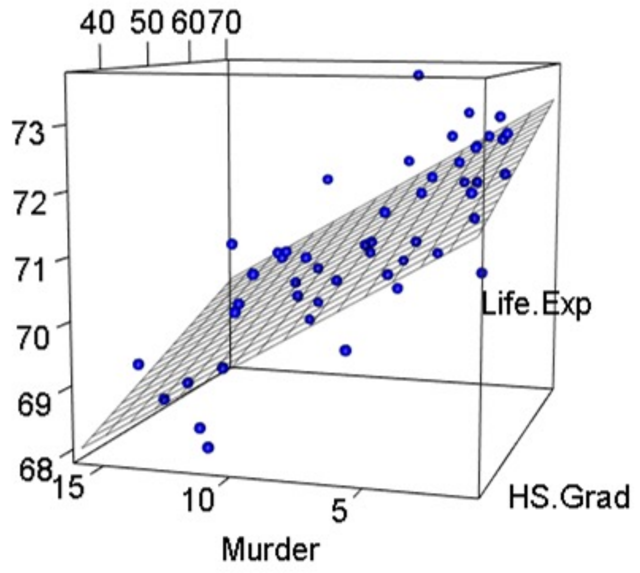


Figure 4: Multiple Regression

7.3.3 K-Nearest Neighbors (KNN)

How does kNN work?

- Let’s start by looking at “k” in the kNN.
- Since the algorithm makes its predictions based on the nearest neighbors, we need to tell the algorithm the exact number of neighbors we want to consider.
- Hence, “k” represents the number of neighbors and is simply a hyperparameter that we can tune.
- Now let’s assume that we have picked k=5, and we have a dataset containing house size (sq.ft.), the year house was built, and house price.
- We want to train kNN on this data and then use it to predict the price of another house that we are interested in. See Figure 5.

7.3.3 K-Nearest Neighbors (KNN)

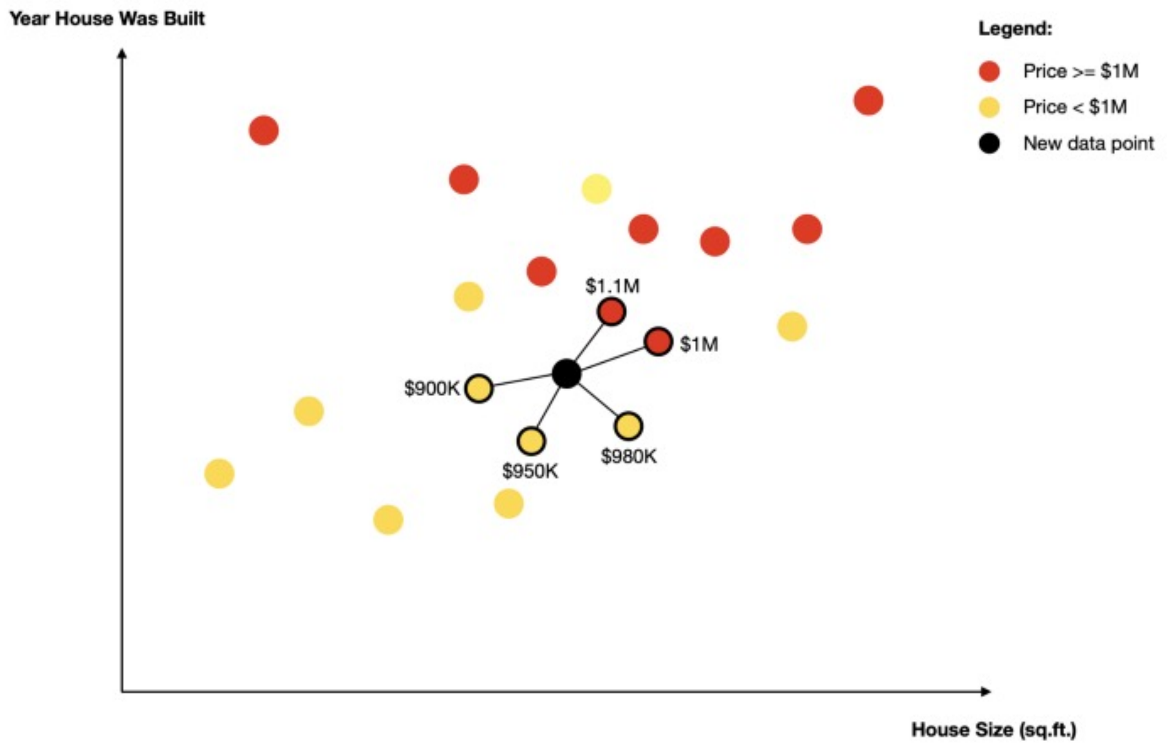


Figure 5: kNN Application

7.3.3 K-Nearest Neighbors (KNN)

- From Figure 5, the black circle represents a new data point (the house we are interested in).
- Since we have set k=5, the algorithm finds five nearest neighbors of this new point.
- Once the neighbors are found, one of the two things will happen depending on whether you are performing classification or regression analysis.

7.3.3 K-Nearest Neighbors (KNN)

- **Classification:**
  - the algorithm uses simple majority voting to assign the label to the new data point.
  - In our example, the majority consists of 3 neighbors with a price<$1M. Hence, the predicted label for the new data point is <$1M.
- **Regression:**
  - the algorithm calculates the average value of all the neighbors.
  - In our example this would be: ($900K + $950K + $980K + $1M + $1.1M) / 5 = $986K.
  - So, the predicted price of a house (new data point) is $986K.

7.3.3 K-Nearest Neighbors (KNN)

- As seen from example, kNN is a very intuitive algorithm, making it easy to explain how the predictions were made.
- One final thing to add, the explanation above showed what happens when uniform weights are being used. i.e., each neighbor carries the same weight in the calculation.
- However, in some cases (e.g., when you have sparse data), it may be beneficial to use distance-based weights.
- Distance weighting assigns weights proportional to the inverse of the distance from the query point, which means that neighbors closer to your data point will carry proportionately more weight than neighbors that are further away.

7.3.4 Decision Trees

- In this part we shall discuss the theory and working behind decision trees.
- We shall see some mathematical aspects of the algorithm i.e. entropy and information gain.

7.3.4 Decision Trees

- Suppose we have following plot for two classes represented by black circle and blue squares. See Figure 6.
- Is it possible to draw a single separation line ? Perhaps **no.**

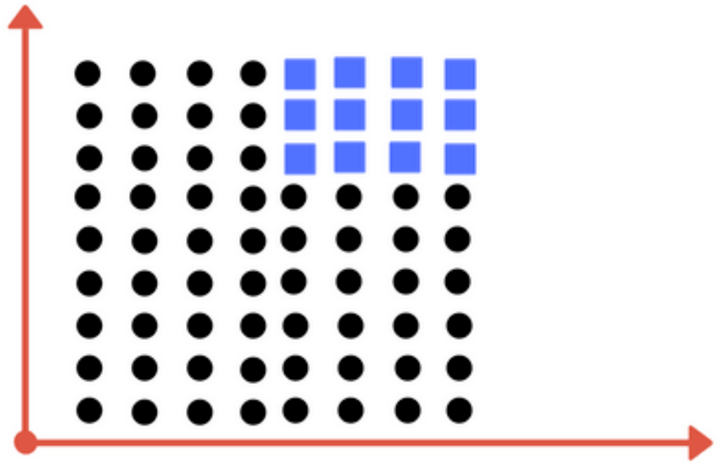


Figure 6 Plot of Two Classes

7.3.4 Decision Trees

- We will need more than one line, to divide into classes.
- Something similar to following image in Figure 7:

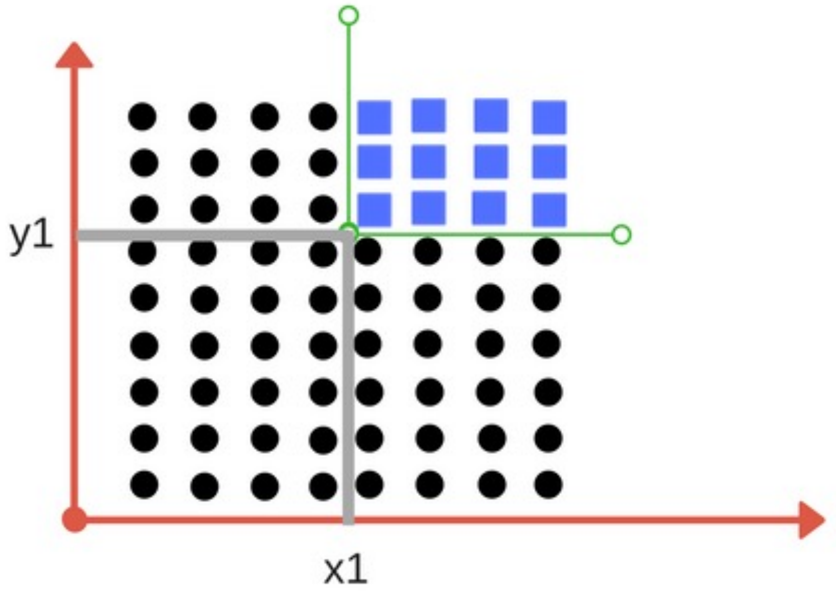


Figure 7 Modified Plot of Two Classes

7.3.4 Decision Trees

- We need two lines here one separating according to threshold value of ***x*** and other for threshold value of ***y.***
- Decision Tree Classifier, repetitively divides the working area(plot) into sub part by identifying lines. (repetitively because there may be two distant regions of same class divided by other as shown in image Figure 8).
- When does it terminate?
    1. Either it has divided into classes that are pure (only containing members of single class)
    2. Some criteria of classifier attributes are met.

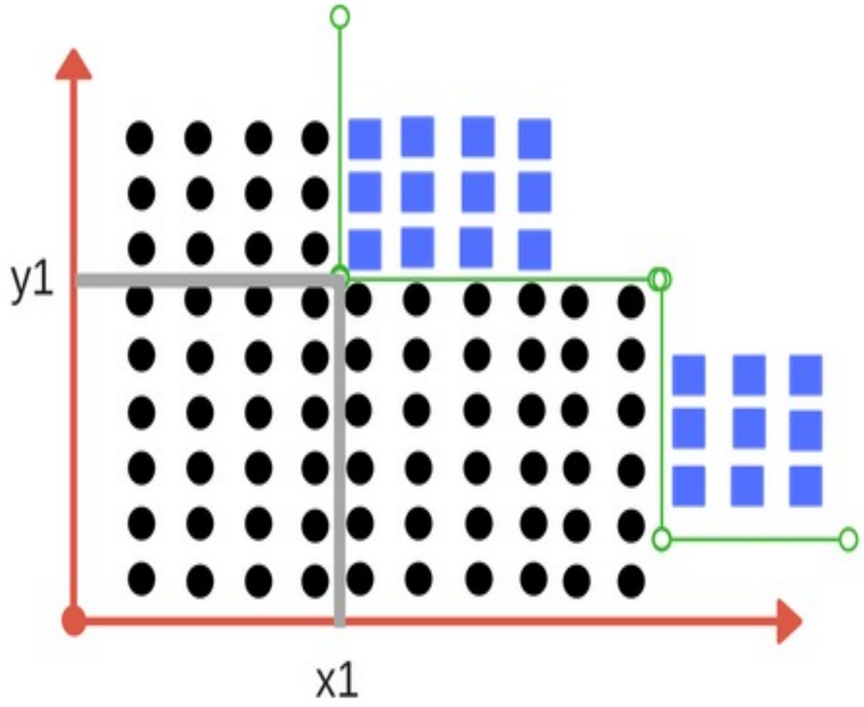


Figure 8 Modified Plot of Two Classes with Subplot

7.3.4 Decision Trees

- *We define few terms related to decision tree and then perform those calculation with sample example.*

1. Impurity

- Impurity is when we have a traces of one class division into other.
- This can arise due to following reason:
  - ❑We run out of available features to divide the class upon.
  - ❑We tolerate some percentage of impurity (we stop further division) for faster performance. (There is always trade off between accuracy and performance).

7.3.4 Decision Trees

- For example, in Figure 9 we may stop our division when we have x number of fewer number of elements left.
- This is also known as ***gini impurity.***

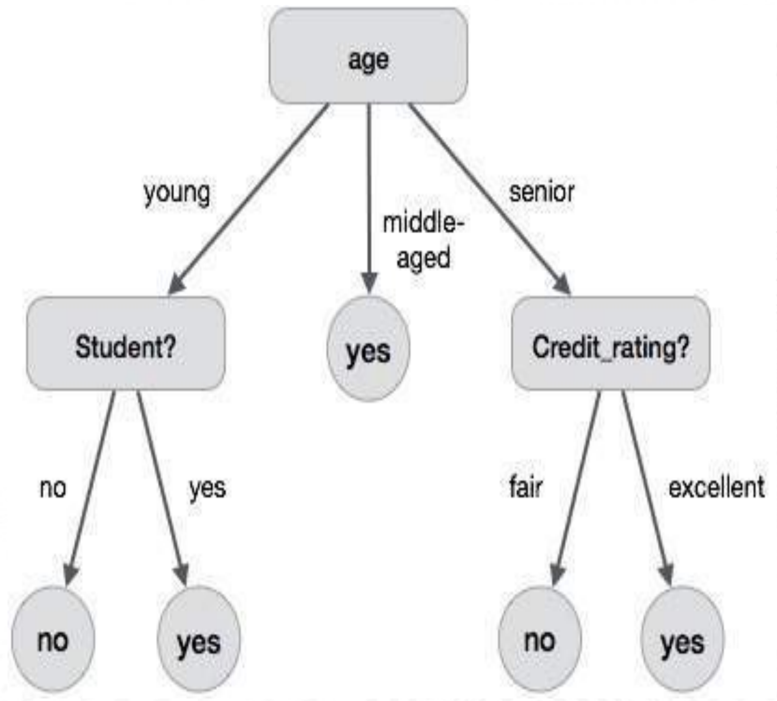


Figure 9 Impurity Example

7.3.4 Decision Trees

2. Entropy

- Entropy is degree of randomness of elements or in other words it is ***measure of impurity.***
- *Mathematically, it can be calculated with the help of probability of the items as:*

$$H = -\sum p(x) \log p(x)$$

- It is negative summation of probability times the log of probability of item x.

7.3.4 Decision Trees

- **For example,**
  if we have items as number of dice face occurrence in a throw event as **1123,**
  the entropy is
  p(1) = 0.5
  p(2) = 0.25
  p(3) = 0.25
  entropy = - (0.5 * log(0.5)) - (0.25 * log(0.25)) -(0.25 * log(0.25)
  = 0.45

7.3.4 Decision Trees

3. Information Gain

- Suppose we have multiple features to divide the current working set.
- What feature should we select for division? Perhaps one that gives us less impurity.
- Suppose we divide the classes into multiple branches as follows, the information gain at any node is defined as:

Information Gain (n) =
*Entropy(x) — ([weighted average] * entropy(children for feature))*

7.3.4 Decision Trees

- Suppose we have following class to work with initially: 112234445
- Suppose we divide them based on property: divisible by 2
- *Entropy at root level : 0.66*
  Entropy of left child : 0.45

  *weighted value = (4/9) * 0.45 = 0.2*
  Entropy of right child: 0.29

  *weighted value = (5/9) * 0.29 = 0.16*

  Information Gain** = 0.66 - [0.2 + 0.16] = **0.3

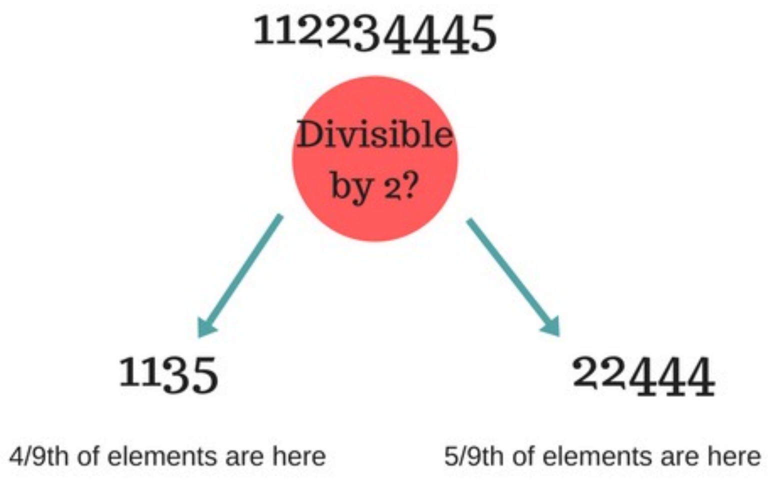


Figure 10 Example Information Gain

7.3.5 Support Vector Machines

- A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane.
- In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples.
- In two-dimensional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side.

7.3.5 Support Vector Machines

- Suppose you are given plot of two label classes on graph as shown in Figure 11.
- Can you decide a separating line for the classes?

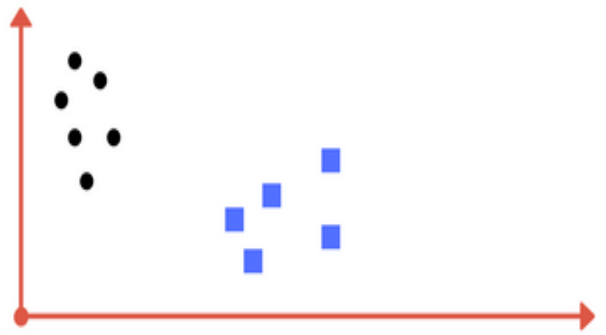


Figure 11: Black Circles and Blue Squares

7.3.5 Support Vector Machines

- You might have come up with something similar to Figure 12
- It fairly separates the two classes.
- Any point that is left of line falls into black circle class and on right falls into blue square class.
- ***Separation of classes. That's what SVM does.***
- It finds out a line/ hyper-plane (in multidimensional space that separate outs classes).

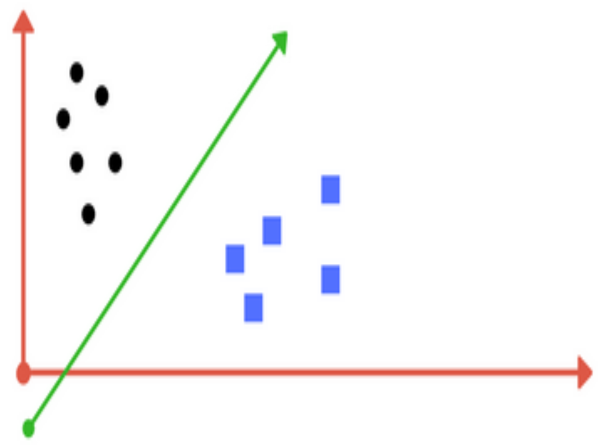


Figure 12: Sample Divided into Two Classes

7.3.5 Support Vector Machines

- **Tuning parameters: Kernel, Regularization, Gamma and Margin.**

1. Kernel

- The learning of the hyperplane in linear SVM is done by transforming the problem using some linear algebra.
- This is where the kernel plays role. Figure 13a and 13b shows the kernel.

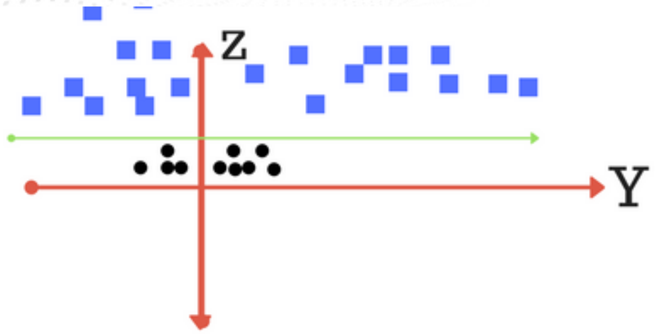


Figure 13a: Kernel View from ZY Plane

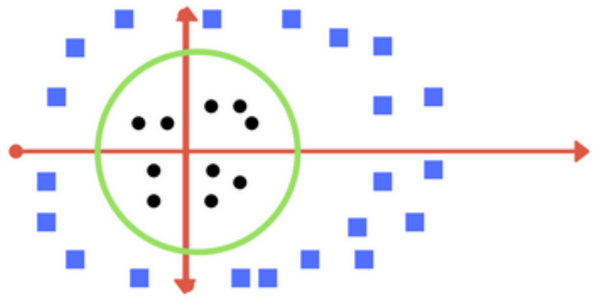


Figure 13b: Kernel View from XY Plane

7.3.5 Support Vector Machines

2. Regularization

- The Regularization parameter (C) tells the SVM optimization how much you want to avoid misclassifying each training example.
- For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly.
- Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points.

7.3.5 Support Vector Machines

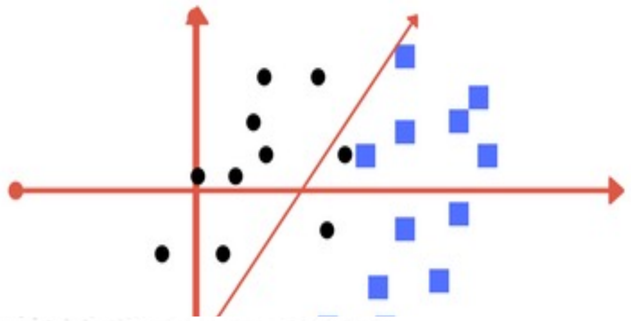


Figure 14a: Low Regularization Value

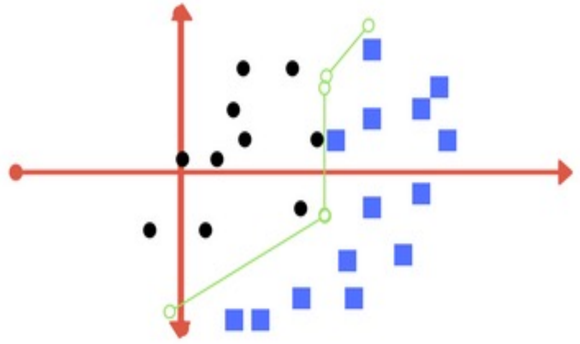


Figure 14b: High Regularization Value

7.3.5 Support Vector Machines

3. Gamma

- The gamma parameter defines how far the influence of a single training example reaches, with low values meaning ‘far’ and high values meaning ‘close’.
- In other words, with low gamma, points far away from plausible separation line are considered in calculation for the separation line.
- Where as high gamma means the points close to plausible line are considered in calculation.

7.3.5 Support Vector Machines

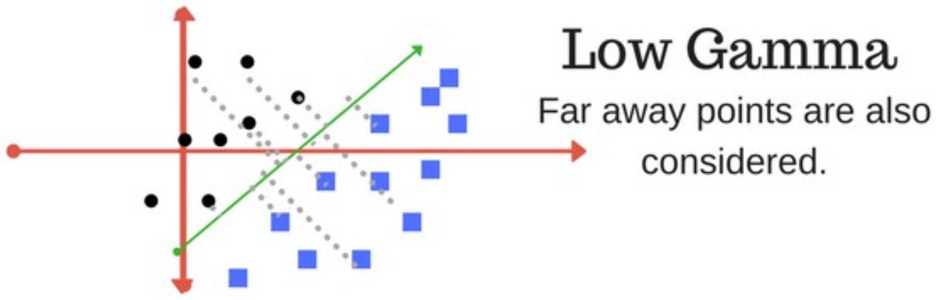


Figure 15a Low Gamma

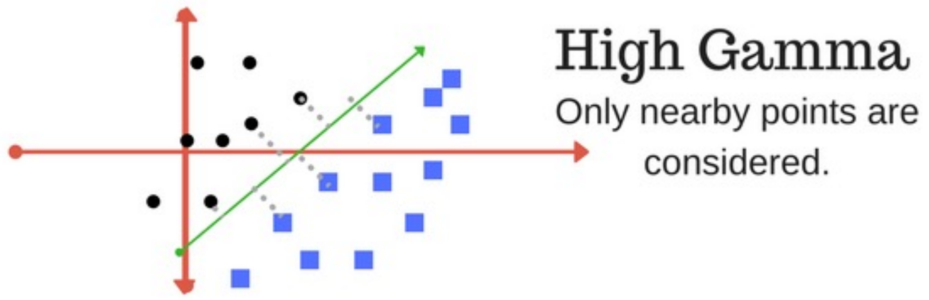


Figure 15b High Gamma

7.3.5 Support Vector Machines

4. Margin

- ***A margin is a separation of line to the closest class points.***
- A ***good margin*** is one where this separation is larger for both the classes.
- Figure 16a and 16b gives to visual example of good and bad margin respectively.
- A good margin allows the points to be in their respective classes without crossing to other class.

7.3.5 Support Vector Machines

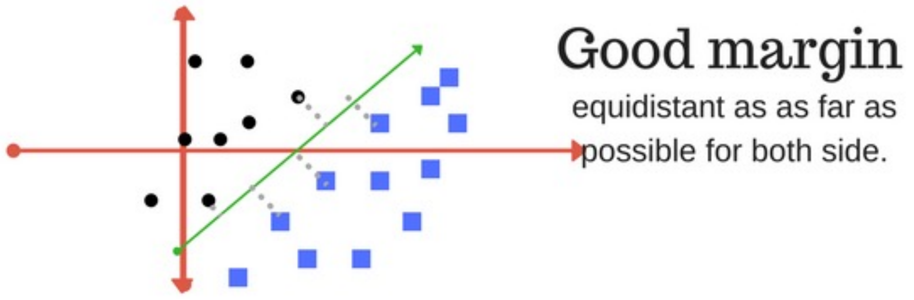


Figure 16a Good Margin

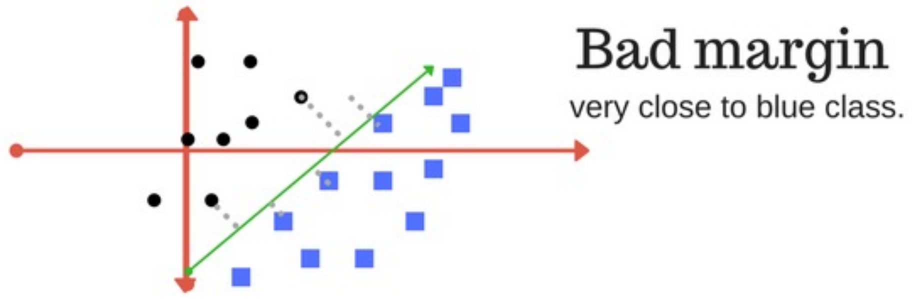


Figure 16b Bad Margin

REFERENCES

- https://towardsdatascience.com/classification-using-neural-networks-b8e98f3a904f
- https://medium.com/towards-data-science/supervised-learning-basics-of-linear-regression-1cbab48d0eba
- https://medium.com/p/1c1ff404d265
- https://towardsdatascience.com/k-nearest-neighbors-knn-how-to-make-quality-predictions-with-supervised-learning-d5d2f326c3c2
- https://medium.com/machine-learning-101/chapter-3-decision-trees-theory-e7398adac567
- https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72

UTHM Universiti Tun Hussein Onn Malaysia 1993-2023 AMANAH • PROFESIONAL • INOVATIF

Thank you