**CHAPTER 1: INTRODUCTION TO MACHINE LEARNING**

**Theory**

This chapter is intended to provide a comprehensive introduction to machine learning. You will understand the concepts such as what is supervised learning, what is unsupervised learning, what is a classification problem, understanding the k-nearest neighbor’s algorithm, evaluation metrics for a classification model, model fitting, and hyperparameter tuning.

**What is Machine Learning?**

Machine learning is the science of teaching machines how to learn by themselves. now, you might be thinking - why on earth would we want missions to learn by themselves? Well - it has a lot of benefits.

Machines can do high-frequency repetitive tasks with high accuracy without getting bored.

For example - the task of mopping and cleaning the floor. when a human does the task - the quality of outcome would vary. We get exhausted/bored after a few hours of work and the chance of getting sick also impacts the outcome.

On the other hand, if we can teach machines to detect whether the floor needs cleaning and mopping and how much cleaning is required based on the condition of the floor and the type of floor - machines would perform the same job far better. They can go on to do that job without getting tired or sick!

This is what machine learning aims to do - Enable machines to learn on their own. To answer the questions like:

* Whether the floor needs cleaning and mopping?
* How long does the floor need to be cleaned?
* Whether the floor needs cleaning or not?
* For how long it needs to be cleaned, and so on.

**What is the definition of Machine Learning?**

Machine learning algorithms use statistics to find patterns in massive amounts of data. and data, here, encompasses a lot of things – numbers, words, images, and clicks. If it can be digitally stored, it can be fed into a machine learning algorithm.

Machine learning is the process that powers many of the services we use today - recommendation systems like those on Netflix, YouTube, and Spotify; search engines like Google and buy Baidu; social media feeds like Facebook and Twitter; voice assistance like Siri and Alexa. the list goes on.

In all these instances, each platform is collecting as much data about you as possible - what genres do you like watching, what links you are clicking, which statuses you are reacting to - and using machine learning to make a highly educated guess about what you might want next. Or, in the case of voice assistance, about which words match best with the funny sounds coming out of your mouth.

Frankly, this process is quite basic: find the pattern, apply the pattern. But it pretty much runs the world. That's a big part thanks to an invention in 1986, courtesy of Geoffrey Hinton, today is known as the father of deep learning.

**Review of Probability**



**Assigning Probabilities**

Classical method – ***A prior or Theoretical***

Probability can be determined before conducting any experiment.

Text

Description automatically generated with medium confidence

Experiment: Tossing of a fair dice

A picture containing icon

Description automatically generated

Outcome: Possible Result of experiment {1, 2, 3, 4, 5, 6}

Sample Space: S = {1, 2, 3, 4, 5, 6}

Event: The thing of our interest, for example: getting a number ‘4’ on dice

P (4) = 1/6 = 1.6667

Empirical Method – ***A posteriori or Frequentist***

Probability can be determined post conducting a thought experiment.

A picture containing text

Description automatically generated

Example 1: Tossing off a weighted die…well! Or even a fair die.

This is the most used method in statistical inference.

Example 2:

100 calls handled by an agent at a call center

Chart, scatter chart

Description automatically generated

Chart, bar chart

Description automatically generated

Next 100 calls handled by an agent at a call canter

Chart, scatter chart

Description automatically generated

Chart, bar chart

Description automatically generated

Average over the long run

Chart

Description automatically generated

Chart, bar chart

Description automatically generated

P (easy) = 0.7

Subjective Method: Based on feelings, insights, knowledge, etc. of a person

What is the probability of India winning the upcoming series against England?

**Probability Terminology**

Sample Space – Set of all possible outcomes, denoted S.

Example: After 2-coin tosses, the set of all possible outcomes are {HH, HT, TH, TT}

Event – A subset of the samples space.

An Event of interest might be – HH

**Probability – Rules**

Shape, rectangle

Description automatically generated Graphical user interface

Description automatically generated with low confidence A picture containing chart

Description automatically generated

P(S) = 1 0 <= P (A) <= 1 P (A or B) = P (A) + P (B)

Area of the rectangle denotes sample space, and since probability is associated with area, it cannot be negative.



If A and B are mutually exclusive

P (A and B) = 0

But the probability of A or B is the sum of the individual probabilities.

P (A or B) = P (A) + P (B)

Example: A card cannot be a King and Queen at the same time when it is picked from a deck of cards.

The Probability of a card being King is P (king) = 1/13

The Probability of a card being Queen is P (Queen) = 1/13

When we combine those two events: P (King or Queen) = (1/13) + (1/13) = 2/13

Diagram, schematic

Description automatically generated

P (A or B) = P (A) + P (B) – P (A and B)



Events A and B cannot prevent the occurrence of one another, so from here we can say that events A and B have something common in them.

Example: In the case of rolling a die, the event of getting an “even face” and the event of getting “less than 5” are not mutually exclusive and they are also known as compatible events.

Let ‘A’ is denoted as an event of getting an ‘even face’ and ‘B’ is denoted as an event of getting ‘less than 5”

The events of getting an even number (A) = {**2**, **4**, 6}

The events of getting less than 5 (B) = {1, **2**, 3, **4**}

Between events A and B, the common outcomes are 2 and 4

Therefore, events A and B are compatible events/mutually non-exclusive.

**Probability – Types**

Contingency table summarizing 2 variables, Loan Default and Age:

Table

Description automatically generated

Convert it into probabilities:

Table

Description automatically generated

Marginal Probability: Probability describing a single attribute.

Diagram

Description automatically generated

Example:

P (No) = 0.816

P (Old) = 0.008

Joint Probability: Probability describing a combination of attributes.

Diagram

Description automatically generated

Example:

P (Yes and Young) = 0.077

Union Probability: Probability describing a new set that contains all the elements that are in at least one of the two sets.

A picture containing diagram

Description automatically generated

P (Yes or Young) = P (Yes) + P (Young) – P (Yes and Young) = 0.184 + 0.302 – 0.077 = 0.409



The sample space is restricted to a single row or column. This makes the rest of the sample irrelevant.

A picture containing chart

Description automatically generated

Example:

What is the probability that a person will not default on the loan payment given he/she is middle age?

Table

Description automatically generated

P (No | Middle-Aged) = 0.586/0.690 = 0.85

Note that this is the ratio of Joint Probability to Marginal Probability, i.e.

Text

Description automatically generated with medium confidence

P (A/B) = P (A and B)/P (B) => P (A and B) = P (B) \* P (A/B)

Similarly

P (B/A) = P (A and B)/P (A) => P (A and B) = P (A) \* P (B/A)

Equating, we get

P (A/B) \* P (B) = P (A) \* P (B/A)

Text

Description automatically generated with low confidence

Now, given that the probability that someone defaults on a loan are 0.184, find the probability that an older person defaults on the loan. Older people make up only 0.8% of the clientele. P (Yes/Old) =?

P (Yes/Old) = (P (Yes) \* P (Old/Yes))/P (Old)

Table

Description automatically generated

P (Yes) = 8557/46687 = 0.184

P (Old/Yes) = P (Old and Yes) / P (Yes) = 120/8557 = 0.014

P (Old) = 379/46687 = 0.008

P (Yes/Old) = (0.184 \* 0.014) / 0.008 = 0.32

The Probability that an older person defaults on the loan are 32%

**Types of machine learning systems**

there are so many different types of machine learning systems that are useful to classify them in broad categories based on:

* whether or not they are trained with human supervision
* Whether or not they can learn incrementally on the fly
* whether they work by simply comparing new data points to known data points, or instead detect patterns in the training data and build a predictive model, much like scientists do

These criteria are not exclusive; you can combine them in any way you like. for example, a state of art spam filter may learn on the fly using a deep neural network model trained using examples of spam and ham; this makes it a Supervised learning system

Let's look at each of these criteria a bit more closely.

**Introduction to Supervised Learning**

In supervised learning, the computer is caught by example. it learns from past data and applies the learning to the present data to predict future events. In this case, both input and desired output data Provide help to the prediction of future events.

Supervised learning uses labeled data to train the model. but what does that mean in theory? Let's walk through some examples to start.

With supervised learning, the model is provided both inputs and corresponding outputs. Suppose we are training the model to identify I'm classified different kinds of fruits. In this example, you will provide several pictures of fruits as the input, along with their shape, size, color, and flavor profile. Next, you will provide the model with the names of each fruit as your output.

Eventually, the algorithm will pick up a pattern between the fruit’s characteristics (the inputs) and their names (the outputs). once this happens, the model can be provided with new input, and it would predict the output for you. this kind of supervised learning, called classification.

**Supervised machine learning categorization**

It is important to remember that all supervised learning algorithms are essentially complex algorithms, categorized as either classification or regression models.

1. **Classification models** - classification models are used for problems where the output variable can be categorized, such as “Yes” or “No”, or “Pass” or “Fail”. Classification models are used to predict the category of the data. real-life examples include spam detection, sentiment analysis, scorecard prediction of exams, etc.
2. **Regression models** – repression models are used for problems where the output variable is a real value such as unique number, dollars, salary, weight, or pressure, for example. It is most often used to predict numerical values based on previous data observations. some of the more familiar regression algorithms include linear regression, polynomial regression, ridge regression, and lasso regression.

Chart, scatter chart

Description automatically generated

There are some very practical applications of supervised learning algorithms in real life, including

* Text classification
* Spam detection
* Weather forecasting
* Predicting house prices based on the prevailing market price
* Stock price predictions, among others

**Unsupervised machine learning categorization**

Contrarily, unsupervised learning works by teaching the model to identify patterns on its own (hence unsupervised) from unlabelled data. This means that input is provided, but no output.

To understand how this works, let's continue with the fruit example given above. with unsupervised learning, you will provide the model with the input dataset (the pictures of the fruits and their characteristics), but you will not provide the output (the names of the fruits)

The model will use a suitable algorithm to train itself to divide the fruits into different groups according to the most similar features between them. this kind of unsupervised learning, called clustering, is the most common.

The idea is to explore the machines to the large volumes of wearing data and allow it to learn from data to provide insights that were previously unknown and to identify hidden patterns. As such, there aren't necessarily defined outcomes from unsupervised learning algorithms. Rather, it determines what is different are interesting from the given dataset.

The machine needs to be programmed to learn by itself. the computer needs to understand and provide insights from both structured add unstructured data. Here’s an accurate illustration of unsupervised learning

Diagram

Description automatically generated with medium confidence

1. Clustering is one of the most common unsupervised learning methods, the method of clustering involves organizing unlabelled data into similar groups called clusters. Thus, a cluster is a collection of similar data items. the primary goal here is to find similarities in the data points and group data points into a cluster.
2. Anomaly detection is the method of identifying rare items, events, or observations that differ significantly from most of the data. we generally look for anomalies that are outliers in data because they are suspicious. anomaly detection is often utilized in bank fraud and medical error detection.

Applications of unsupervised learning algorithms

Some practical applications of unsupervised learning algorithms include:

* Fraud detection
* Malware detection
* Identification of human errors during data entry
* Conducting accurate basket analysis, etc.

**Classification Predictive Modelling in Machine Learning**

Classification usually refers to any kind of problem where a specific type of class label is the result to be predicted from the given input field of data. Some types of clarification challenges are:

* Classifying emails as spam or ham
* Classify a given handwritten character to be either are known character or not
* Classify recent user behavior as turn or not

for any model, you will require a training dataset with many examples of inputs and outputs from which the model will train itself. the training data must include all the possible scenarios of the problem and must have sufficient data for each label for the model to be trained correctly. class labels are often returned as string values and hence need to be encoded into an integer like either representing 0 for “spam” and 1 for “ham”.

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

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

The rotation 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.

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 the purpose of 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. now we will create a dataset of our own and use binary classification on it. 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.

**Code**

Graphical user interface, text, application

Description automatically generated

**Output:**

Chart, scatter chart, bubble chart

Description automatically generated

The above example creates a data set of 5000 samples and divides him into input “X” and output “Y” elements. the distribution shows us that any one instance can either belongs to either class 0 or Class 1 they are approximately 50% in each.

The first 10 examples in the data set are shown with the input values which are numeric, 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 values. we can easily see two distinct clusters which we can discriminate.

Multi-class classification

these types of classification problems have no fixed two labels that can have any number of labels. some popular examples of multiclass classification are:

* Plant Species Classification
* Face Classification
* Optical Character Recognition

Here there is no notation of a normal and abnormal outcome, but the result will belong to one of many among a range of variables of known classes. There can also be a huge number of labels like predicting s picture as to how closely it might belong to one out of the tens of thousands of 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 category distribution unlike Bernoulli for binary classification. in a categorical distribution, an event can have multiple endpoints in our results and hence the model predicts the probability of input with respect to each of the output labels.

The most common algorithms which are used for multiclass classification are:

* K - Nearest Neighbours
* Naïve Bayes
* Decision Trees
* Gradient Boosting
* Random Forest

You can also use the algorithms for binary classification here on a basis of either one class versus all the other classes, also known as one versus rest, or one model for a pair of classes in the model which is also known as one versus one.

**One Vs Rest** – The main task here is to fit one model for each class which will be versus all the 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\_blob() function of the scikit learn module.

The following code demonstrates it.

**Code:**

Graphical user interface, text, application

Description automatically generated

Chart, scatter chart

Description automatically generated

Here you can see that there are more than two class types, and we can classify them separately into different types.

Multi-Label Classification

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 apple and a person, etc. the main difference is the ability to predict multiple labels and not just one.

You cannot use a binary classification model or a 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. the common algorithms used here are:

* Multi-label Random Forests
* Multi-label Decision trees
* Multi-label Gradient Boosting

One more approach is to use a separate classification algorithm for the label prediction for each type of class. we will use a library from scikit-learn to generate Our multi-label classification data set from scratch. the following code creates sand shows a working example of multi-label classification of 1000 samples and four types of classes.

**Code**

Graphical user interface, text, application

Description automatically generated

**Output:**

Text

Description automatically generated

Imbalanced Classification

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 data is of the normal class type and a minority of them belongs 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 are 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 to machine learning models. Especially for cases like:

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

Now we will look to develop a data set for the imperative classification problem. we will use the classification function of scikit learn to generate a full list synthetic and imbalanced binary classification dataset of 1000 samples

**Code:**

Text

Description automatically generated

**Output:**

Chart, scatter chart

Description automatically generated

Here we can see the distribution of the labels and we can see a severe imbalance of the data classes where 983 elements belong to one type and 17 belong to another type. you can see most of the type 0 or class 0 are expected. these types of datasets are more difficult to identify but they have a more general and practical use case.

Let’s start our machine learning journey with a very simple machine learning algorithm called KNN (K-Nearest Neighbours)

**K-Nearest Neighbours**

KNN is one of the most fundamental algorithms for classification and regression in the machine learning world.

But before proceeding with the algorithm, let's first discuss the life cycle of any machine learning model. This diagram explains the creation of a machine learning model from scratch and then taking the same model further with hyperparameter tuning to increase its accuracy, deciding the deployment strategies for that model, and once deployed, setting up the logging and monitoring frameworks to generate reports and dashboards based on the client requirements. A typical life cycle diagram for a machine learning model looks like this:

A picture containing timeline

Description automatically generated

**K-Nearest Neighbours (KNN)** is the 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 data and all the training points. It then selects the k points which are closest to the test data. once the points are 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 k selected training points.

Let's understand this with an illustration:

1. Given a training data set 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 of 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 and Probability (Red) = 1/3

Since the probability for the green class is highest 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 for 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 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

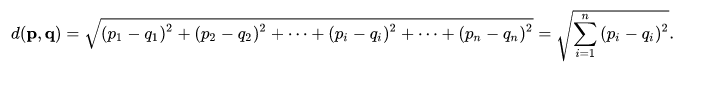
Description automatically generated

Image source: Wikipedia

Text

Description automatically generated

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



**Hamming distance**

According to Wikipedia, hamming distance is the 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 distance to be 0 if both the values are the same and the difference is 1 if both the values are different.

**Manhattan distance**

According to Wikipedia, the Manhattan distance, also known as the L1 norm, Taxicab norm, rectilinear distance, or City block distance. this distance represents some of the absolute distances between the opposite values in the 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.

**Lazy learners**

KNN algorithms are often termed, lazy learners. let's understand why that is? most of the algorithms like Bayesian classification, logistic regression, SVM, etc. are called eager learners.

These algorithms generalize over the training set before receiving the test data. i.e., they create a model based on the training data before receiving the test data and then do the prediction/classification on the text data. But this is not the case with the KNN algorithm. It doesn't create a generalized model for the training set but waits for the test data. once test data is provided then only it starts generalizing the training data to classify the test data. So, a lazy learner just stores the training data and wait for the test set. search algorithms work lists while training and more while classifying a given test dataset.

**Weighted Nearest Neighbours**

In weighted KNN, we assign weights to the k nearest neighbors. the weights are typically assigned based on distance. sometimes rest of the data points are assigned a weight of 0 also. the main intuition is that the points in neighbor should have more weight than the further points.

**Choosing the value of k**

The value of k effect’s the KNN classifier drastically. The flexibility of the model decreases with the increase of ‘k’. with the 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 a very low value of ‘k’ there is a chance of the 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, and test error rate:

Chart, line chart

Description automatically generated

Image source: “ISLR”

We can clearly see that the training error 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 a value of ‘k’ for which we get a minimum of both the errors and avoid overfitting as well as underfitting. views 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 very easy to understand.
* There is no need to create a model or do hyperparameter 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 the training samples.
* Since the model is not saved beforehand in this algorithm (Lazy learner), so every time one predicts our 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.
* Expensive in the testing phase.

**Different ways to perform KNN**

Above we studied the way the KNN classifier classifies the data by calculating the distance of test data from each of the observations and selecting ‘k’ values. This approach is also known as “Brute Force KNN”. This is computationally very expensive. So, there are other ways as well to perform KNN which are comparatively less expensive than the Brute force approach. The idea behind using other algorithms for the KNN classifier is to reduce the time during the test period by pre-processing the training data in such a way that the test data can be easily classified in the appropriate clusters.

let's discuss and understand the two most famous algorithms:

**K-Dimensional Tree (KD tree)**

k-d Tree is a hierarchical binary tree. when this algorithm is used for KNN classification, it rearranges the whole data set in a binary tree structure, so that pen test data is provided, it would give out the results by traversing through the tree, which takes less time than brute search.

Diagram

Description automatically generated

The data set is divided like a tree as shown in the above figure. say we have three-dimensional data i.e. (x, y, z) then the tree is formed with root node being on the dimensions, here we start with ‘x’. then on the next level, the split is done on basis of the second dimension, ‘y’ in our case. Similarly, the third level with the third dimension and so on. and in the case of ‘k’ dimensions, each split is made based on ‘k’ dimensions. let's understand how k-d trees are formed with an example:

Text, letter

Description automatically generated

A paper with writing on it

Description automatically generated with low confidence

A picture containing text, whiteboard

Description automatically generated

Diagram, schematic

Description automatically generated

Once the trees are formed, it is easy for the algorithm to search for the probable nearest neighbor just by traversing the tree. the main problem with k-d trees is that it gives probable nearest neighbors but can miss out on actual nearest neighbors.

**Ball Tree**

Like k-d trees, ball trees are also hierarchical data structures. these are very efficient especially in the case of high dimensions.

These are formed by the following steps:

* Two clusters are created initially.
* All the data points must belong to at least one of the clusters.
* One point cannot be in both clusters.
* The distance of the point is calculated from the centroid of each cluster. the point closer to the centroid goes into that cluster.
* Each cluster is then divided into sub-clusters again, and then the points are classified into each cluster based on distance from the centroid.
* This is how the clusters are kept being divided till a certain death.

Diagram, shape, venn diagram

Description automatically generated

Ball tree formation initially takes a lot of time but once the nested clusters are created finding nearest neighbors is easier.

**Mini Exercise**

Let’s see the practical implementation of all the above concepts in python.

We will be using the diabetes dataset for this exercise

Let’s import the dataset to start our analysis.



Graphical user interface, application

Description automatically generated

There are 8 independent variables and 1 dependent variable (Outcome). Here we want to predict the presence of “diabetes” given the health parameters like “Pregnancies”,” Glucose”, “BloodPressure” and so.

After doing the data exploration, data cleaning, and scaling the data we ended up with the below data.

Graphical user interface, text, application

Description automatically generated

Let’s split data into training and test sets to fit our model.



Initially, I am going to fit the KNN classifier model with default parameters and see how our model performs with default parameters.

A picture containing text

Description automatically generated

Text

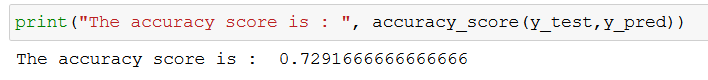
Description automatically generated with low confidence

Graphical user interface, text, application, Word

Description automatically generated

The accuracy score of the KNN model with default parameters on the training set is 81.94%

Let’s how the model performs on the test set.



The accuracy score on the test set is 72.91%, which is far different from our training set. It clearly states that our model is overfitting to our training set.

**Hyperparameter tuning**

We can perform the hyperparameter tuning using the “GridSearchCV” method from the sklearn package. GridSearchCV implements a “fit” and a “score” method. It also implements “score\_samples”,” predict\_proba”, “decision\_function”, “transform”, and inverse\_transform” if they are implemented in the estimator used. The parameters of the estimator used to apply these methods are optimized by cross-validation grid search over a parameter grid.

First, create a grid with parameters with which we want to tune our model.

A picture containing text

Description automatically generated

Created grid with three parameters such as ‘algorithm’, ‘leaf\_size’, and ‘n\_neighbors’ and passed a variety of values to check the best parameters out of the following parameters.





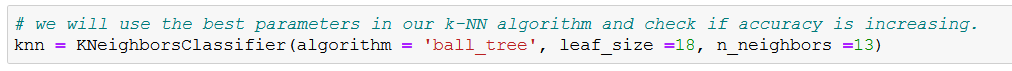
The above codes created a gridsearch object and fitted the model with hyperparameters. Let’s check the best parameters.

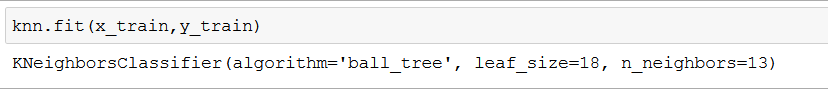
Graphical user interface, text, application

Description automatically generated

We can see that “ball\_tree” algorithm with 18 “leaf\_size” and 9 ‘n\_neighbors’ gave the best results for our data.

Now we can go ahead and build the model by passing these parameters.





Graphical user interface, text, application

Description automatically generated

Looks like accuracy for training has decreased, maybe our model was overfitting the data before. Let’s see how it performs on the test data.

Graphical user interface, text, application

Description automatically generated

Great, accuracy score has increased for our test data. So, indeed our model was overfitting before, Now, it looks better.

**Cross-Validation**

Suppose you train a model on a given dataset using any specific algorithm. you tried to find the accuracy of the trained model using the same training data and found the accuracy to be 95% or maybe even 100%. what does this mean? is your model ready for prediction? the answer is no. why? because your model has trained itself on the given data, i.e., it knows the data and it has generalized over it the reveal. but when you try and predict over a new set of data, it's most likely to give you very bad accuracy, because it has never seen the data before and thus it fails to generalize well over it. is the problem of overfitting. tracking search problem, cross-validation comes into the picture. cross-validation is a resampling technique with a basic idea of dividing the training dataset into two parts i.e., train and test. On one which is unseen for the modern, you make the prediction man check how well your model works on it. if the model works with good accuracy on your test data, it means that the model has not overfitted the training data and can be trusted with the prediction, whereas if it performs with bad accuracy then our model is not to be trusted and we need to tweak our algorithm.

let's see the different approaches for cross-validation:

* Holdout weather: It is the most basic of the CV techniques. it simply divides the dataset into two sets of training and test. the training dataset is used to train the modern and then test data is fitted in the trained model to make predictions. we checked the accuracy and assessed our model on that basis. this method is used as it is computationally less costly. but the evaluation based on the hold-out set can have a high variance because it depends heavily on which data points end up in the training set and which in test data. the evaluation will be different every time this division changes.
* K-fold Cross-Validation: To tackle the high variance of the holdout method, the k-fold method is used. the idea is simple, divide the whole data set into ‘k’ sets preferably of equal sizes. When the first set is selected as a test set and the rest ‘k-1’ sets are used to train the data. Error is calculated for this particular dataset. Then the steps are repeated i.e., the second set is selected as the test data, and the remaining ‘k-1’ sets are used as training data. Again, the error is calculated. Similarly, the process continues for ’k’ times. In the end, the CV error is given as the mean of the total errors calculated individually, mathematically given as:

A picture containing text, clock

Description automatically generated

The variance in error decreases with the increase in ‘k’. the disadvantage of ‘k-fold’ cv is that it is computationally expensive as the algorithm from scratch for ‘k’ times

Chart, diagram

Description automatically generated

Image source: Wikipedia

* Leave One Out Cross Validation (LOOCV): LOOCV is a special case of k-fold CV, where k becomes equal to n (number of observations). So instead of creating subsets, it selects a single observation as test data and the rest of the data as the training data. The error is calculated for these test observations. Now, the second observation is selected as test data, and the rest of the data is used as the training set. Again, the error is calculated for this particular test observation. this process continues ’n’ times and in the end. CV error is calculated as:

Diagram, schematic

Description automatically generated

Chart

Description automatically generated with low confidence

**Exercise:**

Let’s use k-fold cross-validation and check how well our model is generalizing over our dataset: We are randomly selecting our k to be 12 for the ‘k’ fold.

Graphical user interface, text, application, Word

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Let’s visualize the test accuracy with the value of k in k-fold

Text

Description automatically generated

Chart, line chart

Description automatically generated

The average train score is: 79.71%

The average test score is: 75.26%

Our cross-validation tells that on average our model has 75% accuracy on our test data. So, that’s how we can use cross-validation to compute how well our model is generalizing on our data.

Let’s use our skills which we have learned till now and solve the below use case

**Case Study**

Using a source of 10,000 bank records, we must create a model to demonstrate the ability to apply machine learning to predict the likelihood of customer churn.

This data set contains details of a bank’s customers, and the target variable is a binary variable reflecting the fact whether the customer left the bank (closed his account), or he continues to be a customer.

**Importing the dependencies**

Text

Description automatically generated

**Data importing and Exploring**

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Exploring the variables data types, and looking for any null values

Graphical user interface

Description automatically generated with low confidence

There are no “null” values in any of the variables

Let’s check for any duplicate values in the dataset.

Graphical user interface, text, application, email

Description automatically generated

Unused Features: To make the data frame easily readable we will drop features that do not have any importance in building a machine learning model.

* RowNumber
* CustomerId
* Surname

These features won’t contribute to modeling building, we can go ahead and drop them from our dataset.

Graphical user interface, text, application, website

Description automatically generated

Now we can start our analysis, let’s start with understanding the distributions of the numeric features by plotting the histogram grid

Text

Description automatically generated

Graphical user interface, chart, bar chart

Description automatically generated

Let’s also look for a summary statistics for the numeric features

A picture containing text

Description automatically generated

Table

Description automatically generated with low confidence

From the summary statistics and the histograms, we can conclude that all the features look good. We don’t see any extreme values for any feature.

Let’s analyze the distributions of categorical features

Text

Description automatically generated with low confidence

Graphical user interface, table

Description automatically generated

**Data Visualization**

We will start with the “gender” variable and see how data points are distributed among the gender.

Graphical user interface, text

Description automatically generated

Chart, bar chart

Description automatically generated

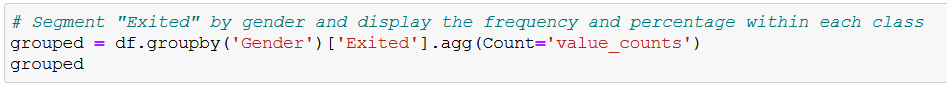
Graphical user interface, text, application, chat or text message

Description automatically generated

Chart, bar chart

Description automatically generated

Bivariate analysis with the target variable



Table

Description automatically generated with low confidence

Reorganizing the data frame for visualization

Graphical user interface, text, application, email

Description automatically generated

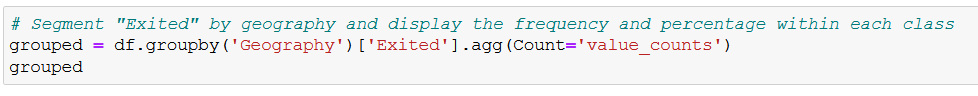
Text

Description automatically generated

Chart, bar chart

Description automatically generated

Churn segmentation based on geography



Graphical user interface

Description automatically generated with medium confidence

Reorganizing the data frame for visualization

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

Chart, bar chart, waterfall chart

Description automatically generated

Checking the correlation between numeric variables

Table

Description automatically generated

Generating a heatmap to visualize the correlation

Graphical user interface, text, application

Description automatically generated

Chart

Description automatically generated

Most of the variables were having a very weak negative correlation with the target variable “Exited”. “Age”, “Balance” and “Estimated Salary” are having very weak positive correlations.

Creating a pair plot to visualize the relationship between “Exited” and independent variables like “Age”, “IsActiveMember”, “NumofProducts”, “Balance”.

A picture containing text, shelf, colorful

Description automatically generated

From density plots, we can see that older customers and customers with more products more often leaving the bank.

Let’s create a violin plot segmenting “age”, “Balance”, “Number of Products” and “Active Membership” with “Exited”

Text

Description automatically generated

Text

Description automatically generated

Chart, radar chart

Description automatically generated

Chart, radar chart

Description automatically generated

Violin plots are confirming the earlier statement that older customers and customers with more products are more likely to leave the bank.

Finally, let’s check the distribution of our Target feature

Graphical user interface, application

Description automatically generated

Let’s define a small helper function that displays the count and percentage per class of the target function.

Text

Description automatically generated

Graphical user interface, text, table

Description automatically generated

Chart

Description automatically generated

We can see that our dataset is imbalanced. The majority class, “Stays” (0), has around 80% data points and the minority class, “Exits” (1), has around 20% data points.

To address this, in our machine learning algorithms we will use SMOTE (Synthetic Minority Over-Sampling Technique).

**Finalizing the Data frame**

Let’s check our final dataset before we proceed further with model building.

A picture containing graphical user interface

Description automatically generated

Text, table

Description automatically generated with medium confidence

Our dataset looks good, and it is ready to be saved.

Graphical user interface, text, application

Description automatically generated

Now let move to the model building part before that we need to load a few more libraries to proceed further.

We must create a dummy variable of the categorical features.

A picture containing text

Description automatically generated

Table

Description automatically generated

We have a filtered all the features with the ‘object’ data type and assigned them to ‘cat\_subset’.

Table

Description automatically generated

You can see we have used the “get\_dummies” method from the ‘pandas’ package and created dummy variables.

In a similar way, creating a numeric subset using the below code.

Table

Description automatically generated with medium confidence

Now, we have concatenated the numeric and categorical subset and assigned to data.

Graphical user interface, text, application

Description automatically generated

Now I’m going to separate the independent and dependent features and assign them to X and y respectively.

A picture containing text

Description automatically generated

Great!!! Let’s proceed by splitting the data into train and test sets. Before that, we should scale our data. Let’s use the ‘MinMaxScaler” for that.

Text

Description automatically generated

Table

Description automatically generated

**Train and Test Split**



Text, letter

Description automatically generated

Let’s fit the data into the KNN model and see how well it performs

Text

Description automatically generated with medium confidence

A picture containing text

Description automatically generated

Graphical user interface, text

Description automatically generated

Wow!!! Our model performed well on the training set. Let’s go ahead and check the performance of the test set.

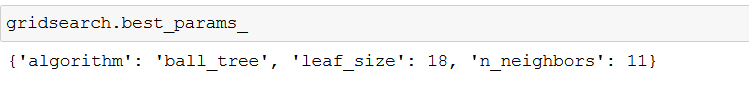
Graphical user interface

Description automatically generated with medium confidence

Great!! The model performed well even on the test set. Let’s try to increase the accuracy by using hyperparameter tuning.



Let’s see the best parameters according to gridsearch



We will use the best parameters in our KNN algorithm and check if our accuracy is increasing.



Graphical user interface, application, Word

Description automatically generated

Looks like accuracy for training data has decreased, let’s see how it performs on the test data.

Graphical user interface, text, application, Word

Description automatically generated

There is no change in the accuracy of our test data.

Accuracy is always not the best metric for the classification problem. Let’s look at metrics for classification models

**Evaluation of Classification Model**

In machine learning, once we have a result of the classification problem, how do we measure how accurate our classification is?

In a Classification problem, the credibility of the model is measured using the confusion matrix generated, i.e., how accurately the true positive and true negatives were predicted.

The different metrics used for this purpose are:

* Accuracy
* Recall
* Precision
* F1 score
* Specificity or True Negative Rate
* AUC (Area under the curve)
* RUC (Receiver Operator Characteristic)
* **Confusion Matrix**

A typical confusion matrix looks like the figure shown.

Table

Description automatically generated

Where the terms have the meaning:

* **True Positive (TP):** A result that was predicted as positive by the classification model and is positive.
* **True Negative (TN):** A result that was predicted as negative by the classification model and is negative.
* **False Positive (FP):** A result that was predicted as positive by the classification model but is negative.
* **False Negative (FN):** A result that was predicted as negative by the classification model but concerning the ability of the model is based on how many correct predictions the model makes.

**Accuracy**

The mathematical formula is:

A picture containing shape

Description automatically generated

It can be said that it’s defined as the total number of correct classifications divided by the total number of classifications.

**Recall or Sensitivity**

The mathematical formula is:

Text

Description automatically generated with medium confidence

As the name suggests, it is a measure of from the total number of positive results how many positives were correctly predicted by the model.

It shows how relative the model is, in terms of positive results only.

Let’s suppose in the previous model, the model made 50 correct predictions (TP) but failed to identify 200 cancer patients (FN). Recall in that case will be:

**= 0.2**

The model was able to recall only 20% of the cancer patients

**Precision**

Precision is a measure of amongst all the positive predictions, how many of them were actually positive. Mathematically,



Let’s suppose in the previous example,

The model identified 50 people as cancer patients (TP) but also raised a false alarm for 100 (FP).

**= 0.33**

The model only has a precision of 33%

**But we have a problem!!**

As evident from the previous example, the model had a very high Accuracy but performed poorly in terms of Precision and Recall. So, necessarily *Accuracy* is not the metric to use for evaluating the model in this case.

Imagine a scenario, where the requirement was that the model recalled all the defaulters who did not pay back the loan. Suppose there were 10 such defaulters and to recall those 10 defaulters, and the model gave you 20 results out of which only the 10 are the actual defaulters. Now, the recall of the model is 100%, but the precision goes down to 50%.

**Trade-off?**

Chart, histogram

Description automatically generated

As observed from the graph, with an increase in the Recall, there is a drop in Precision of the model.

So, the question is - what to go for? Precision or Recall?

Well, the answer is: it depends on the business requirement.

For example, if you are predicting cancer, you need a 100 % recall. But suppose you are predicting whether a person is innocent or not, you need 100% precision.

Can we maximize both at the same time? No

So, there is a need for a better metric then?

Yes. And it’s called an *F1 Score*

**F1 Score**

From the previous examples, it is clear that we need a metric that considers both Precision and Recall for evaluating a model. One such metric is the F1 score.

F1 score is defined as the harmonic mean of Precision and Recall.

The mathematical formula is:

**F1 Score** =

**Specificity or True Negative Rate**

This represents how specific is the model while predicting the True Negatives. Mathematically,

**Specificity** =

It can be said that it quantifies the total number of negatives predicted by the model concerning the total number of actual negative or non-favorable outcomes.

Similarly, False Positive rate can be defined as (1 – Specificity) or

**ROC (Receiver Operator Characteristic)**

We know that the classification algorithms work on the concept of probability of occurrence of the possible outcomes. A probability value lies between 0 and 1. Zero means that there is no probability of occurrence, and one means that the occurrence is certain.

But while working with real-time data, it has been observed that we seldom get a perfect 0 or 1 value. Instead of that, we get different decimal values lying between 0 and 1. Now the question is if we are not getting binary probability values determining the class in our classification problem?

There comes the concept of Threshold. A threshold is set, any probability value below the threshold is a negative outcome, and anything more than the threshold is a favorable or positive outcome. For Example, if the threshold is 0.5, any probability value below 0.5 means a negative or an unfavorable outcome and any value above 0.5 indicates a positive or favorable outcome.

Now, the question is what should be an ideal threshold?

The following diagram shows a typical logistic regression curve.

Chart, line chart

Description automatically generated

* The horizontal lines represent the various values of thresholds ranging from 0 to 1.
* Let’s suppose our classification problem was to identify the obese people from the given data.
* The green markers represent obese people, and the red markers represent non-obese people.
* Our confusion matrix will depend on the value of the threshold chosen by us.
* For Example, if 0.25 is the threshold then
* TP (Actually obese) = 3
* TN (Not obese) = 2
* FP (Not obese but predicted obese) = 2 (The two red squares above the 0.25 line)
* FN (Obese but predicted as not obese) = 1 (The Green circle below 0.25 line)

A typical ROC curve looks like the following figure.

Chart, line chart

Description automatically generated

* Mathematically, it represents the various confusion matrices for various thresholds. Each black dot is one confusion matrix.
* The green dotted line represents the scenario when the true positive rate equals the false positive rate.
* As evident from the curve, as we move from the rightmost dot towards the left, after a certain threshold, the false positive rate decreases.
* After some time, the false positive rate becomes zero.
* The point encircled in green is the best point as it predicts all the values correctly and keeps the false positive as a minimum.
* But that is not a rule of thumb. Based on the requirement, we need to select the point of a threshold.
* The ROC curve answers our question of which threshold to choose.

**But we have confusion!!**

Let’s suppose that we used different classification algorithms, and different ROCs for the corresponding algorithms have been plotted. The question is: which algorithm to choose now? The answer is to calculate the area under each ROC curve.

**Area Under Curve (AUC)**

Chart, line chart

Description automatically generated

* It helps us to choose the best model amongst the models for which we have plotted the ROC curves
* The best model is the one that encompasses the maximum area under it.
* In the adjacent diagram, amongst the two curves, the model that resulted in the red one should be chosen as it has the blue one.

Let’s check the performance of the model using the confusion matrix, precision, recall, and f1 score.

A picture containing text

Description automatically generated

Text

Description automatically generated with medium confidence

Table

Description automatically generated

The precision, recall and f1score are not up to the mark. The main reason for the score is less is the class imbalance problem.

Graphical user interface, text, application

Description automatically generated

Almost 80% of our data belongs to one class. Our model is more biased to class “0”. The solution for this could be SMOTE (Synthetic Minority Oversampling Technique).

Let’s now use k-fold cross-validation and check how well our model is generalizing over our dataset: we are randomly selecting our k to be 10 for the ‘k’ fold.

Graphical user interface, text

Description automatically generated with medium confidence

Text

Description automatically generated

Scatter chart

Description automatically generated

Let’s plot the test accuracy with the value of k in k-fold.

Chart, line chart

Description automatically generated

The average train score is: 85.34%

The average test score is: 81.95%

Our cross-validation tells that on average our model has 81% accuracy on our test data. So, that’s how we can use cross-validation to compute how well our model is generalizing on our data.

**SUMMARY**

Throughout the chapter, we have walked through a complete end-to-end machine learning use case. We started with an introduction to machine learning, probability, supervised and unsupervised learning then moved on to the use case. We imported the data, done with data cleaning, moved on to the model building, and finally looked at how to generalize the model. As a reminder, the general structure of the machine learning project is below:

* Data importing
* Data cleaning
* Exploratory data analysis
* Feature Engineering and selection
* Building the machine learning model
* Performing hyperparameter tuning
* Evaluate the model on the test set

While the exact steps vary by project, and machine learning is often an iterative rather than linear process, this chapter should serve you well as you tackle future machine learning chapters.

**Program Assignment**

Using the dataset used in the case study. Perform the following tasks.

1. Data Cleaning
2. Experiment with different values of k
3. Validate the model using LOOCV
4. Evaluate the model performance

**Assessment**

**Choose the appropriate option**

1. In what type of learning labeled training data is used?
   1. Unsupervised Learning
   2. Supervised Learning
   3. Reinforcement Learning
   4. None of the above
2. What characterizes unlabelled examples in machine learning?
   1. There is no prior knowledge
   2. There is no confusing knowledge
   3. There is prior knowledge
   4. There is plenty of confusing knowledge
3. Which of the following is the best machine learning method?
   1. Scalable
   2. Accuracy
   3. Fast
   4. All of the above
4. Data used to build a data mining model.
   1. Training data
   2. Validation data
   3. Test data
   4. Hidden data
5. The problem of finding hidden structures in unlabelled data is called.
   1. Supervised learning
   2. Unsupervised learning
   3. Reinforcement learning
   4. None of the above

**Fill in the spaces with appropriate answers**

1. When model fits very well to the training data and not perform well on test data such kind of issue is called as \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. What is “K” in the KNN Algorithm?
3. KNN Algorithm is known as \_\_\_\_\_\_\_\_\_\_\_\_ Learner.
4. Choosing the right value of K is done through a process known as \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.
5. If K value is too large, then our model is \_\_\_\_\_\_\_\_\_\_\_\_\_.

**True or False**

1. Accuracy is the best evaluation metric for classification problems.
   1. True
   2. False
2. KNN can be used to solve both classification and regression problem statements.
   1. True
   2. False
3. KNN is a Parametric algorithm
   1. Ture
   2. False
4. In K-fold cross-validation, the entire data is used for training.
   1. True
   2. False
5. In LOOCV, all of the data except one record is used for training and one record is used for testing.
   1. True
   2. False

**Assessment Solutions**

**Choose the appropriate option**

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

**Fill in the spaces with appropriate answers**

1. Overfitting
2. Number of Nearest Neighbours
3. Lazy
4. Hyperparameter Tuning
5. Underfitting

**True or False**

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