# CHAPTER 6: Classification Algorithms

## Introduction

In the previous chapter, we studied the nonparametric approach to solving the Linear Regression, in this, we are going to look at Machine Learning algorithms to solve the Classification problem statement

A common job of machine learning algorithms is to recognize objects are being able to separate them into categories. This process is called classification, and it helps us segregate vast quantities of data into discrete values, i.e.: distinct, like 0/1, Ture/False, or a pre-defined output label class.

## What is Supervised Learning

Before we dive into Classification, let’s look at what Supervised Learning is. In supervised learning, you train your model on the labeled dataset which means we have both row input data as well as results. We split our data into a training dataset and test dataset where the training data is used to train our network whereas the test dataset acts as new data for predicting results or to seeing the accuracy of our model.

Hence, in supervised learning, our model learns from seen results the same as a teacher teaches his students because the teacher already knows the results. Accuracy is what we achieve in supervised learning as model perfection is usually high.

Some examples of Supervised Learning include:

1. It classifies spam Detection by teaching a model of what mail is spam and not spam.
2. Speech recognition is where you teach a machine to recognize your voice.
3. Object Recognition is by showing a machine what an object looks like and having it pick that object from among other objects.

We further divide Supervised Learning into the following:

Diagram

Description automatically generated

## What is Classification?

Classification is a task of Machine Learning which assigns a label value to a specific class and then can identify a particular type to be of one kind or another. It is the process of recognition, understanding, and grouping of objects and ideas into pre-set categories in machine learning programs that leverage a wide range of algorithms to classify future datasets into respective and relevant categories.

In short, a classification is a form of “pattern recognition”. Here, classification algorithms applied to the training data find the same pattern (similar number sequences, words or sentiments, and the like) in future data sets.

Diagram

Description automatically generated

There are mainly 4 different types of classification tasks that you might encounter in your day-to-day challenges. Generally, the different types of predictive models in machine learning are as follows:

* Binary Classification
* Multi-Label Classification
* Multi-Class Classification
* Imbalanced Classification

We will go over them one by one.

## Binary Classification for Machine Learning

A binary classification refers to those tasks which can give either of any two class labels as the output. Generally, one is considered as the normal state and the other is the abnormal state. The following examples will help you to understand them better.

* Email Spam detection: Normal State – Not Spam, Abnormal State – Spam
* Conversion prediction: Normal State – Not churned, Abnormal State – Churn
* Conversion Prediction: Normal State – Bought an item, Abnormal State – Not bought an item

You can also add the example of “No cancer detected” to be a normal state and “Cancer detected” to be the abnormal state. The notation mostly followed is that the normal state gets assigned the value of 0 and the class with the abnormal state gets assigned the value of 1. For each example, one can also create a model which predicts the Bernoulli probability for the output. In short, it returns a discrete value that covers all cases and will give the output as either the outcome will have a value of 1 or 0. Hence after the association to two different states, the model can give an output for either of the values present.

The most popular algorithms which are used for binary classification are:

* K-Nearest Neighbours
* Logistic Regression
* Support Vector Machine
* Decision Trees
* Naive Bayes

Out of the mentioned algorithms, some algorithms were specifically designed for binary classification and natively do not support more than two types of class. Some examples of such algorithms are Support Vector Machines and Logistic Regression.

We will use the make\_blob() function of the scikit-learn module to generate a binary classification dataset. The example below uses a dataset with 1000 examples that belong to either of the two classes present with two input features.

Example:

Text

Description automatically generated

Output:

Text

Description automatically generated

Chart, scatter chart

Description automatically generated

The above example creates a dataset of 5000 samples and divides them into input ‘X’ and output ‘Y’ elements. The distribution shows us that any one instance can either belong to either class 0 or class 1 and there are approximately 50% in each.

The first 10 examples in the dataset are shown with the numeric input values and the target value is an integer which represents a class membership.

Then a scatter plot is created for the input variables where the resultant points are color-coded based on the class value. We can easily see two distinct clusters which we can discriminate.

## Multi-Class Classification

These types of classification problems have no fixed two labels but can have any number of labels. Some popular examples of multi-class classification are:

* Plant Species Classification
* Face Classification
* Optical Character Classification

Here there is no notion of a normal and abnormal outcome, but the result will belong to one of the many among a range of variables of known classes. There can also be a huge number of labels like predicting a picture as to how closely it might belong to one out of the tens of thousands of the faces of the recognition system.

Another type of challenge where you need to predict the next word of a sequence like a translation model for text could also be considered as multi-class classification. In this scenario, all the words of the vocabulary define all the possible number of classes and that can range in millions.

These types of models are generally done using a Categorical Distribution unlike Bernoulli for binary classification. In a Categorical Distribution, an event can have multiple endpoints or results and hence the model predicts the probability of input concerning each of the output labels.

The most common algorithms which are used for Multi-Class Classification are:

* K-Nearest Neighbours
* Naïve Bayes
* Decision trees
* Random Forest

You can also use the algorithms for Binary Classification here on a basis of either one class vas all the other classes, also known as one-vs-rest, or one model for a pair of classes in the model which is also known as one-vs-one.

One Vs Rest – The main task here is to fit one model for each class which will be versus all other classes.

One Vs One – The main task here is to define a binary model for every pair of classes.

We will again take the example of multi-class classification by using the make\_blobs() function of the scikit learn module.

The following code demonstrates it.

Example:

Graphical user interface, text, application, email

Description automatically generated

Output:

Text

Description automatically generated with medium confidence

Chart, scatter chart, bubble chart

Description automatically generated

## Multi-Label Classification for Machine Learning

In multi-label classification, we refer to those specific classification tasks where we need to assign two or more specific class labels that could be predicted for each example. A basic example can be photo classification where a single photo can have multiple objects in it like a dog or an apple etc. The main difference is the ability to predict multiple labels and not just one.

You cannot use a binary classification model or multi-class classification model for multi-label classification, and you must use a modified version of the algorithm to incorporate multiple classes which can be possible and then to look for them all. It becomes more challenging than a simple yes or no statement.

One more approach is to use a separate classification algorithm for the label prediction for every type of class.

We will use a library from scikit-learn to generate our multi-label classification dataset from scratch. The following code creates and shows the working example of multi-label classification of 1000 samples and 4 types of classes.

Example:

Text

Description automatically generated

A picture containing text

Description automatically generated

## Imbalanced Classification for Machine Learning

An imbalanced Classification refers to those tasks where the number of examples in each of the classes are unequally distributed. Generally, imbalanced classification tasks are binary classification jobs where a major portion of the training dataset is of normal class type and a minority of them belong to the abnormal class.

The most important examples of these use cases are:

* Fraud Detection
* Outlier Detection
* Medical Diagnosis Test

The problems are transformed into binary classification tasks with some specified techniques. You can either utilize undersampling for the majority classes or oversampling for the minority classes. The most prominent examples are:

* Random Undersampling
* SMOTE Oversampling

Special modeling algorithms can be used to give more attention to the minority class when the model is being fitted on the training dataset which includes cost-sensitive machine learning models. Especially for cases like:

* Cost-Sensitive Logistic Regression
* Cost-Sensitive Decision Trees
* Cost-Sensitive Support Vector Machines

So after choosing the model, we need to access the model and score of it for which we can either use Precision, Recall, or F-Measure score. Now we will take a look to develop a dataset for the imbalanced classification problem.

We will use the Classification function of scikit-learn to generate a fully synthetic and imbalanced binary classification dataset of 1000 samples.

Example:

Text

Description automatically generated

Text

Description automatically generated

Chart, scatter chart

Description automatically generated

Here we can see the distribution of the labels and we can see a severe imbalance of the classes where 983 elements belong to one type and only 17 belong to the other type. We can see most of the type 0 or class 0 as expected. These types of datasets are no more difficult to identify because they have a more general and practical use case.

**Classification Algorithms**

Let’s start understanding the classification algorithms one by one.



K-nearest neighbors (KNN) is a type of supervised learning algorithm which is used for both regression and classification purposes, but mostly it is used for the latter. Given a dataset with different classes, KNN tries to predict the correct class of test data by calculating the distance between the test and all the training points. It then selects the k points which are closest to the test data. Once the point is selected, the algorithm calculates the probability (in the case of classification) of the test point belonging to the classes of the K training points, and the class with the highest probability is selected. In the case of a regression problem, the predicted value is the mean of the k selected training points.

Let’s understand this with an illustration:

1. Given a training dataset as given below. We have new test data that we need to assign to one of the two classes.

Shape

Description automatically generated

1. Now, the KNN algorithm calculates the distance between the test data and the given training data.

Shape, arrow

Description automatically generated

1. After calculating the distance, it will select the K training points which are nearest to the test data. Let’s assume the value K is 3 for our example.

Shape

Description automatically generated with medium confidence

1. Now, 3 nearest neighbors are selected, as shown in the figure above. Let’s see in which class our test data will be assigned:

Number of Green class values = 2 Number of Red class values = 1 Probability (Green) = 2/3 Probability (Red) = 1/3

Since the probability for the Green class is higher than the Red, the KNN algorithm will assign the test data to the Green class.

Similarly, if this were the case of a regression problem, the predicted value of the test data will simply be the mean of all the 3 nearest values.

This is the basic working algorithm for KNN. Let’s understand how the distance is calculated:

**Euclidean Distance:**

It is the most commonly used method to calculate the distance between two points. The Euclidean distance between two points p(p1, p2) and q(q1, q2) is calculated as:

Diagram

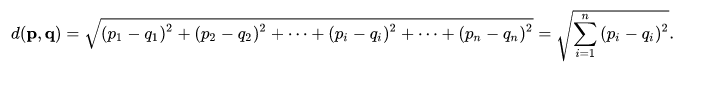
Description automatically generated

Image source: Wikipedia

Text

Description automatically generated

Similarly, for n-dimensional space, the Euclidean distance is given as:



**Manhattan Distance:**

The Manhattan distance is also known as the L1 norm, taxicab norm, rectilinear distance, or City blocks distance. This distance represents the sum of the absolute differences between the opposite values in vectors.

A picture containing graphical user interface

Description automatically generated

Manhattan Distance is less influenced by outliers than the Euclidean distance. With very high dimensional data it is more preferred.

**Hamming Distance:**

Hamming distance is a distance metric that measures the number of mismatches between two vectors. It is mostly used in the case of categorical data.

A picture containing text

Description automatically generated

Generally, if we have features as categorical data then we consider the difference to be 0 if both the values are the same and the difference is 1 if both the values are different.

**Choosing the value of K**

The value of K affects the KNN classifier drastically. The flexibility of the model decreases with the increase of ‘K’. With a lower value of ‘K’ variance is high and bias is low but as we increase the value of ‘K’ variance starts decreasing and bias starts increasing. With very low values of ‘K’, there is a chance of algorithm Overfitting the data whereas with a very high value of ‘K’ there is a chance of Underfitting. Let’s visualize the trade-off between ‘1/K’, ‘train error rate’, ‘test error rate’:

Chart, line chart

Description automatically generated

Image source: “ISLR”

We can see that the train error rate increases with the increase in the value of ‘K’ whereas the test error rate decreases initially and then increases again. So, our goal should be to choose such value of ‘K’ for which we get a minimum of both the errors and avoid Overfitting as well as Underfitting. We use different ways to calculate the optimum value of ‘K’ such as cross-validation, error versus ‘K’ curve, checking accuracy for each value of ‘K’ etc.

**Pros and Cons of KNN Algorithm**

**Pros:**

* It can be used for both regression and classification problems.
* It is very simple and easy to implement.
* The mathematics behind the algorithm is easy to understand.
* There is no need to create a model or do hyper-parameter tuning.
* KNN doesn’t make any assumption for the distribution of the given data.
* There is not much time cost in the training phase.

**Cons:**

* Finding the optimum value of ‘K’
* It takes a lot of time to compute the distance between each test sample and all training samples.
* Since the model is not saved beforehand in this algorithm, every time one predicts a test value, it follows the same steps again and again.
* Since we need to store the whole training set for every test set, it requires a lot of space.
* It is not suitable for high-dimensional data.
* It is not suitable for high-dimensional data.
* Expensive in the testing phase



**Bayes’s Theorem**

In probability theory and statistics, “Bayes’s Theorem” describes the probability of an event, based on prior knowledge of conditions that might be related to the event. Mathematically, it can be written as

Text

Description automatically generated

Where A and B are events and P(B)≠0

* P(A/B) is a conditional probability: the likelihood of event A occurring given that B is true.
* P(B/A) is also conditional probability: the likelihood of event B occurring given that A is true.
* P(A) and P(B) are the probabilities of observing A and B respectively; they are known as the marginal probability.

Let’s understand it with the help of an example:

**The problem statement:**

There are two machines which manufacture bulbs. Machine 1 produces 30 bulbs per hour and machine 2 produces 20 bulbs per hour. Out of all bulbs produced, 1 % turns out to be defective. Out of all the defective bulbs, the share of each machine is 50%. What is the probability that a bulb produced by machine 2 is defective?

We can write the information given above in mathematical terms as:

The probability that a bulb was made by Machine 1, P(M1)=30/50=0.6

The probability that a bulb was made by Machine 2, P(M2)=20/50=0.4

The probability that a bulb is defective, P(Defective)=1%=0.01

The probability that a defective bulb came out of Machine 1, P(M1 | Defective)=50%=0.5

The probability that a defective bulb came out of Machine 2, P(M2 | Defective)=50%=0.5

Now, we need to calculate the probability of a bulb produced by machine 2 being defective i.e., P(Defective | M2). Using the Bayes Theorem above, it can be written as:



Substituting the values, we get



We will extend this same to understand the Naive Baye’s Algorithm.

Algorithm steps:

1. Let’s consider that we have a binary classification problem i.e., we have two classes in our data as shown below:

Chart, scatter chart

Description automatically generated

1. Now suppose if we are given a new data point, which class does that point belong to?

Chart, scatter chart

Description automatically generated

1. The formula for a point ‘X’ belonging in class1 can be written as:

Diagram, text, letter

Description automatically generated

Where the numbers represent the order in which we are going to calculate different probabilities.

1. A similar formula can be utilized for class 2 as well.
2. The probability of class 1 can be written as:



1. For calculating the probability of X, we draw a circle around the new point and see how many points (excluding the new point) lie inside that circle:

Diagram

Description automatically generated

The points inside the circle are considered to be similar points.



1. Now, we need to calculate the probability of a point to be in the circle that we have made given that it’s of class 1.



1. We can substitute all the values into the formula in step 3. We get:



1. And if we calculate the probability that X belongs to Class2, we’ll get 0.69. It means that our point belongs to class 2.

**The Generalization for Multiclass:**

The approach discussed above can be generalized for multiclass problems as well. Suppose, P1, P2, P3…Pn is the probabilities for the classes C1, C2, C3…Cn, then the point X will belong to the class for which the probability is maximum. Or mathematically the point belongs to the result of:



**Why it is called Naïve Bayes?**

The entire algorithm is based on Baye’s theorem to calculate probability. So, it also carries forward the assumptions for Bayes’s theorem. But those assumptions (that the features are independent) might not always be true when implemented over a real-world dataset. So, those assumptions are considered Naïve and hence the name.



Logistic regression is one such regression algorithm that can be used for performing classification problems. It calculates the probability that a given value belongs to a specific class. If the probability is more than 50%, it assigns the value in that particular class else if the probability is less than 50%, the value is assigned to the other class. Therefore, we can say that logistic regression acts as a binary classifier.

**Working of a Logistic Regression**

For linear regression, the model is defined by: y = mx+c, and for logistic regression, we calculate probability, i.e. y is the probability of a given variable x belonging to a certain class. Thus, it is obvious that the value of y should lie between 0 and 1.

But, when we use equation(i) to calculate probability, we would get values less than 0 as well as greater than 1. That doesn’t make any sense. So, we need to use such an equation which always gives values between 0 and 1, as we desire while calculating the probability.

**Sigmoid Function**

We use the sigmoid function as the underlying function in Logistic regression. Mathematically and graphically, it is shown as:

Chart

Description automatically generated

**Why do we use the sigmoid function?**

1. The sigmoid function’s range is bounded between 0 and 1. Thus it’s useful in calculating the probability for the Logistic function.
2. Its derivative is easy to calculate than other functions which are useful during gradient descent calculation.
3. It is a simple way of introducing non-linearity to the model.

Although there are other functions as well, which can be used, sigmoid is the most common function used for logistic regression.

**Advantages and Disadvantages of Logistic Regression**

The main advantage of logistic regression is that it is much easier to set up and train than other machine learning and AI applications.

Another advantage is that it is one of the most efficient algorithms when the different outcomes or distinctions represented by the data are linearly separable. This means that you can draw a straight line separating the results of logistic regression calculation.

One of the disadvantages is, if the number of observations is lesser than the number of features, Logistic Regression should not be used, otherwise, it may lead to overfitting.



In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is the number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well (look at the below snapshot)

Chart, scatter chart

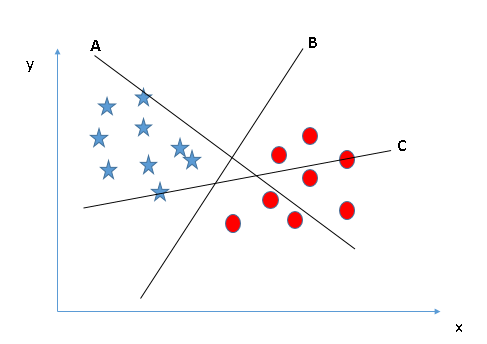
Description automatically generated

Support Vectors are simply the coordinates of individual observation. The SVM classifier is a frontier that best segregates the two classes (hyper-plane/line).

**How does it work?**

Above, we got accustomed to the process of segregating the two classes with a hyper-plane. Now the burning question is “How can we identify the right hyper-plane?”. Don’t worry, it’s not as hard as you think!

Let’s understand:

* **Identify the right hyper-plane (Scenario-1):** Here, we have three hyper-planes (A, B, and C). Now, identify the right hyper-plane to classify stars and circles.  
  

You need to remember a thumb rule to identify the right hyper-plane: “Select the hyper-plane which segregates the two classes better”. In this scenario, hyper-plane “B” has excellently performed this job.

* **Identify the right hyper-plane (Scenario-2):**Here, we have three hyper-planes (A, B, and C) and all are segregating the classes well. Now, How can we identify the right hyper-plane?

Chart, scatter chart

Description automatically generated

Here, maximizing the distances between the nearest data point (either class) and hyper-plane (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called a Margin.

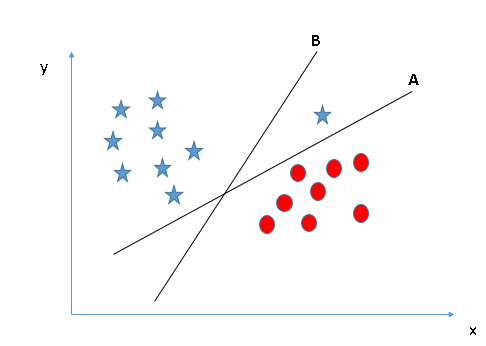
Let’s look at the below snapshot:

A picture containing sky

Description automatically generated

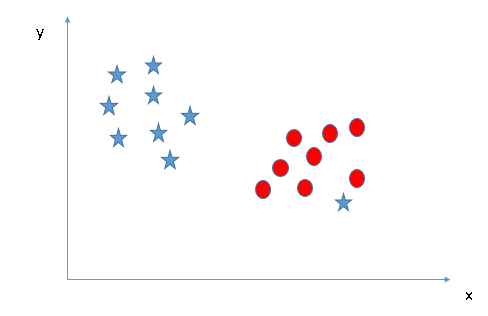
Above, you can see that the margin for hyper-plane C is high as compared to both A and B. Hence, we name the right hyper-plane as C. Another lighting reason for selecting the hyper-plane with a higher margin is robustness. If we select a hyper-plane having a low margin then there is a high chance of miss-classification.

* **Identify the right hyper-plane (Scenario-3):** Hint: Use the rules as discussed in the previous section to identify the right hyper-plane

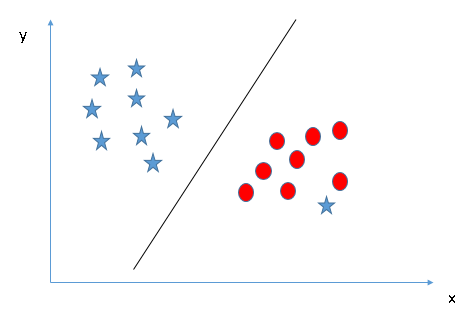
****

Some of you may have selected the hyper-plane B as it has a higher margin compared to A. But, here is the catch, SVM selects the hyper-plane which classifies the classes accurately before maximizing margin. Here, hyper-plane B has classification error and A has classified all correctly. Therefore, the right hyper-plane is A.

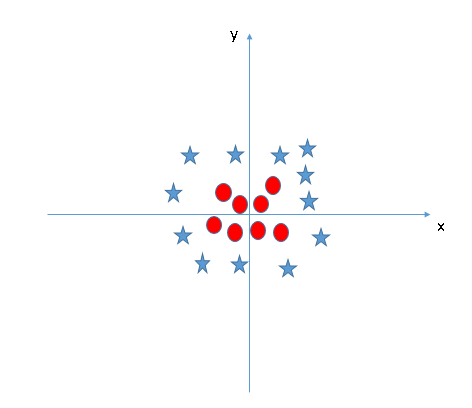
* **Can we classify two classes (Scenario-4)?:** Below, I am unable to segregate the two classes using a straight line, as one of the stars lies in the territory of the other(circle) class as an outlier.



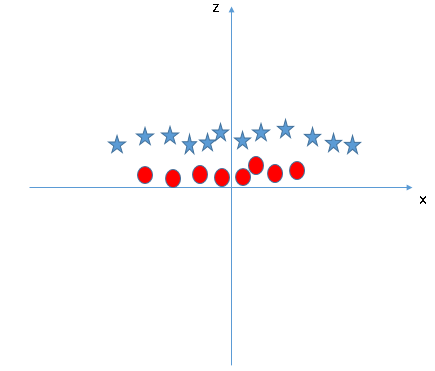
As I have already mentioned, one star at the other end is like an outlier for star class. The SVM algorithm has a feature to ignore outliers and find the hyper-plane that has the maximum margin. Hence, we can say, SVM classification is robust to outliers.



* **Find the hyper-plane to segregate into classes (Scenario-5):** In the scenario below, we can’t have a linear hyper-plane between the two classes, so how does SVM classify these two classes? Till now, we have only looked at the linear hyper-plane.



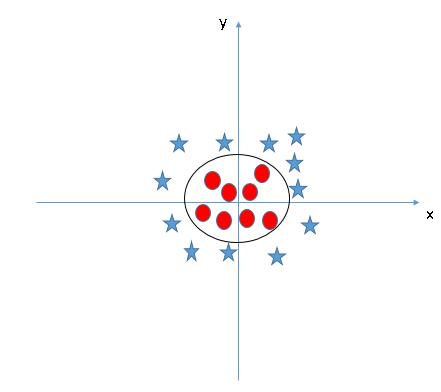
SVM can solve this problem. Easily! It solves this problem by introducing the additional feature. Here, we will add a new feature z=x^2+y^2. Now, let’s plot the data points on axis x and z:

[](https://www.analyticsvidhya.com/wp-content/uploads/2015/10/SVM_9.png)In the above plot, points to consider are:

* + All values for z would be positive always because z is the squared sum of both x and y
  + In the original plot, red circles appear close to the origin of the x and y axes, leading to a lower value of z and a star relatively away from the original result to the higher value of z.

In the SVM classifier, it is easy to have a linear hyper-plane between these two classes. But another burning question that arises is, should we need to add this feature manually to have a hyper-plane. No, the SVM algorithm has a technique called the [kernel](https://en.wikipedia.org/wiki/Kernel_method) trick. The SVM kernel is a function that takes low dimensional input space and transforms it to a higher dimensional space i.e., it converts not separable problem to separable problem. It is mostly useful in non-linear separation problems. Simply put, it does some extremely complex data transformations, then finds out the process to separate the data based on the labels or outputs you’ve defined

When we look at the hyper-plane in original input space it looks like a circle:



**Advantages and Disadvantages of SVM Classifier**

**Advantages**

* It works well with a clear margin of separation
* It is effective in high-dimensional spaces.
* It is effective in cases where the number of dimensions is greater than the number of samples.
* It uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.

**Disadvantages**

* It doesn’t perform well when we have large data set because the required training time is higher
* It also doesn’t perform very well, when the data set has more noise i.e. target classes are overlapping
* SVM doesn’t directly provide probability estimates, these are calculated using an expensive five-fold cross-validation. It is included in the related SVC method of the Python scikit-learn library.



Regression trees are used for quantitative data. In the case of qualitative data or categorical data, we use classification trees. In regression trees, we split the nodes based on RSS criteria, but in classification, it is done using classification error rate, Gini impurity, and entropy. Let’s understand these terms in detail.

Multiple algorithms are used by the decision tree to decide the best split for the problem. Like IDE while using Entropy and Information gain. The second one is Gini Impunity.

Let’s first look at the most common and popular out of all them, which is IDE

**IDE**

**Entropy**

Entropy is the measure of randomness in the data. In other words, it gives the impurity present in the dataset.

Bubble chart

Description automatically generated with medium confidence

When we split our nodes into two regions and put different observations in both the regions, the main goal is to reduce the entropy i.e. reduce the randomness in the region and divide our data cleanly than it was in the previous node. If splitting the node doesn’t lead to entropy reduction, we try to split based on a different condition, or we stop. A region is clean (low entropy) when it contains data with the same labels and random if there is a mixture of labels present (high entropy). Let’s suppose there are ‘m’ observations, and we need to classify them 1 and 2. Let’s say that category 1 has ‘n’ observations and category 2 has ‘m-n’ observations.

p= n/m and q = m-n/m = 1-p

then, entropy for the given set is:



When all the observations belong to category 1, then p = 1 and all observations belong to category 2, then p =0, in both cases E =0, as there is no randomness in the categories. If half of the observations are in category 1 and another half in category 2, then p =1/2 and q =1/2, and the entropy is maximum, E =1.

Chart, line chart

Description automatically generated

**Information Gain**

Information gain calculates the decrease in entropy after splitting a node. It is the difference between entropies before and after the split. The more information gained, the more entropy is removed.

Text

Description automatically generated

Where T is the parent node before split and X is the split node from T.

A tree that split on basis of entropy and information gain value looks like this:

A picture containing text, clock

Description automatically generated

Now let’s look at the Gini Impurity method to build the decision tree.

**Gini Impurity**

It measures the impurity of the nodes and is calculated as:

A picture containing text

Description automatically generated

Were,

A picture containing text, watch, clock

Description automatically generated

Let’s first understand what Gini is and then I’ll show you how you can calculate the Gini impurity for split and decide the right split. Let’s say we have a node like this

Qr code

Description automatically generated

So, what Gini says is that if we pick two points from a population at random, the pink ones highlighted here, then they must be from the same class. Let’s say we have a completely pure node

Qr code

Description automatically generated

Can you guess what would be the probability that a randomly pricked point will be below the same class? Well, obviously it will be 1 since all the points here belong to the same class. So, no matter which two points you picked, they will always belong to that one class and hence the probability will always be 1 if the node is pure. And that is what we want to achieve using Gini.

Gini ranges from zero to one, as it is a probability and the higher this value, the more will be the purity of the nodes. And of course, a lesser value means lesser pure nodes.

**How to calculate the Gini Impurity for a split**

1. Calculate Gini for sub-nodes using the success(p) and failure(q) formulas (p2+q2).
2. Calculate the Gini Impurity for each split node using the weighted Gini score.

Entropy and Gini impurity can be used reversibly. It doesn’t affect the result much. Although, Gini is easier to compute than entropy since entropy has a ‘log’ term calculation. That’s why the CART algorithm uses

If we plot Gini vs entropy graph, we can see there is not much difference between them:

Diagram

Description automatically generated with medium confidence

**Advantages of Decision Tree:**

* It can be used for both Regression and Classification problems.
* Decision Trees are very easy to grasp as the rules of splitting are mentioned.
* Complex decision tree models are very simple when visualized. It can be understood just by visualizing.
* Scaling and normalizing are not needed.

**Disadvantages of Decision Tree:**

* A small change in data can cause instability in the model because of the greedy approach.
* The probability of overfitting is very high for Decision Trees.
* It takes more time to train a decision tree model than other classification algorithms.



Decision trees are one of such models which have low bias but high variance. We have studied that decision trees tend to overfit the data. So bagging technique becomes a very good solution for decreasing the variance in a decision tree. Instead of using a bagging model with the underlying model as a decision tree, we can also use Random forest which is more convenient and well optimized for decision trees. The main issue with bagging is that there is not much independence among the sampled datasets i.e. there is a correlation. The advantage of random forests over bagging models is that the random forests make a tweak in the working algorithm of the bagging model to decrease the correlation in trees. The idea is to introduce more randomness while creating trees which will help in reducing correlation.

Let’s understand how the algorithm works for a random forest model:

1. Just like in bagging, different samples are collected from the training dataset using bootstrapping.
2. On each sample, we train our tree model and we allow the trees to grow with high depths. Now, the difference within the random forest is how the trees are formed. In bootstrapping we allow all the sample data to be used for splitting the nodes but not with random forests. When building a decision tree, each time a split is to happen, a random sample of ‘m’ predictors are chosen from the total ‘p’ predictors. Only those ‘m’ predictors are allowed to be used for the split.

Why is that?

Suppose in those ‘p’ predictors, 1 predictor is very strong. Now each sample this predictor will remain the strongest. So, whenever trees will be built for these sampled data, this predictor will be chosen by all the trees for splitting and thus will result in a similar kind of tree formation for each bootstrap model. This introduces correlation in the dataset and averaging correlated dataset results do not lead to low variance. That’s why in the random forest the choice for selecting node for split is limited and it introduces randomness in the formation of the trees as well.

Most of the predictors are not allowed to be considered for the split.

Generally, the value of ‘m’ is taken as m ≈√p, where ‘p’ is the number of predictors in the sample.

When m=p, the random forest model becomes the bagging model.

\*This method is also referred to as “Feature Sampling”

1. Once the trees are formed, a prediction is made by the random forest by aggregating the predictions of all the models. For the regression model, the mean of all the predictions is the final prediction and for classification mode, the mode of all the predictions is considered the final predictions.

**Advantages and Disadvantages of Random Forest**

1. It can be used for both regression and classification problems.
2. Since the base model is a tree, handling of missing values is easy.
3. It gives very accurate results with very low variance.
4. Results of a random forest are very hard to interpret in comparison with decision trees.
5. High computational time than other respective models.

Random Forest should be used where accuracy is up utmost priority and interpretability is not very important. Also, the computational time is less expensive than the desired outcome.

## Hands-on

**Binary Classification**

**Problem Statement**

Among all the industries, the Banking domain has the largest use of analytics and machine learning algorithms. This data set would provide you with enough taste of working on data sets from the banking domain, what challenges are faced, what strategies are used, which variable influences the outcome etc.

The company wants to automate the loan eligibility process (real-time) based on customer detail provided while filling out the online application form. These details are Gender, Marital Status, Education, Number of Dependents, Income, Loan Amount, Credit History, and others.

Data:

**Step1**: Importing dependencies and data

First, we can load in the data as Pandas Data frame and look:

Text

Description automatically generated

Text

Description automatically generated

Let’s have a glance at the top 5 observations

Text

Description automatically generated with medium confidence

Table

Description automatically generated

We can also have a look at the statistical summary.

Text

Description automatically generated

Table

Description automatically generated

**Step2:** Checking Missing Values

Text

Description automatically generated

A picture containing text

Description automatically generated

We have a few missing values in the data. Let’s drop those missing values.

Text

Description automatically generated

Table

Description automatically generated

**Step3:** Label Encoding

We need to do label encoding for our categorical variable.



Let’s check “Dependents” variables

Graphical user interface, text, application

Description automatically generated

We can see the “3+” category. Since our machine learning algorithm doesn’t accept string let’s encode it to 4.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text

Description automatically generated

Now we are good to go with the labels data.

**Step4:** Data Visualization

Chart

Description automatically generated with low confidence

Chart, bar chart

Description automatically generated

Most of the loans were provided to “Graduate” compared to “Not Graduate”. The probability of graduates getting loan approval is high.

A picture containing graphical user interface

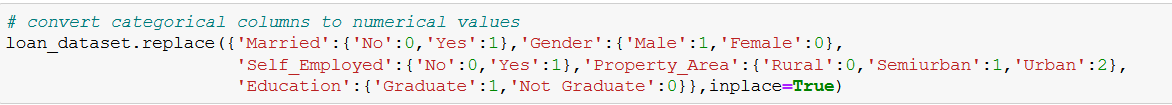
Description automatically generated

Chart, bar chart

Description automatically generated

We can also see a similar trend even in marital status. Married applicants are having the probability of loan approval.

We can see even these variables are strings that are needed to convert categorical to numerical values.



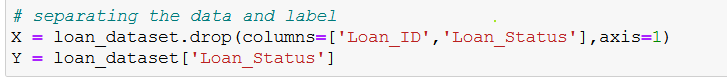
Let’s have a look at the data after converting the categorical data into numerical.



Graphical user interface, application

Description automatically generated

**Step5:** Separating the dependent and independent variables.



**Step6:** Splitting the dataset into train and test.



Let’s validate the shape of the dataset after the split.

Graphical user interface, text, application

Description automatically generated

**Step7:** Training the model

**Support Vector Machines**

Graphical user interface, text, application

Description automatically generated

**Model Evaluation**

Text

Description automatically generated

Text

Description automatically generated

**Logistic Regression**

Text

Description automatically generated

**Model Evaluation**

Text

Description automatically generated

Text

Description automatically generated

**KNN Classifier**

Graphical user interface, text, application

Description automatically generated

Model Evaluation

Text

Description automatically generated

Text

Description automatically generated

**Decision Tree**

Graphical user interface, text, application

Description automatically generated

**Model Evaluation**

Text

Description automatically generated

Text

Description automatically generated

**Random Forest**

Graphical user interface, text

Description automatically generated

**Model Evaluation**

Graphical user interface, text

Description automatically generated

Text

Description automatically generated

Let’s visually compare the performance of the models

Text

Description automatically generated

Chart, bar chart

Description automatically generated

We can see that Random Forest, SVC, and Logistic Regression have performed equally on the test data.

I am going with a random forest classifier since the model has performed well on the training dataset has well.

**Step7:** Checking Feature Importance

Text

Description automatically generated

Table

Description automatically generated

The top four variables are explaining nearly 75% of the variation in the model.

**Multi-Class Classification**

**Problem Statement**

The dataset is related to the fruits, given certain features we need to predict the type of the fruit.

**Step1:** Import the libraries and data

Text

Description automatically generated

Text

Description automatically generated with low confidence

Table

Description automatically generated

We have different features like mass, width, height, and color\_score of fruits. Our goal is to predict the fruit name based on the given independent features.

**Step2:** Statistical Summary

Table

Description automatically generated

We can see that the numerical values do not have the same scale. We will need to apply to scale to the test set that we computed for the training set.

**Step3:** Visual Analysis of the data

Chart, bar chart

Description automatically generated

The data is pretty balanced except for mandarin. We will just have to go with it.

Box plot for each numeric variable will give us a clearer idea of the distribution of the input variables:

Text

Description automatically generated with medium confidence

Chart, box and whisker chart

Description automatically generated

Let’s also see the histogram for this variable.

Graphical user interface, text

Description automatically generated



Chart, box and whisker chart

Description automatically generated

It looks like perhaps the color score has a near Gaussian distribution.

Let’s check the correlation using scatter plots.

A picture containing diagram

Description automatically generated

Chart

Description automatically generated

Some pairs of attributes are correlated (mass and width). This suggests a high correlation and a predictable relationship

**Step3:** Separating the dependent and independent variables.

A picture containing graphical user interface

Description automatically generated

**Step4:** Splitting the dataset into train and test

A picture containing diagram

Description automatically generated

**Step5:** Scaling train and test splits

Text

Description automatically generated

**Step6:** Model Fitting

**Decision Tree**

Text, letter

Description automatically generated

A screenshot of a computer

Description automatically generated with low confidence

We can see the model is overfitting on the training set. We can generalize the model by tweaking the parameter in the model.

Setting max decision tree depth to help avoid overfitting.

Text, letter

Description automatically generated

**K-Nearest Neighbors**

Text, letter

Description automatically generated

**Support Vector Machines**

Text, letter

Description automatically generated

**Random Forest**

Text, letter

Description automatically generated

The KNN algorithm was the most accurate model that we tried. Since we are working on a Multi-Class classification problem. It’s not appropriate to go with an accuracy score alone. We need to use some other metrics like precision, recall, and F1 score to make sure our model performance is better on the test set.

Let’s go ahead and extract the performance report using confusion matrix, precision, recall, and f1 score

Text, letter

Description automatically generated

**Confusion Matrix**

A picture containing icon

Description automatically generated

**Classification report**

Table

Description automatically generated

Finally, the classification report provides a breakdown of each class by precision, recall, f1-score, and support showing excellent results (However, the test set was small).

**SUMMARY**

In this chapter, we studied what is a classification problem, the types of classification problems, and different algorithms like Decision Tree, Random Forest, Support Vector Machines, and KNN Classifier. We have also seen the implementation of all the algorithms.

## Assessment

**Choose the appropriate option**

1. **Decision tree is the most powerful for \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**
   1. Classification
   2. Regression
   3. Both A and B
   4. None of these
2. **Decision tree is a flowchart like \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**
   1. Leaf structure
   2. Tree structure
   3. Steam
   4. None of the above
   5. All of the above
3. **Which of the following is true about Manhattan distance?**
4. It can be used for continuous variables
5. It can be used for categorical variables
6. It can be used for categorical as well as continuous
7. All of the above
8. **Which of the following distance measure do we use in the case of a categorical variable in KNN?**
9. Euclidean Distance
10. Manhattan Distance
11. Hamming Distance
12. None of the above
13. **Which of the following is/are true about bagging trees?**
14. **In bagging trees, individual trees are independent of each other**
15. **Bagging is the method for improving performance by aggregating the results of weak learners**
    1. 1
    2. 2
    3. 1 and 2
    4. None of these

**Fill in the spaces with appropriate answers**

1. Bootstrap and Aggregation, commonly known as \_\_\_\_\_\_\_\_\_\_\_\_\_
2. Random Forest has \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ as base learning models

Question 3-4

Euclidean Distance between the two data points A (1, 3) and B (2, 3).

Suppose, you have given the following data where x and y are the 2 input variables and class is the dependent variable.

Table

Description automatically generated

Below is a scatter plot that shows the above data in 2D space.

Calendar

Description automatically generated

1. Suppose, you want to predict the class of new data point x=1 and y=1 using Euclidian distance in 3-NN, in which class this data point belongs to \_\_\_\_\_\_\_\_\_\_\_\_?
2. SVM’s have linearly separable training data. No data points are allowed in the margin area. This type of linear classification is known as \_\_\_\_\_\_\_\_\_\_\_\_\_ margin classification.

**True or False**

1. Random Forest usually does not require pruning because it will not over-fit like a single decision tree.
   1. True
   2. False
2. The decision boundary is a line or a plane that separates the target variable into different classes that can be either linear or nonlinear. In the case of a logistic regression model, the decision boundary is a polynomial line.
   1. True
   2. False
3. To solve the problem of outliers, a sigmoid function is used in Logistic Regression.
   1. True
   2. False
4. A fully grown decision tree can lead the model to overfit.
   1. True
   2. False
5. ID3 is one of the algorithms used for deriving decision trees.
   1. True
   2. False

## Programming Assessment

Using the data in the below URL, Perform the following tasks

https://github.com/fenago/MLBook/blob/main/Chapter%206%20-%20Classification/Code/Dataset/Cars.txt

1. Import the data
2. Perform Data Cleaning
3. Perform EDA
4. Fit Decision Tree, KNN, SVC, and Random Forest Classification models.
5. Evaluate the models.

## Assessment Solutions

**Choose the appropriate options**

1. C
2. B
3. A
4. C
5. C

**Fill in the spaces with appropriate answers**

1. Bagging
2. Multiple decision trees
3. + Class
4. - Class
5. Hard Margin

**True or False**

1. True
2. False
3. True
4. True
5. True