Data Science with Python

TriveraTech®
TECHNOLOGY TRAINING

Table of Contents

1. Introduction to Scikit-Learn: 3
2. Unsupervised Learning – Real-Life Applications: 62
3. Supervised Learning – Key Steps: 111
4. Supervised Learning Algorithms: Predicting Annual Income: 148
5. Supervised Learning – Key Steps: 187
6. Building Your Own Program: 227

1. Introduction to Scikit-Learn

Introduction to Scikit-Learn

Overview

- This lesson introduces the two main topics of this course: machine learning and scikit-learn.
- By reading this course, you will learn about the concept and application of machine learning.
- You will also learn about the importance of data in machine learning, as well as the key aspects of data preprocessing to solve a variety of data problems.

Introduction

- Machine learning (ML), without a doubt, is one of the most relevant technologies nowadays as it aims to convert information (data) into knowledge that can be used to make informed decisions.
- In this lesson, you will learn about the different applications of ML in today's world, as well as the role that data plays.
- This will be the starting point for introducing different data problems throughout this course that you will be able to solve using scikit-learn.

Introduction to Machine Learning

- Machine learning (ML) is a subset of Artificial Intelligence (AI) that consists of a wide variety of algorithms capable of learning from the data that is being fed to them, without being specifically programmed for a task.
- This ability to learn from data allows the algorithms to create models that are capable of solving complex data problems by finding patterns in historical data and improving them as new data is fed to the models.

Applications of ML

Some of the popular tasks that can be solved using ML algorithms are price/demand predictions, product/service recommendation, and data filtering, among others. The following is a list of real-life examples of such tasks:

- On-demand price prediction
- Recommendations in entertainment
- Email filtering

Choosing the Right ML Algorithm

- When it comes to developing ML solutions, it is important to highlight that, more often than not, there is no one solution for a data problem, much like there is no algorithm that fits all data problems.

- According to this and considering that there is a large quantity of algorithms in the field of ML, choosing the right one for a certain data problem is often the turning point that separates outstanding models from mediocre ones.

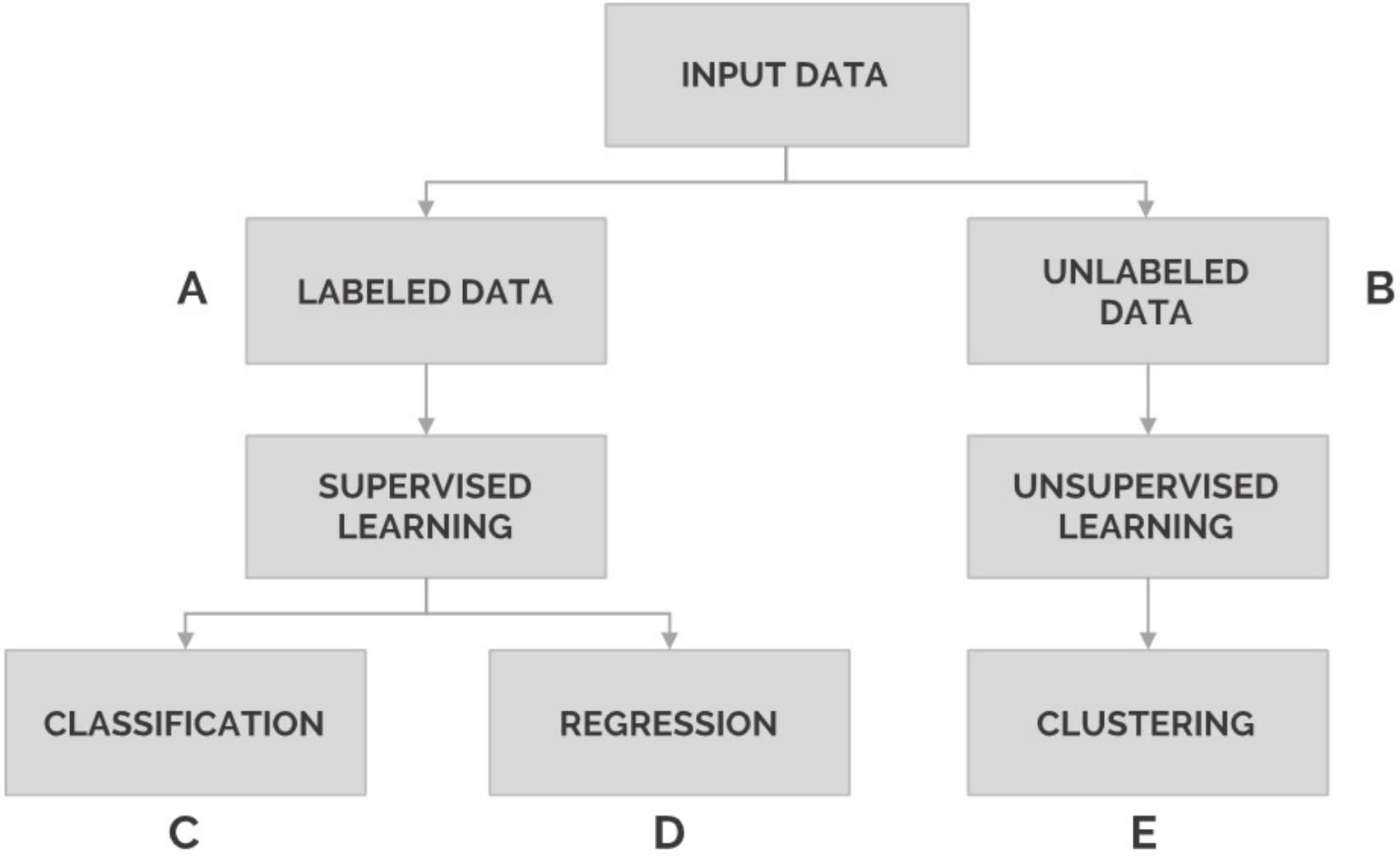
Choosing the Right ML Algorithm

- **Understand your data:** Considering that data is the key to being able to develop any ML solutions, the first step should always be to understand it in order to be able to filter out any algorithm that is unable to process such data.
- For instance, considering the quantity of features and observations in your dataset, it is possible to determine whether an algorithm capable of producing outstanding results with a small dataset is required.

Choosing the Right ML Algorithm

- **Categorize the data problem:** As per the following diagram, in this step, you should analyze your input data to determine if it contains a target feature (a feature whose values you want to be modeled and predicted) or not.

- Datasets with a target feature are also known as labeled data and are solved using supervised learning (A) algorithms.


INPUT DATA
A
LABELED DATA
UNLABELED DATA
B
SUPERVISED LEARNING
UNSUPERVISED LEARNING
CLASSIFICATION
REGRESSION
CLUSTERING
C
D
E

Choosing the Right ML Algorithm

- **Choose a set of algorithms:** Once the preceding steps have been performed, it is possible to filter out the algorithms that perform well over the input data and that are able to arrive at the desired outcome.

- Depending on your resources and time limitations, you should choose from this list of apt algorithms the ones that you want to test out over your data problem, considering that it is always a good practice to try more than one algorithm.

Scikit-Learn

- Created in 2007 by David Cournapeau as part of a Google Summer of Code project, scikit-learn is an open source Python library made to facilitate the process of building models based on built-in ML and statistical algorithms, without the need for hardcoding.
- The main reasons for its popular use are its complete documentation, its easy-to-use API, and the many collaborators who work every day to improve the library.

Scikit-Learn

- To carry out cross-validation and performance metrics analysis to understand the results that have been obtained from the model, and thereby improve its performance
- To obtain sample datasets to test algorithms on them
- To perform feature extraction to extract features from images or text data

Scikit-Learn

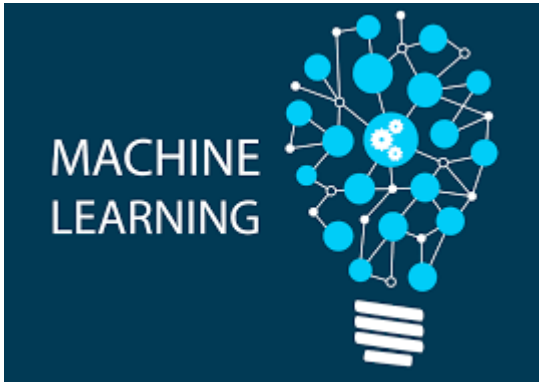
Some of the leading companies that are using scikit-learn are as follows:

- Spotify
- Booking.com
- Evernote
- Change.org:

Advantages of Scikit-Learn

The following is a list of the main advantages of using scikit-learn for ML purposes:

- Ease of use
- Uniformity
- Documentation/tutorials
- Reliability and collaborations
- Coverage



Disadvantages of Scikit-Learn

The following is a list of the main disadvantages of using scikit-learn for ML purposes:

- Inflexibility
- Not good for deep learning

Other Frameworks

Other popular ML frameworks are as follows:

- TensorFlow
- PyTorch
- Keras

Data Representation

- The main objective of ML is to build models by interpreting data.
- To do so, it is highly important to feed the data in a way that is readable by the computer.
- To feed data into a scikit-learn model, it must be represented as a table or matrix of the required dimensions, which we will discuss in the following section.

Tables of Data

- Most tables that are fed into ML problems are two-dimensional, meaning that they contain rows and columns.

- Conventionally, each row represents an observation (an instance), whereas each column represents a characteristic (feature) of each observation.

Columns: Features ↓

Rows: Observations/Instances →

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5	3.6	1.4	0.2	setosa
5	5.4	3.9	1.7	0.4	setosa
6	4.6	3.4	1.4	0.3	setosa
7	5	3.4	1.5	0.2	setosa
8	4.4	2.9	1.4	0.2	setosa
9	4.9	3.1	1.5	0.1	setosa

Features and Target Matrices

- For many data problems, one of the features of your dataset will be used as a label.
- This means that out of all the other features, this one is the target that the model should generalize the data to.
- For example, in the preceding table, we might choose the species as the target feature, so we would like the model to find patterns based on the other features to determine whether a plant belongs to the setosa species.

Complete Exercise 1.01: Loading a Sample Dataset and Creating the Features and Target Matrices

Complete Activity 1.01: Selecting a Target Feature and Creating a Target Matrix

Data Preprocessing

- Data preprocessing is a very critical step for developing ML solutions as it helps make sure that the model is not trained on biased data.

- It has the capability to improve a model's performance, and it is often the reason why the same algorithm for the same data problem works better for a programmer that has done an outstanding job preprocessing the dataset.

Messy Data

- Data that is missing information or that contains outliers or noise is considered to be messy data.
- Failing to perform any preprocessing to transform the data can lead to poorly created models of the data, due to the introduction of bias and information loss.
- Some of the issues with data that should be avoided will be explained here.

ID	Feature 1	Feature 2	Feature 3	Feature 4	Feature 5	Feature 6	Feature 7	Feature 8
Instance 1		1	3	5	6	8	1	
Instance 2	4	5	2	6	7	2	1	3
Instance 3	2		7	5	9	8	1	
Instance 4	1	2	7	5	2	1	6	
Instance 5	5	8	4	4	6	7	8	5
Instance 6	4	5	9	1	3	4	6	
Instance 7	7	6	5		4	8	6	
Instance 8								
Instance 9	8	2	3	1	2	4	5	3
Instance 10	4	5	9	6	4	9	7	

INSTANCE MISSING VALUES

FEATURE MISSING VALUES

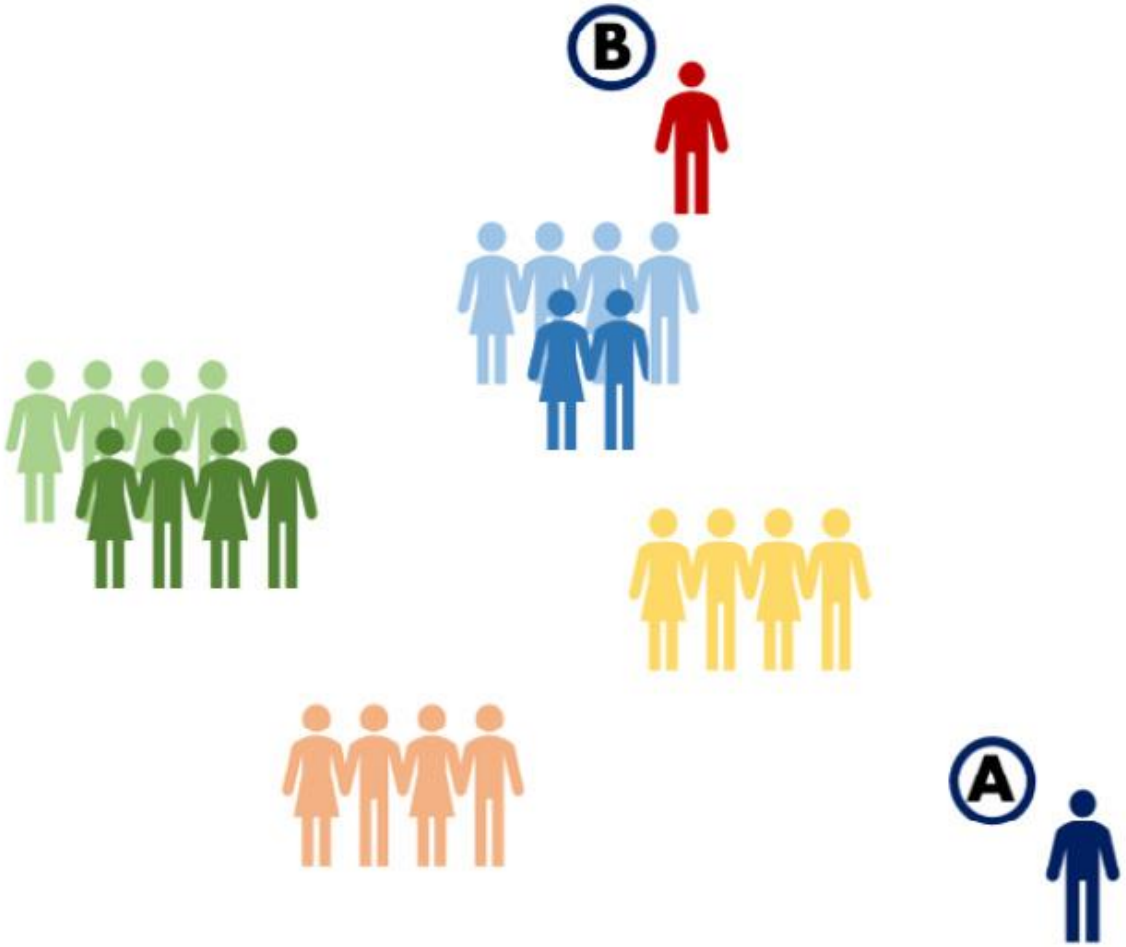
Data Preprocessing

When dealing with a feature with a high absence rate, it is recommended to either eliminate it or fill it with values. The most popular ways to replace the missing values are as follows:

- **Mean imputation:** Replacing missing values with the mean or median of the features' available values
- **Regression imputation:** Replacing missing values with the predicted values that have been obtained from a regression function

Outliers

- Outliers are values that are far from the mean.
- This means that if the values from a feature follow a Gaussian distribution, the outliers are located at the tails.
- Outliers can be global or local.
- The former group represents those values that are far from the entire set of values for a feature


B
A

Data Preprocessing

Once the outliers have been detected, there are three common ways to handle them:

- Delete the outlier
- Define a top
- Assign a new value

Complete Exercise 1.02: Dealing with Messy Data

Dealing with Categorical Features

- Categorical features are features that comprise discrete values typically belonging to a finite set of categories.

- Categorical data can be nominal or ordinal.

- Nominal refers to categories that do not follow a specific order, such as music genre or city names, whereas ordinal refers to categories with a sense of order, such as clothing sizes or level of education.

Feature Engineering

- Even though improvements in many ML algorithms have enabled the algorithms to understand categorical data types such as text, the process of transforming them into numeric values facilitates the training process of the model, which results in faster running times and better performance.

- This is mainly due to the elimination of semantics available in each category

Feature Engineering

Genre
Rock
Pop
Pop
Country
Rock
Pop

Genre
1
2
2
3
1
2

Complete Exercise 1.03: Applying Feature Engineering to Text Data

Rescaling Data

- Rescaling data is important because even though the data may be fed to a model using different scales for each feature, the lack of homogeneity can cause the algorithm to lose its ability to discover patterns from the data

- Due to the assumptions it has to make to understand it, thereby slowing down the training process and negatively affecting the model's performance.

Normalization:

- Data normalization in ML consists of rescaling the values of all features so that they lie in a range between 0 and 1 and have a maximum length of one.
- This serves the purpose of equating attributes of different scales.
- The following equation allows you to normalize the values of a feature:

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$

Standardization:

- This is a rescaling technique that transforms the data into a Gaussian distribution with a mean equal to 0 and a standard deviation equal to 1.
- One simple way of standardizing a feature is shown in the following equation:

$$z_i = \frac{x_i - mean(x)}{std(x)}$$

Complete Exercise 1.04: Normalizing and Standardizing Data

Complete Activity 1.02: Pre-processing an Entire Dataset

Scikit-Learn API

- The objective of the scikit-learn API is to provide an efficient and unified syntax to make ML accessible to non-ML experts, as well as to facilitate and popularize its use among several industries.

How Does It Work?

- Although it has many collaborators, the scikit-learn API was built and has been updated by considering a set of principles that prevent framework code proliferation, where different code performs similar functionalities.

- On the contrary, it promotes simple conventions and consistency.

Estimator

- This is considered to be the core of the entire API, as it is the interface in charge of fitting the models to the input data.

- It works by instantiating the model to be used and then applies a fit() method, which triggers the learning process so that it builds a model based on the data.

Scikit-Learn API

- The following is an example of a model being trained:

```
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(X_train, Y_train)
```

Scikit-Learn API

- Feature extraction, which involves transforming input data into numerical features that can be used for ML purposes.
- Feature selection, which selects the features in your data that contribute to the prediction output of the model.
- Dimensionality reduction, which takes high-dimensional data and converts it into a lower dimension.

Predictor

- As explained previously, the predictor takes the model created by the estimator and uses it to perform predictions on unseen data.

- In general terms, for supervised models, it feeds the model a new set of data, usually called X_test, to get a corresponding target or label based on the parameters that were learned while training the model.

Scikit-Learn API

Following the preceding example, the implementation of the predictor can be seen as follows:

```
Y_pred = model.predict(X_test)
```

Transformer

- As we saw previously, data is usually transformed before being fed to a model.
- Considering this, the API contains a transform() method that allows you to perform some preprocessing techniques.
- It can be used both as a starting point to transform the input data of the model (X_train), as well as further along to modify data that will be fed to the model for predictions.

Scikit-Learn API

- The following is an example of a transformer that normalizes the values of the training data:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
```

Scikit-Learn API

- The StandardScaler class standardizes the data that it receives as arguments.

- As you can see, after importing and instantiating the transformer (that is, StandardScaler), it needs to be fit to the data to then effectively transform it:

```
X_test = scaler.transform(X_test)
```

Supervised and Unsupervised Learning

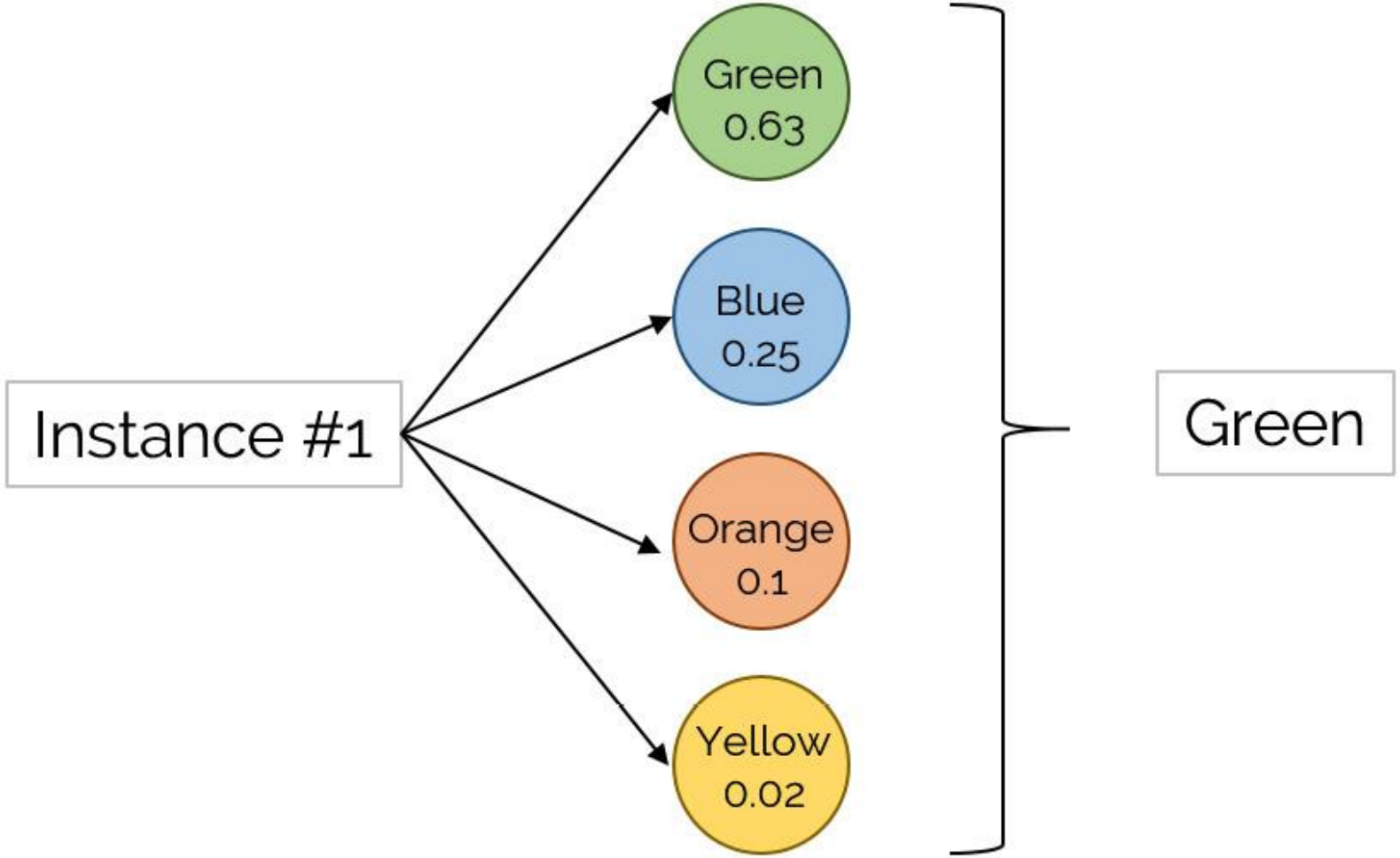
ML is divided into two main categories: supervised and unsupervised learning.

- **Supervised Learning**

- Supervised learning consists of understanding the relationship between a given set of features and a target value, also known as a label or class.
- For instance, it can be used for modeling the relationship between a person's demographic information and their ability to pay loans

Supervised and Unsupervised Learning

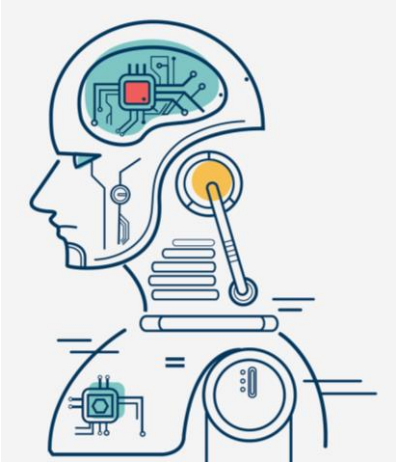
Age	Sex	Education level	Income level	Marital status	Previous loan paid
30	Female	College	$97,000	Single	Yes
53	Male	High school	$80,000	Single	No
26	Male	Masters	$157,000	Married	Yes
35	Female	None	$55,000	Married	No
44	Female	Undergraduate	$122,000	Single	Yes


Green
0.63
Blue
0.25
Orange
0.1
Yellow
0.02
Instance #1
Green

Supervised and Unsupervised Learning

Some of the most common classification algorithms are as follows:

- Decision trees
- Naïve Bayes classifier
- Artificial neural networks (ANNs)



Supervised and Unsupervised Learning

- The most popular algorithm for regression tasks is linear regression.
- It consists of only one independent feature (x) whose relationship with its dependent feature (y) is linear.
- Due to its simplicity, it is often overlooked, even though it performs very well for simple data problems.
- Other, more complex, regression algorithms include regression trees and support vector regression, as well as ANNs once again.

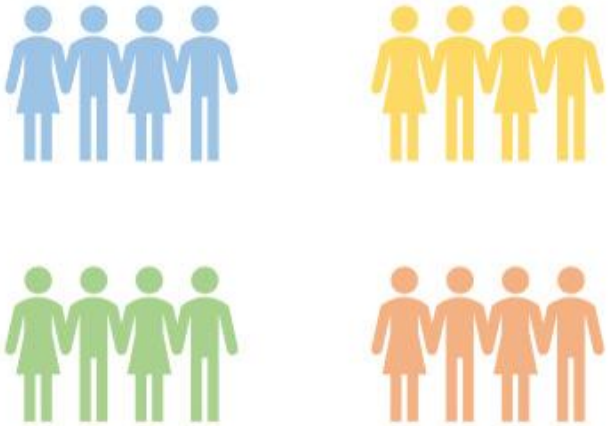
Unsupervised Learning

- Unsupervised learning consists of fitting the model to the data without any relationship with an output label, also known as unlabeled data.

- This means that algorithms in this category try to understand the data and find patterns in it.

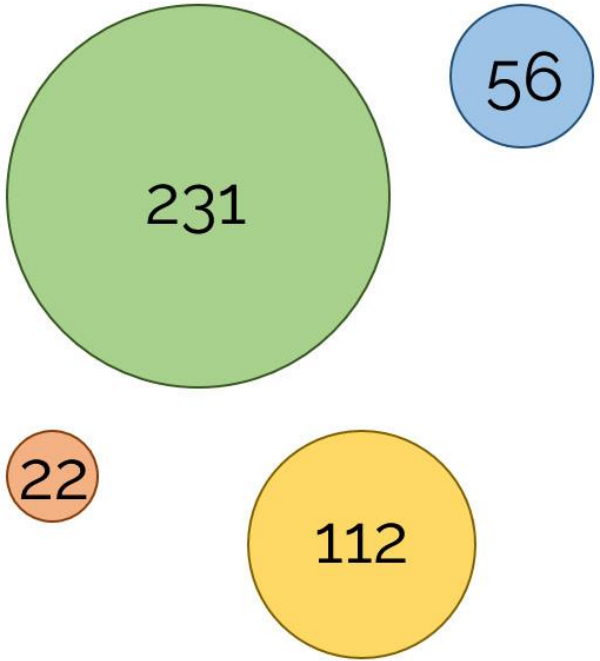
- For instance, unsupervised learning can be used to understand the profile of people belonging to a neighborhood

Age: 25 - 30
Marital status:
Married – no kids
Education level:
entrepreneurs



Age: 20 – 25
Marital status: single
Ocupation: currently
in college

Supervised and Unsupervised Learning



Supervised and Unsupervised Learning

Some of the most popular clustering algorithms are as follows:

- k-means
- Mean-shift clustering
- Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

Summary

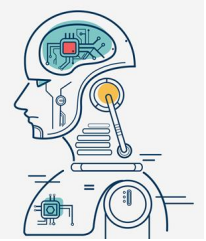
- ML consists of constructing models that are able to convert data into knowledge that can be used to make decisions, some of which are based on complicated mathematical concepts to understand data.

- Scikit-learn is an open source Python library that is meant to facilitate the process of applying these models to data problems, without much complex math knowledge required.

2. Unsupervised Learning – Real-Life Applications

TriveraTech®
TECHNOLOGY TRAINING

Unsupervised Learning – Real-Life Applications



Overview

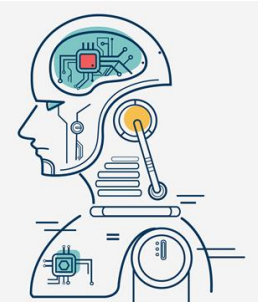
- This lesson explains the concept of clustering in machine learning.
- It explains three of the most common clustering algorithms, with a hands-on approximation to solve a real-life data problem.

Introduction

- In the previous lesson, we learned how to represent data in a tabular format, created features and target matrices, pre-processed data, and learned how to choose the algorithm that best suits the problem at hand.

- We also learned how the scikit-learn API works and why it is easy to use, as well as the difference between supervised and unsupervised learning.

Clustering



- Clustering is a type of unsupervised learning technique where the objective is to arrive at conclusions based on the patterns found within unlabeled input data.

- This technique is mainly used to segregate large data into subgroups in order to make informed decisions.

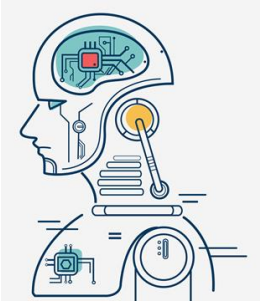
Clustering Types

- Clustering algorithms can classify data points using a methodology that is either hard or soft.
- The former designates data points completely to a cluster, whereas the latter method calculates the probability of each data point belonging to each cluster.
- For example, for a dataset containing customer's past orders that are divided into eight subgroups (clusters), hard clustering occurs when each customer is placed inside one of the eight clusters.

Clustering Types

Considering that clusters are created based on the similarity between data points, clustering algorithms can be further divided into several groups, depending on the set of rules used to measure similarity. Four of the most commonly known sets of rules are explained as follows:

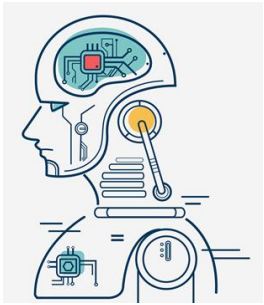
- Connectivity-based models
- Density-based models
- Distribution-based models
- Centroid-based models



Applications of Clustering

As with all machine learning algorithms, clustering has many applications in different fields, some of which are as follows:

- Search engine results
- Recommendation programs
- Image recognition
- Market segmentation



Exploring a Dataset – Wholesale Customers Dataset

- As part of the process of learning the behavior and applications of clustering algorithms
- The following sections of this lesson will focus on solving a real-life data problem using the Wholesale Customers dataset, which is available at the UC Irvine Machine Learning Repository.

Exploring a Dataset – Wholesale Customers Dataset

- Access the following link: http://archive.ics.uci.edu/ml/datasets/Wholesale+customers.
- Below the dataset's title, find the download section and click on Data Folder.
- Click on the Wholesale Customers data.csv file to trigger the download and save the file in the same path as that of your current Jupyter Notebook.

Understanding the Dataset

- Each step will be explained generically and will then be followed by an explanation of its application in the current case study (the Wholesale Customers dataset)

- First of all, it is crucial to understand the way in which data is presented by the person who's responsible for gathering and maintaining it.

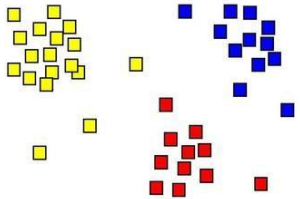
Understanding the Dataset

- Next, it is important to determine the purpose of the study, which is dependent on the data that's available.

- Even though this might seem like a redundant statement, many data problems become problematic because the researcher does not have a clear view of the purpose of the study, and hence the pre-processing methodology, the model, and the performance metrics are chosen incorrectly.

Understanding the Dataset

- Subsequently explore all the features that are available.



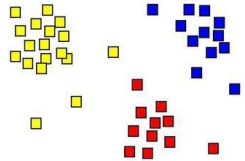
- This is mainly done for two reasons: first, to rule out features that are considered to be of low relevance based on the purpose of the study or that are considered to be redundant, and second, to understand the way the values are presented to determine some of the pre-processing techniques that may be needed.

Variable	Meaning	Type	Relevance
FRESH	Annual spending* on Fresh products	Continuous	These features help to identify the combination of categories that sell together based on the spending
MILK	Annual spending* on Dairy	Continuous	
GROCERY	Annual spending* on Grocery products	Continuous	
FROZEN	Annual spending* on Frozen products	Continuous	
DETERGENTS_PAPER	Annual spending* on Detergents and paper	Continuous	
DELICATESSEN	Delicatessen products	Continuous	
CHANNEL	Customer's sales channel	Nominal**	Both features help define users based on their purchasing habits by region and sales channel
REGION	Customer's region	Nominal**	

* Annual spending measured in monetary units.
** The author of the dataset converted the nominal features into their numeric representation.

Data Visualization

- Specific features that are causing trouble (for example, those that contain many missing or outlier values) and how to deal with them.

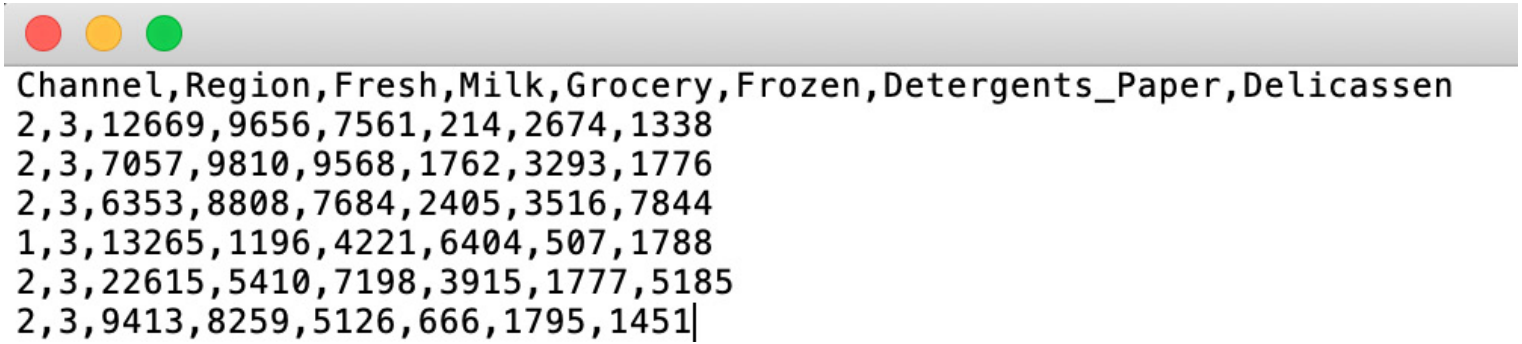


- The results from the model, such as the clusters that have been created or the number of predicted instances for each labeled category.

- The performance of the model, in order to see the behavior along different iterations.

Loading the Dataset Using pandas

- Most datasets found in online repositories or gathered by companies for data analysis are in Comma-Separated Values (CSV) files.
- CSV files are text files that display the data in the form of a table. Columns are separated by commas (,) and rows are on separate lines:

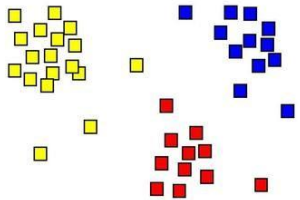


```
Channel,Region,Fresh,Milk,Grocery,Frozen,Detergents_Paper,Delicassen
2,3,12669,9656,7561,214,2674,1338
2,3,7057,9810,9568,1762,3293,1776
2,3,6353,8808,7684,2405,3516,7844
1,3,13265,1196,4221,6404,507,1788
2,3,22615,5410,7198,3915,1777,5185
2,3,9413,8259,5126,666,1795,1451
```

Data Visualization

The following code shows how to load a dataset using pandas:

```
import pandas as pd
file_path = "datasets/test.csv"
data = pd.read_csv(file_path)
print(type(data))
```



Data Visualization

- First of all, pandas is imported. Next, the path to the file is defined in order to input it into the read_csv() function.

- Finally, the type of the data variable is printed to verify that a Pandas DataFrame has been created.

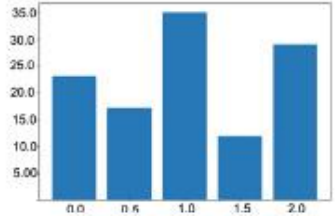
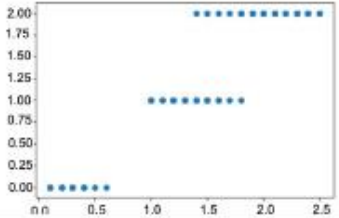
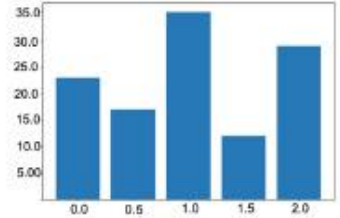
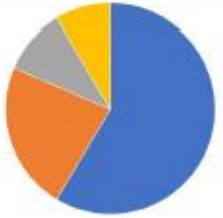
- The output is as follows:

<class 'pandas.core.frame.DataFrame'>

Visualization Tools

- There are different open source visualization libraries available, from which seaborn and matplotlib stand out.
- In the previous lesson, seaborn was used to load and display data; however, from this section onward, matplotlib will be used as our visualization library of choice.
- This is mainly because seaborn is built on top of matplotlib with the sole purpose of introducing a couple of plot types and to improve the format of the displays.

- Some of the most commonly used plot types are explained in the following table:

Plot Type	Definition	Function	Visual Representation
Histograms	Display the distribution of continuous data	plt.hist()	
Scatter plots	Display values for two variables using Cartesian coordinates	plt.scatter()	
Bar charts	Represent variables using bars, with heights proportional to the values they represent	plt.bar()	
Pie charts	A circular representation that displays proportions	plt.pie()	

Complete Exercise 2.01: Plotting a Histogram of One Feature from the Circles Dataset

Complete Activity 2.01: Using Data Visualization to Aid the Pre-processing Process

k-means Algorithm

- The k-means algorithm is used to model data without a labeled class.
- It involves dividing the data into K number of subgroups.
- The classification of data points into each group is done based on similarity, as explained previously (refer to the Clustering Types section), which, for this algorithm, is measured by the distance from the center (centroid) of the cluster.

Understanding the Algorithm

The k-means algorithm works through an iterative process that involves the following steps:

- Based on the number of clusters defined by the user, the centroids are generated either by setting initial estimates or by randomly choosing them from the data points. This step is known as initialization.
- All the data points are assigned to the nearest cluster in the data space by measuring their respective distances from the centroid, known as the assignment step.
- The objective is to minimize the squared Euclidean distance, which can be defined by the following formula:

$$\min \text{dist}(c,x)2$$

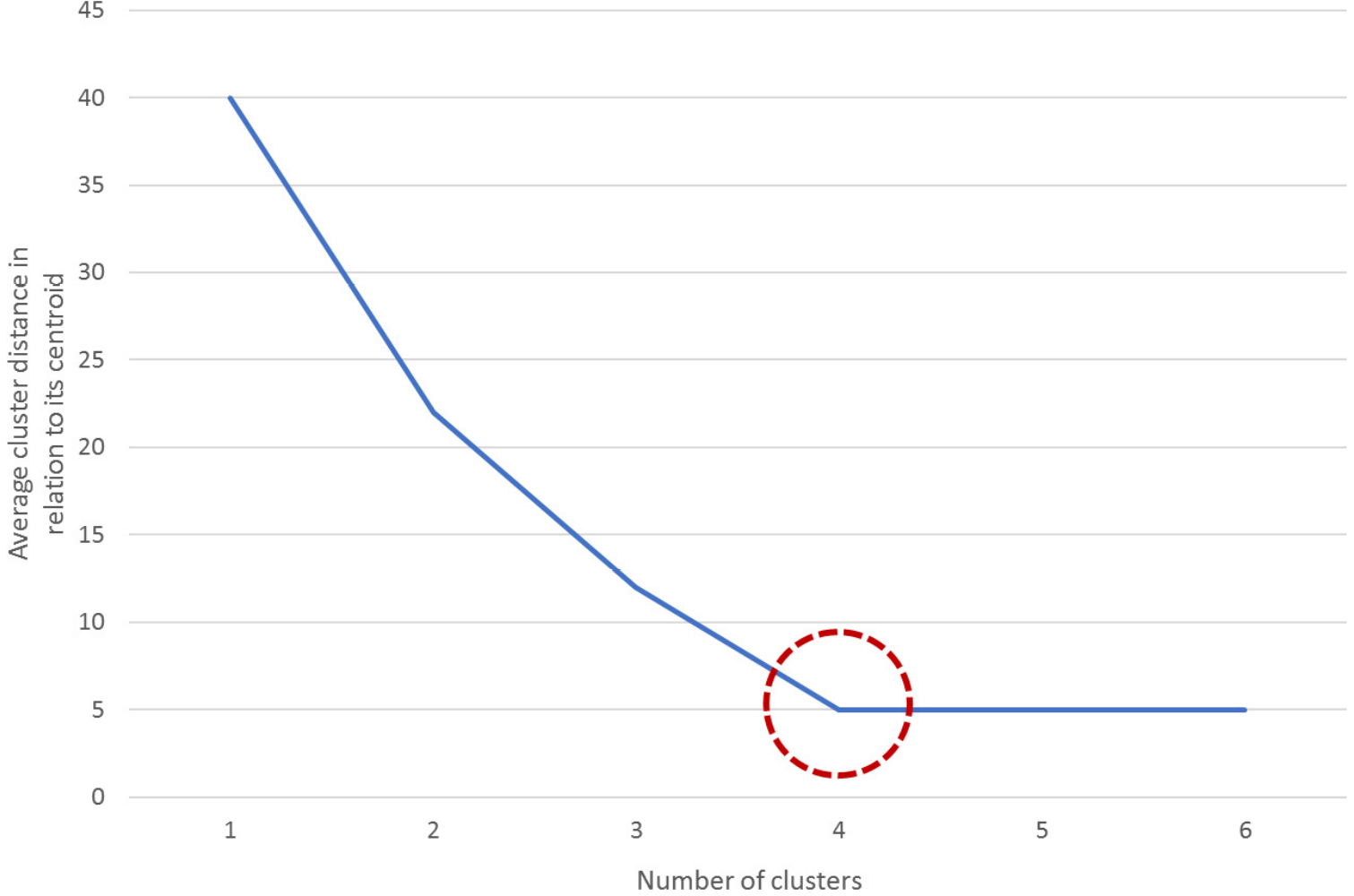
Initialization Methods

- **k-means++:** This is the default option. Centroids are chosen randomly from the set of data points, considering that centroids must be far away from one another.
- To achieve this, the method assigns a higher probability of being a centroid to those data points that are farther away from other centroids.
- **random:** This method chooses K observations randomly from the data points as the initial centroids.

Choosing the Number of Clusters

- As we discussed previously, the number of clusters that the data is to be divided into is set by the user; hence, it is important to choose the number of clusters appropriately.

- One of the metrics that's used to measure the performance of the k-means algorithm is the mean distance of the data points from the centroid of the cluster that they belong to.


45
40
35
30
25
20
15
10
5
0
Average cluster distance in relation to its centroid
1
2
3
4
5
6
Number of clusters

Complete Exercise 2.02: Importing and Training the k-means Algorithm over a Dataset

Complete Activity 2.02: Applying the k-means Algorithm to a Dataset

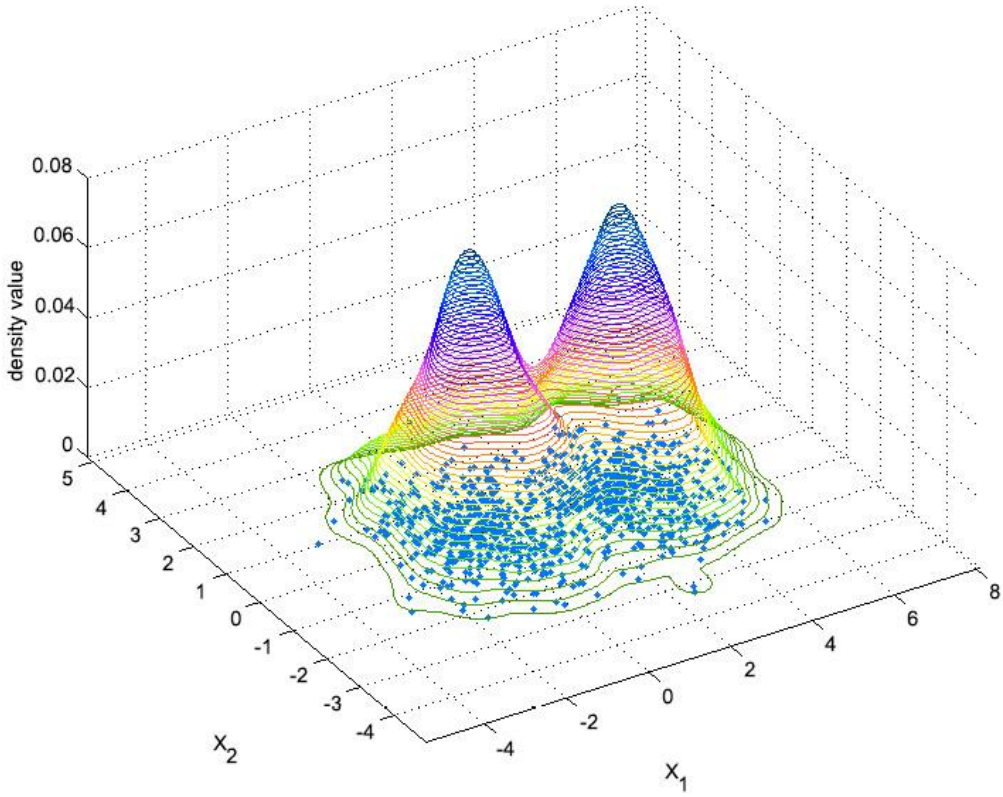
Mean-Shift Algorithm

- The mean-shift algorithm works by assigning each data point a cluster based on the density of the data points in the data space, also known as the mode in a distribution function.

- Contrary to the k-means algorithm, the mean-shift algorithm does not require you to specify the number of clusters as a parameter.

- The first step of the mean-shift algorithm is to represent the data points as a density distribution.

- To do so, the algorithm builds upon the idea of Kernel Density Estimation (KDE), which is a method that's used to estimate the distribution of a set of data:

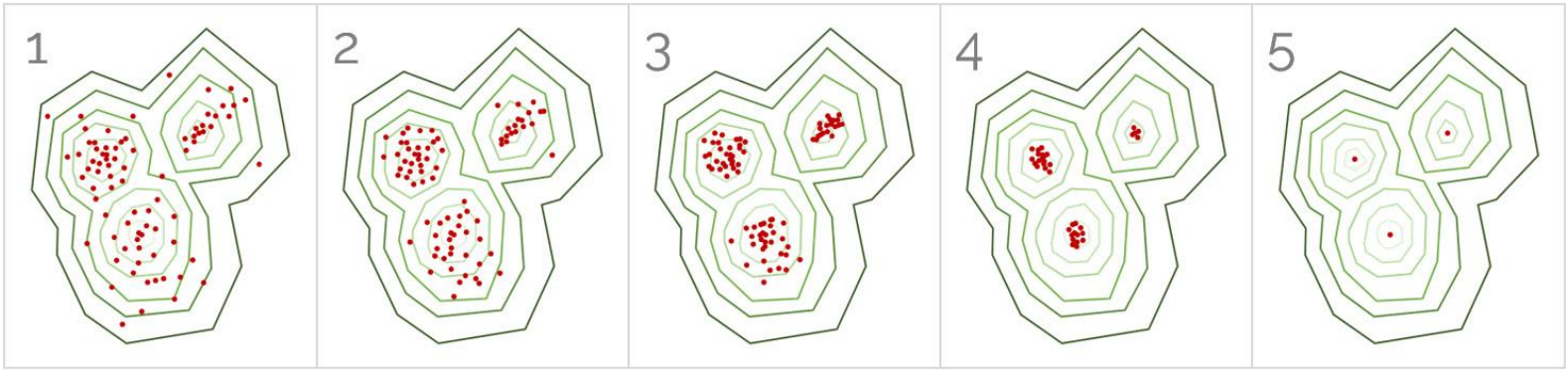


Mean-Shift Algorithm

The process of assigning data points to each cluster is as follows:

- A window of a specified size (bandwidth) is drawn around each data point.
- The mean of the data inside the window is computed.
- The center of the window is shifted to the mean.

- In the following diagram, the estimated distribution is represented by the lines, while the data points are the dots.
- In each of the boxes, the data points shift to the nearest peak.
- All the data points in a certain peak belong to that cluster:



Complete Exercise 2.03: Importing and Training the Mean-Shift Algorithm over a Dataset

Complete Activity 2.03: Applying the Mean-Shift Algorithm to a Dataset

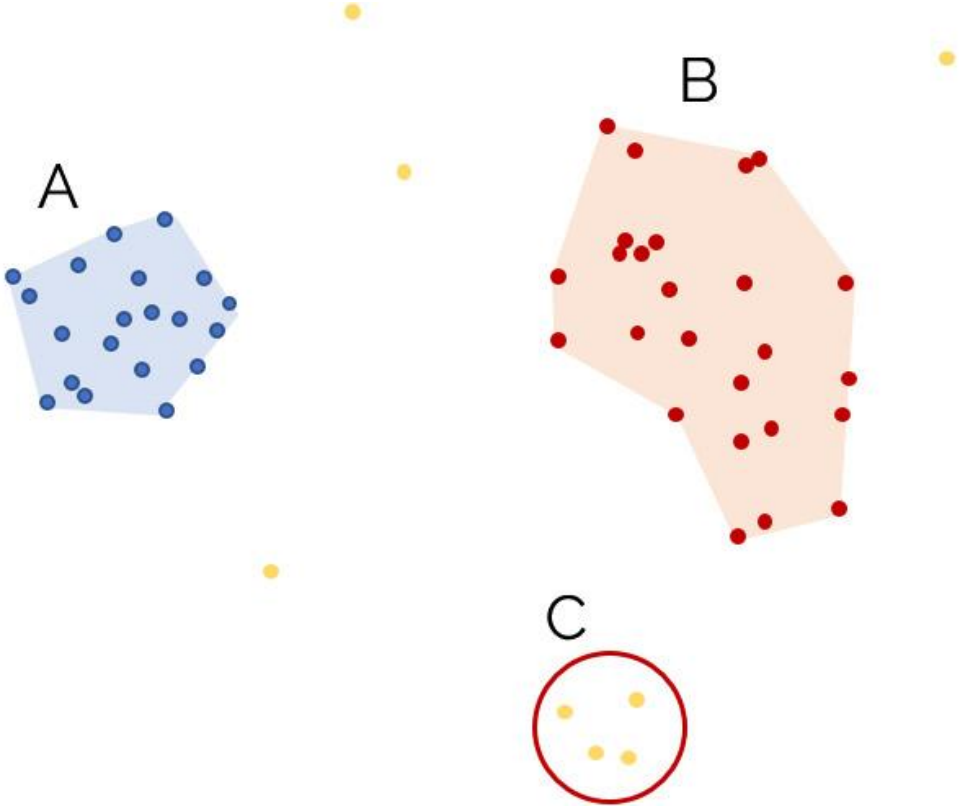
DBSCAN Algorithm

- The density-based spatial clustering of applications with noise (DBSCAN) algorithm groups together points that are close to each other (with many neighbors) and marks those points that are further away with no close neighbors as outliers.

- According to this, and as its name states, the algorithm classifies data points based on the density of all data points in the data space.

Understanding the Algorithm

- The DBSCAN algorithm requires two main parameters: epsilon and the minimum number of observations.
- Epsilon, also known as eps, is the maximum distance that defines the radius within which the algorithm searches for neighbors.
- The minimum number of observations, on the other hand, refers to the number of data points required to form a high-density area (min_samples)


A
B
C

DBSCAN Algorithm

According to this, each data point can be classified as follows:

- **A core point:** A point that has at least the minimum number of data points within its eps radius.
- **A border point:** A point that is within the eps radius of a core point, but does not have the required number of data points within its own radius.
- **A noise point:** All points that do not meet the preceding descriptions.

Complete Exercise 2.04: Importing and Training the DBSCAN Algorithm over a Dataset

Complete Activity 2.04: Applying the DBSCAN Algorithm to the Dataset

Evaluating the Performance of Clusters

- After applying a clustering algorithm, it is necessary to evaluate how well the algorithm has performed.
- This is especially important when it is difficult to visually evaluate the clusters; for example, when there are several features.
- Usually, with supervised algorithms, it is easy to evaluate their performance by simply comparing the prediction of each instance with its true value (class)

Available Metrics in Scikit-Learn

- Scikit-learn allows its users to use three different scores for evaluating the performance of unsupervised clustering algorithms.

- The main idea behind these scores is to measure how well-defined the cluster's edges are, instead of measuring the dispersion within a cluster.

- Hence, it is worth mentioning that the scores do not take into account the size of each cluster.

Evaluating the Performance of Clusters

The two most commonly used scores for measuring unsupervised clustering tasks are explained as follows:

- The Silhouette Coefficient Score calculates the mean distance between each point and all the other points of a cluster (a), as well as the mean distance between each point and all the other points of its nearest clusters (b).
- It relates both of them according to the following equation:

$$s = (b - a) / \max(a,b)$$

Evaluating the Performance of Clusters

- The Calinski–Harabasz Index was created to measure the relationship between the variance of each cluster and the variance of all clusters.
- More specifically, the variance of each cluster is the mean square error of each point with respect to the centroid of that cluster.
- On the other hand, the variance of all clusters refers to the overall inter-cluster variance.

Evaluating the Performance of Clusters

- **Homogeneity score:** This score is based on the premise that a clustering task is homogenous if all clusters only contain data points that belong to a single class label.
- The output from the score is a number between 0 and 1, with 1 being a perfectly homogeneous labeling.
- The score is part of scikit-learn's metrics module, and it receives the list of ground truth clusters and the list of predicted clusters as inputs, as follows:

```
from sklearn.metrics import homogeneity_score
score = homogeneity_score(true_labels, predicted_labels)
```

Evaluating the Performance of Clusters

- **Completeness score:** Opposite to the homogeneity score, a clustering task satisfies completeness if all data points that belong to a given class label belong to the same cluster.
- Again, the output measure is a number between 0 and 1, with 1 being the output for perfect completeness.
- This score is also part of scikit-learn's metrics modules, and it also receives the ground truth labels and the predicted ones as inputs, as follows:

```
from sklearn.metrics import completeness_score
score = completeness_score(true_labels, predicted_labels)
```

Complete Exercise 2.05: Evaluating the Silhouette Coefficient Score and Calinski–Harabasz Index

Complete Activity 2.05: Measuring and Comparing the Performance of the Algorithms

Summary

- Data problems where the input data is unrelated to the labeled output are handled using unsupervised learning models.
- The main objective of such data problems is to understand the data by finding patterns that, in some cases, can be generalized to new instances.
- In this context, this lesson covered clustering algorithms, which work by aggregating similar data points into clusters, while separating data points that differ significantly.

3. Supervised Learning – Key Steps

Overview

- In this lesson, you will learn about key concepts for solving a supervised learning data problem.
- Starting from splitting the dataset to effectively create unbiased models that perform well on unseen data, you will learn how to measure the performance of the model in order to analyze it and take the necessary actions to improve it.
- By the end of this lesson, you will have a firm understanding of how to split a dataset, measure a model's performance, and perform error analysis.

Introduction

- In the preceding lesson, we saw how to solve data problems using unsupervised learning algorithms and applied the concepts that we learned about to a real-life dataset.
- We also learned how to compare the performance of various algorithms and studied two different metrics for performance evaluation.
- In this lesson, we will explore the main steps for working on a supervised machine learning problem

Supervised Learning Tasks

Differing from unsupervised learning algorithms, supervised learning algorithms are characterized by their ability to find relationships between a set of features and a target value (be it discrete or continuous). Supervised learning can solve two types of tasks:

- Classification
- Regression

Model Validation and Testing

- With all the information now available online, it is easy for almost anybody to start working on a machine learning project.

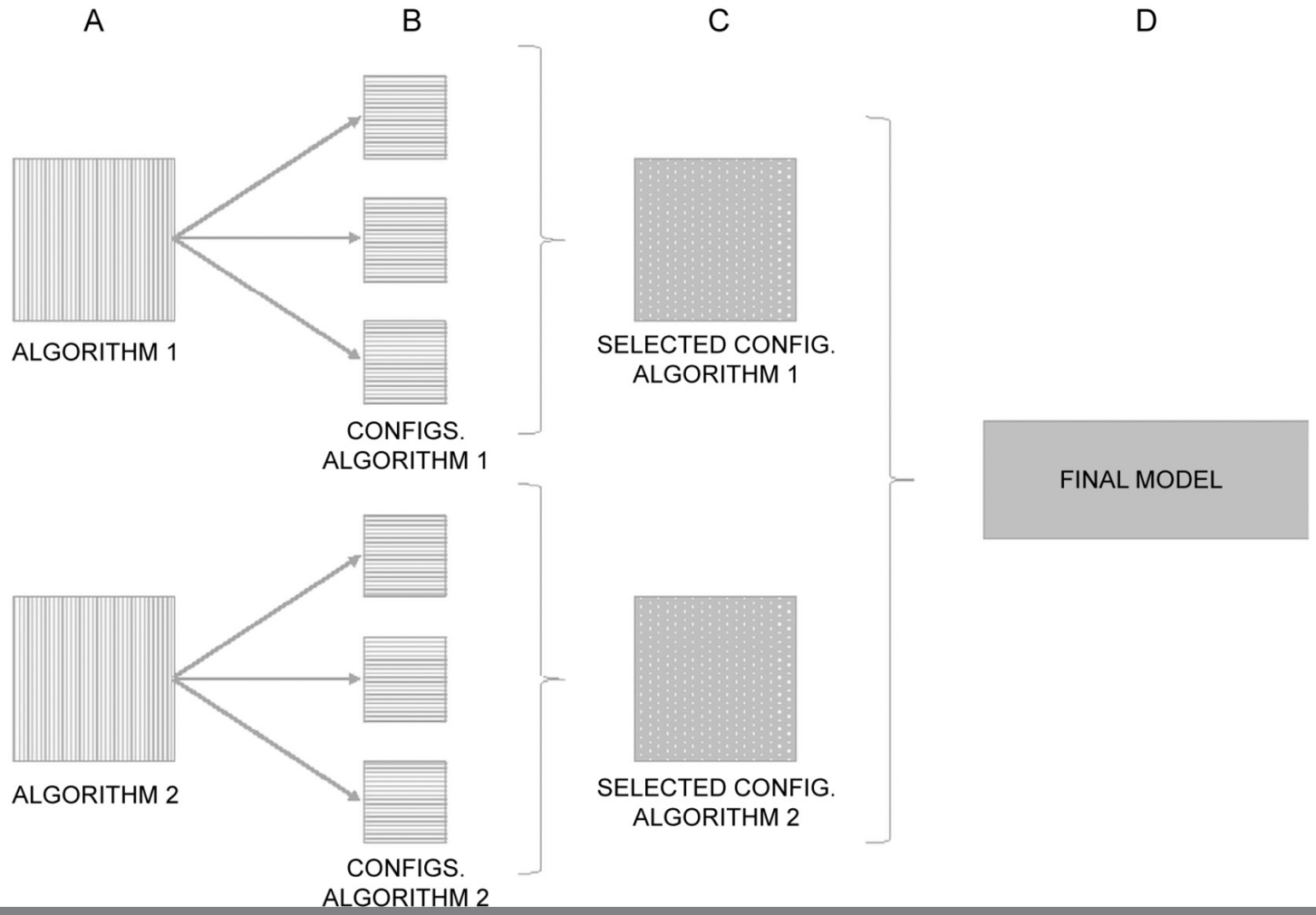
- However, choosing the right algorithm for your data is a challenge when there are many options available.

- Due to this, the decision to use one algorithm over another is achieved through trial and error, where different alternatives are tested.

Data Partitioning

Data partitioning is a process involving dividing a dataset into three subsets so that each set can be used for a different purpose. This way, the development of a model is not affected by the introduction of bias. The following is an explanation of each subset:

- Training set
- Validation set
- Testing set


A
B
C
D
ALGORITHM 1
CONFIGS.
ALGORITHM 1
SELECTED CONFIG.
ALGORITHM 1
FINAL MODEL
ALGORITHM 2
CONFIGS.
ALGORITHM 2
SELECTED CONFIG.
ALGORITHM 2

Split Ratio

Now that the purposes of the various sets are clear, it is important to clarify the split ratio in which data needs to be divided. Although there is no exact science for calculating the split ratio, there are a couple of things to consider when doing so:

- Size of the dataset
- The algorithm:

Split Ratio

Complete Exercise 3.01: Performing a Data Partition on a Sample Dataset

Cross-Validation

- Cross-validation is also a procedure that's used to partition data by resampling the data that's used to train and validate the model.

- It consists of a parameter, K, that represents the number of groups that the dataset will be divided into.

- Due to this, the procedure is also referred to as K-fold cross-validation, where K is usually replaced by a number of your choice.

Cross-Validation

- The procedure of cross-validation is illustrated in the following diagram:

Complete Exercise 3.02: Using Cross-Validation to Partition the Train Set into a Training and a Validation Set

Complete Activity 3.01: Data Partitioning on a Handwritten Digit Dataset

Evaluation Metrics

- Model evaluation is indispensable for creating effective models that not only perform well on the data that was used to train the model but also on unseen data.

- The task of evaluating the model is especially easy when dealing with supervised learning problems, where there is a ground truth that can be compared against the prediction of the model.

Evaluation Metrics for Classification Tasks

- A classification task refers to a model where the class label is a discrete value, as mentioned previously.
- Considering this, the most common measure to evaluate the performance of such tasks is calculating the accuracy of the model, which involves comparing the actual prediction to the real value.
- Even though this may be an appropriate metric in many cases, there are several others to consider as well before choosing one.

Confusion Matrix

The confusion matrix is a table that contains the performance of the model, and is described as follows:

- The columns represent the instances that belong to a predicted class.
- The rows refer to the instances that actually belong to that class (ground truth).

Evaluation Metrics

- The configuration that confusion matrices present allows the user to quickly spot the areas in which the model is having greater difficulty.

- Consider the following table:

Prediction / Ground truth	Pregnant	Not Pregnant
Pregnant	556	44
Not Pregnant	123	477

Evaluation Metrics

Considering that the rows in a confusion matrix refer to the occurrence or non-occurrence of an event, and the columns refer to the model's predictions, the values in the confusion matrix can be explained as follows:

- True positives (TP)
- False positives (FP)
- True negatives (TN)
- False negatives (FN)

Evaluation Metrics

- The values in the confusion matrix can be demonstrated as follows:

	Predicted: True	**Predicted: False**
Actual: True	TP	FN
Actual: False	FP	TN

Accuracy

- Accuracy, as explained previously, measures the model's ability to correctly classify all instances.

- Although this is considered to be one of the simplest ways of measuring performance, it may not always be a useful metric when the objective of the study is to minimize/maximize the occurrence of one class independently of its performance on other classes.

Accuracy

- The accuracy level of the confusion matrix from is measured as follows:

$$Accuracy = \frac{(TP + TN)}{m} = 0.8608 \approx 86\%$$

Precision

- This metric measures the model's ability to correctly classify positive labels (the label that represents the occurrence of the event) by comparing it with the total number of instances predicted as positive.
- This is represented by the ratio between the true positives and the sum of the true positives and false positives, as shown in the following equation:

$$Precision = \frac{TP}{TP + FP}$$

Recall

- The recall metric measures the number of correctly predicted positive labels against all positive labels.

- This is represented by the ratio between true positives and the sum of true positives and false negatives:

$$Recall = \frac{TP}{TP + FN}$$

Complete Exercise 3.03: Calculating Different Evaluation Metrics on a Classification Task

Choosing an Evaluation Metric

- There are several metrics that can be used to measure the performance of a model on classification tasks, and selecting the right one is key to building a model that performs exceptionally well for the purpose of the study.
- Previously, the importance of understanding the purpose of the study was mentioned as a useful insight to determine the pre-processing techniques that need to be performed on the dataset.

Evaluation Metrics for Regression Tasks

- Considering that regression tasks are those where the final output is continuous, without a fixed number of output labels, the comparison between the ground truth and the prediction is based on the proximity of the values rather than on them having exactly the same values.
- For instance, when predicting house prices, a model that predicts a value of USD 299,846 for a house valued at USD 300,000 can be considered to be a good model.

Evaluation Metrics

- **Mean Absolute Error:** This metric measures the average absolute difference between a prediction and the ground truth, without taking into account the direction of the error.

- The formula to calculate the MAE is as follows:

$$MAE = \frac{1}{m} * \sum_{i=1}^{m} |y_i - \hat{y}_i|$$

Evaluation Metrics

- **Root Mean Squared Error:** This is a quadratic metric that also measures the average magnitude of error between the ground truth and the prediction.
- As its name suggests, the RMSE is the square root of the average of the squared differences, as shown in the following formula:

$$RMSE = \sqrt{\frac{1}{m} * \sum_{i=1}^{m} \left(y_i - \hat{y}_i\right)^2}$$

Complete Exercise 3.04: Calculating Evaluation Metrics on a Regression Task

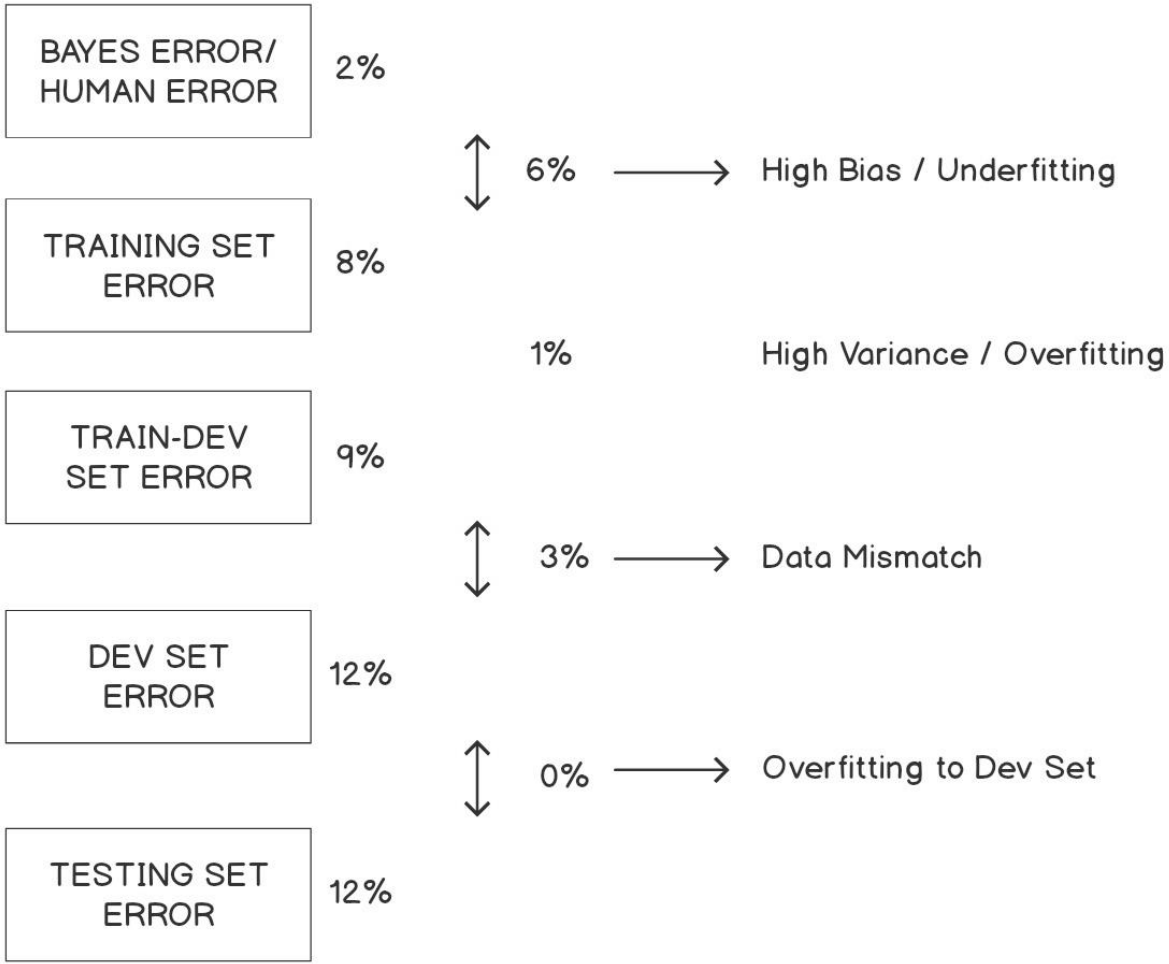
Complete Activity 3.02: Evaluating the Performance of the Model Trained on a Handwritten Dataset

Error Analysis

- Building an average model, as explained so far, is surprisingly easy through the use of the scikit-learn library.
- The key aspects of building an exceptional model come from the analysis and decision-making on the part of the researcher.
- As we have seen so far, some of the most important tasks are choosing and pre-processing the dataset, determining the purpose of the study, and selecting the appropriate evaluation metric.

Bias, Variance, and Data Mismatch

- To understand the different conditions that may affect a machine learning model, it is important to understand what the Bayes error is.
- The Bayes error, also known as the irreducible error, is the lowest possible error rate that can be achieved.
- Before the improvements that were made in technology and artificial intelligence, the Bayes error was considered to be the lowest possible error achievable by humans (human error)


BAYES ERROR/ HUMAN ERROR
2%
6%
High Bias / Underfitting
TRAINING SET ERROR
8%
1%
High Variance / Overfitting
TRAIN-DEV SET ERROR
9%
3%
Data Mismatch
DEV SET ERROR
12%
0%
Overfitting to Dev Set
TESTING SET ERROR
12%

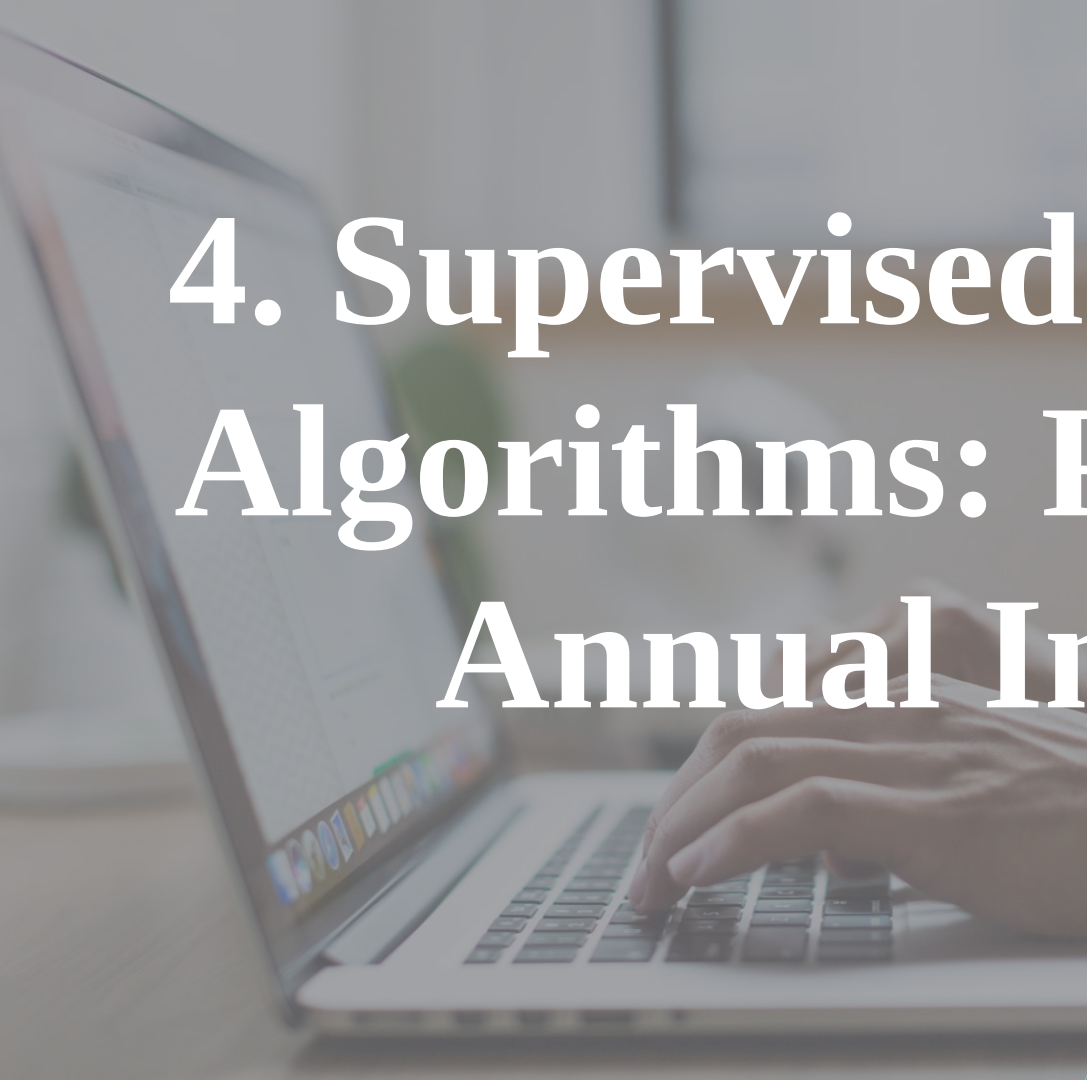
Complete Exercise 3.05: Calculating the Error Rate on Different Sets of Data

Complete Activity 3.03: Performing Error Analysis on a Model Trained to Recognize Handwritten Digits

Summary

- This lesson explained the different tasks that can be solved through supervised learning algorithms: classification and regression.

- Although both of these tasks' goal is to approximate a function that maps a set of features to an output, classification tasks have a discrete number of outputs, while regression tasks can have infinite continuous values as outputs.



4. Supervised Learning Algorithms: Predicting Annual Income

TriveraTech®
TECHNOLOGY TRAINING

Supervised Learning Algorithms

Overview

- In this lesson, we will take a look at three different supervised learning algorithms used for classification.

- We will also solve a supervised learning classification problem using these algorithms and perform error analysis by comparing the results of the three different algorithms.

Introduction

- In the previous lesson, we covered the key steps involved in working with a supervised learning data problem.
- Those steps aim to create high-performing algorithms, as explained in the previous lesson.
- This lesson focuses on applying different algorithms to a real-life dataset, with the underlying objective of applying the steps that we learned previously to choose the best-performing algorithm for the case study.

Exploring the Dataset

- Real-life applications are crucial for cementing knowledge.
- Therefore, this lesson consists of a real-life case study involving a classification task, where the key steps that you learned about in the previous lesson will be applied in order to select the best performing model.
- To accomplish this, the Census Income dataset will be used, which is available at the UC Irvine Machine Learning Repository.

Understanding the Dataset

- To build a model that fits the data accurately, it is important to understand the different details of the dataset, as mentioned in previous lessons.
- First, the data that's available is revised to understand the size of the dataset and the type of supervised learning task to be developed: classification or regression.
- Next, the purpose of the study should be clearly defined, even if it is obvious.

Exploring the Dataset

To edit missing value symbols in Excel, use the Replace functionality, as follows:

- Find what: Input the symbol that is being used to signify a missing value (for example, ?).

- Replace with: Leave it blank (do not enter a space).

Feature	Type	Note	Relevant
`age`	Quantitative (continuous)	The age of the individual	Yes
`workclass`	Qualitative (nominal)	The type of employment of the individual	Yes
`fnlwgt`	Quantitative (continuous)	The number of people the census takers believe the individual represents	No; the values were subjective to the census taker
`education`	Qualitative (ordinal)	The highest education level achieved, by the individual	No; the **`education-num`** feature represents the same information but is preferred because it is presented in numerical form
`education-num`	Quantitative (discrete)	The highest education level achieved in numerical form	Yes
`marital-status`	Qualitative (nominal)	The marital status of the individual	Yes
`occupation`	Qualitative (nominal)	The current occupation of the individual	Yes
`relationship`	Qualitative (nominal)	A relationship value that represents the individual	No; this feature is ignored since its purpose is not clear
`race`	Qualitative (nominal)	The race of the individual	Although (in some cases) this feature may be relevant, for ethical reasons, it will be excluded from the study*
`sex`	Qualitative (nominal)	The gender of the individual	Although (in some cases) this feature may be relevant, for ethical reasons, it will be excluded from the study*
`capital-gain`	Quantitative (continuous)	All of the individual's recorded capital gains	Yes
`capital-loss`	Quantitative (continuous)	All of the individual's recorded capital loss	Yes
`hours-per-week`	Quantitative (continuous)	The number of hours that the individual works per week	Yes
`native-country`	Qualitative (nominal)	The native country of the individual	Yes

Exploring the Dataset

- The preceding process will convert the original dataset into a new dataset with 32,561 instances (since no instances were deleted), but with 9 features and a class label.
- All values should be in their numerical forms. Save the pre-processed dataset into a file using pandas' to_csv function, as per the following code snippet:

```
preprocessed_data.to_csv("census_income_dataset_preprocessed.csv")
```

The Naïve Bayes Algorithm

- Naïve Bayes is a classification algorithm based on Bayes' theorem that naïvely assumes independence between features and assigns the same weight (degree of importance) to all features.
- This means that the algorithm assumes that no single feature correlates to or affects another.
- For example, although weight and height are somehow correlated when predicting a person's age, the algorithm assumes that each feature is independent.

How Does the Naïve Bayes Algorithm Work?

- The algorithm converts the input data into a summary of occurrences of each class label against each feature, which is then used to calculate the likelihood of one event (a class label), given a combination of features.
- Finally, this likelihood is normalized against the likelihood of the other class labels.
- The result is the probability of an instance belonging to each class label.

- Let's take, for example, the data presented in the following tables:

A		
Weather	**Temperature**	**Outcome**
Sunny	Hot	Yes
Sunny	Cool	Yes
Rainy	Cold	No
Sunny	Hot	No
Mild	Cool	Yes
Mild	Cool	Yes
Sunny	Hot	Yes
Rainy	Cool	No
Rainy	Cold	Yes
Sunny	Hot	Yes

B		
Weather	**Temperature**	**Outcome**
Sunny	4	1
Rainy	1	2
Mild	2	0

Temprature	**Yes**	**No**
Hot	3	1
Cool	3	1
Cold	1	1

Overall	**Yes**	**No**
	7	3

The Naïve Bayes Algorithm

- To calculate the likelihood of an event occurring when given a set of features, the algorithm multiplies the probability of the event occurring, given each individual feature, by the probability of the occurrence of the event, independent of the rest of the features, as follows:

Likelihood [A1|E] = P[A1|E1] * P[A1|E2] * … * P[A1|En] * P[A1]

The Naïve Bayes Algorithm

- The preceding equation is calculated for all possible outcomes (all class labels), and then the normalized probability of each outcome is calculated as follows:

$$P[A_1|E] = \frac{likelihood[A_1|E]}{likelihood[A_1|E] + likelihood[A_2|E] + \cdots + likelihood[A_n|E]}$$

- For the example in Figure, given a new instance with weather equal to sunny and temperature equal to cool, the calculation of probabilities is as follows:

$$Likelihood[yes|sunny,cool] = \frac{4}{7} * \frac{3}{7} * \frac{7}{10} = 0.17$$

$$Likelihood[no|sunny,cool] = \frac{1}{3} * \frac{1}{3} * \frac{3}{10} = 0.03$$

$$P[yes|sunny,cool] = \frac{0.17}{0.17+0.03} = 0.85 \approx 85\%$$

$$P[no|sunny,cool] = \frac{0.03}{0.17+0.03} = 0.15 \approx 15\%$$

- For this example, to perform a prediction for a new instance with weather equal to mild and temperature equal to cool using the Laplace estimator, this would be done as follows:

$$Likelihood[yes|mild,cool] = \frac{3}{10} * \frac{4}{10} * \frac{7}{10} = 0.084$$

$$Likelihood[no|mild,cool] = \frac{1}{6} * \frac{2}{6} * \frac{3}{10} = 0.016$$

$$P[yes|mild,cool] = \frac{0.084}{0.084 + 0.016} = 0.84 \approx 84\%$$

$$P[no|mild,cool] = \frac{0.016}{0.084 + 0.016} = 0.16 \approx 16\%$$

Complete Exercise 4.01: Applying the Naïve Bayes Algorithm

Complete Activity 4.01: Training a Naïve Bayes Model for Our Census Income Dataset

The Decision Tree Algorithm

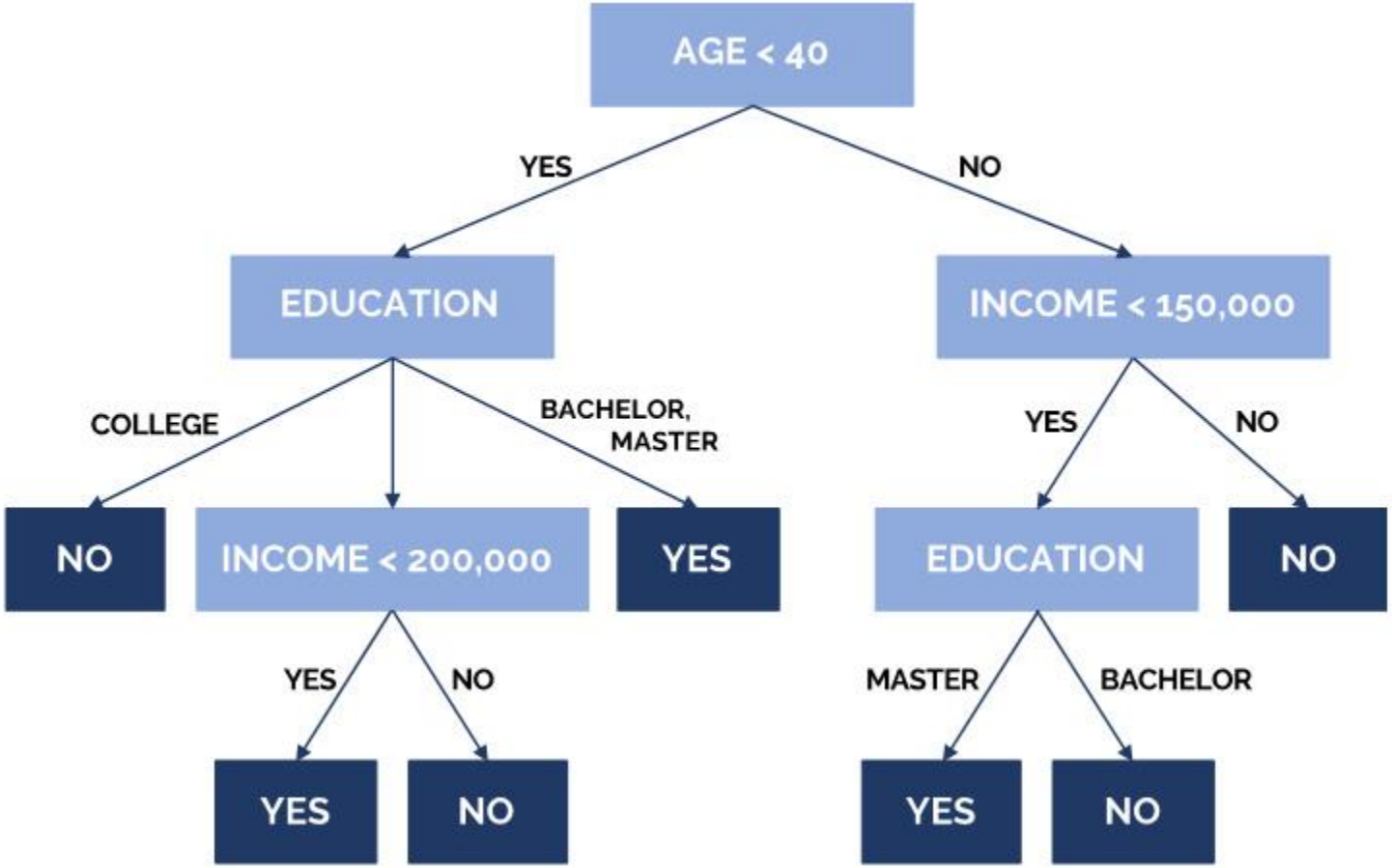
- The decision tree algorithm performs classification based on a sequence that resembles a tree-like structure.
- It works by dividing the dataset into small subsets that serve as guides to develop the decision tree nodes.
- The nodes can be either decision nodes or leaf nodes, where the former represent a question or decision, and the latter represent the decisions made or the final outcome.

How Does the Decision Tree Algorithm Work?

- Considering what we just mentioned, decision trees continually split the dataset according to the parameters defined in the decision nodes.
- Decision nodes have branches coming out of them, where each decision node can have two or more branches.
- The branches represent the different possible answers that define the way in which the data is split.

- For instance, consider the following table, which shows whether a person has a pending student loan based on their age, highest education, and current income:

Age	Highest Level of Education	Current Income	Target
25	Bachelor	0	Yes
32	Doctorate	120,000	Yes
48	Master	120,000	Yes
57	Master	150,000	No
29	College	50,000	No
35	Doctorate	230,000	No
69	Master	120,000	Yes
57	Doctorate	250,000	No
51	Bachelor	90,000	No
30	Master	115,000	Yes


AGE < 40
YES
NO
EDUCATION
INCOME < 150,000
COLLEGE
BACHELOR, MASTER
YES
NO
NO
INCOME < 200,000
YES
EDUCATION
NO
YES
NO
MASTER
BACHELOR
YES
NO
YES
NO

Complete Exercise 4.02: Applying the Decision Tree Algorithm

Complete Activity 4.02: Training a Decision Tree Model for Our Census Income Dataset

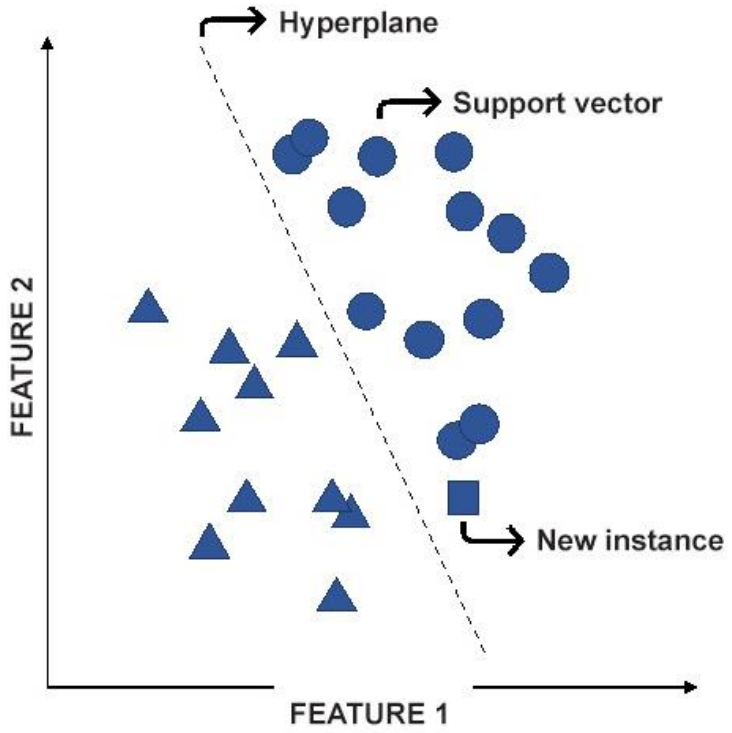
The Support Vector Machine Algorithm

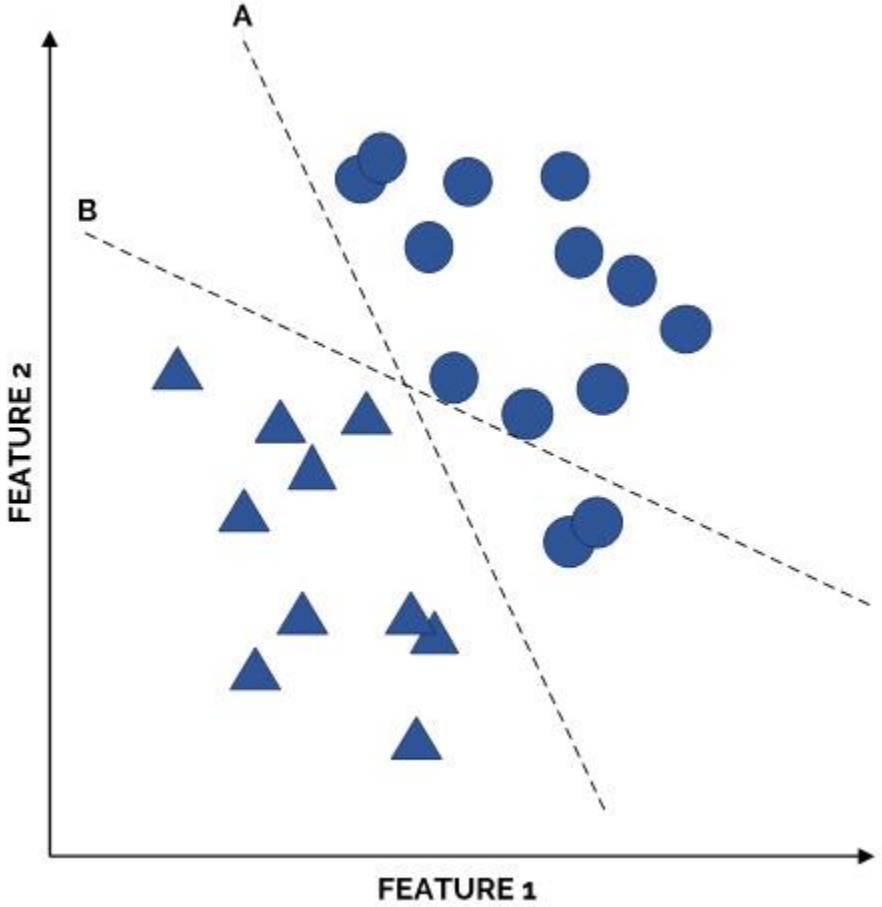
- The Support Vector Machine (SVM) algorithm is a classifier that finds the hyperplane that effectively separates the observations into their class labels.
- It starts by positioning each instance into a data space with n dimensions, where n represents the number of features.
- Next, it traces an imaginary line that clearly separates the instances belonging to a class label from the instances belonging to others.

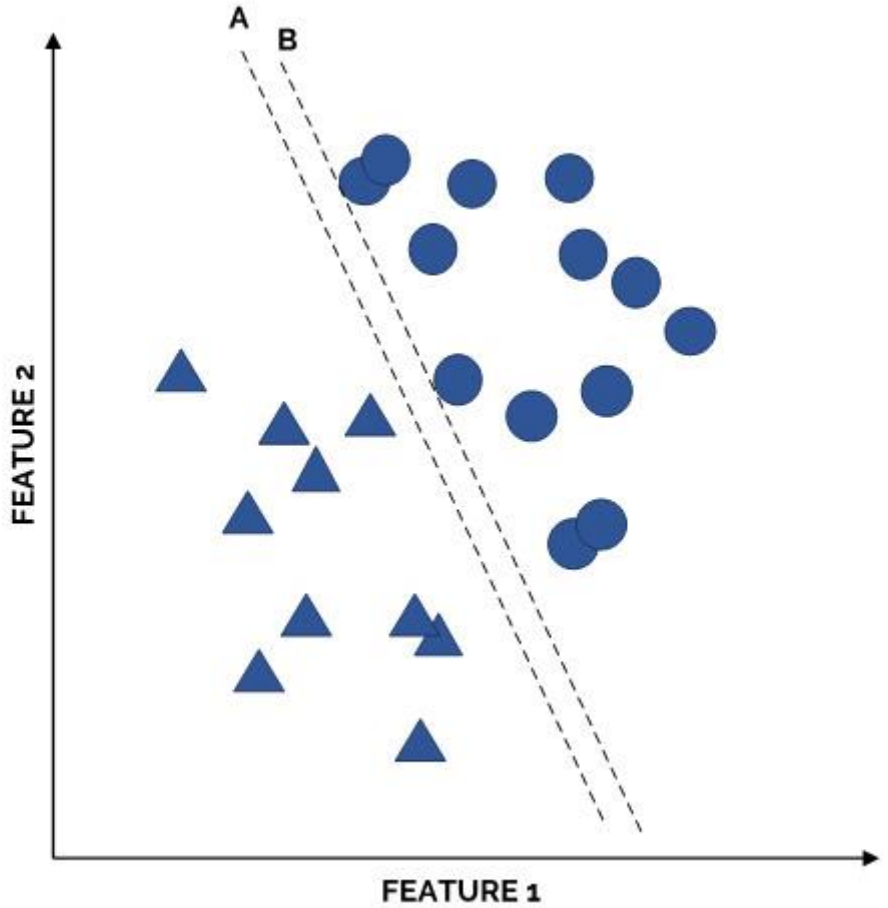
How Does the SVM Algorithm Work?

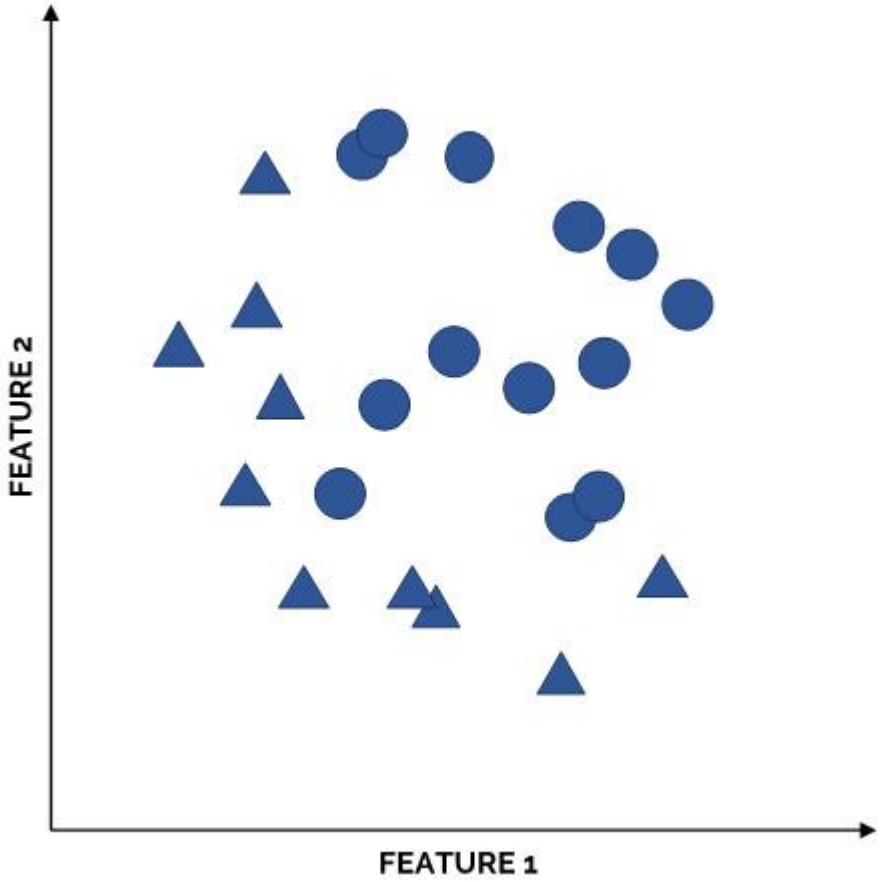
- The following diagram shows a simple example of an SVM model.
- Both the triangles and circular data points represent the instances from the input dataset, where the shapes define the class label that each instance belongs to.
- The dashed line signifies the hyperplane that clearly segregates the data points, which is defined based on the data points' location in the data space.

- The larger the number of features, the more dimensions the data space will have, which will make visually representing the model impossible:




A
B
FEATURE 2
FEATURE 1


A
B
FEATURE 2
FEATURE 1


FEATURE 2
FEATURE 1

How Does the Decision Tree Algorithm Work?

- To segregate these observations, the model would have to draw a circle or another similar shape.
- The algorithm handles this by using kernels (mathematical functions) that can introduce additional features to the dataset in order to modify the distribution of data points into a form that allows a line to segregate them.
- There are several kernels available for this, and the selection of one should be done by trial and error so that you can find the one that best classifies the data that's available.

Complete Exercise 4.03: Applying the SVM Algorithm

Complete Activity 4.03: Training an SVM Model for Our Census Income Dataset

Error Analysis

- In the previous lesson, we explained the importance of error analysis.
- In this section, the different evaluation metrics will be calculated for all three models that were created in the previous activities so that we can compare them.
- For learning purposes, we will compare the models using accuracy, precision, and recall metrics.

Accuracy, Precision, and Recall

First, the three metrics to be used are imported:

```
from sklearn.metrics import accuracy_score, \
precision_score, recall_score
```

- Next, we create two lists containing the different sets of data that will be used inside a for loop to perform the performance calculation on all sets of data for all models:

```
X_sets = [X_train, X_dev, X_test]
Y_sets = [Y_train, Y_dev, Y_test]
```

Accuracy, Precision, and Recall

- A dictionary will be created, which will hold the value of each evaluation metric for each set of data for each model:

```
metrics = {"NB":{"Acc":[],"Pre":[],"Rec":[]},
           "DT":{"Acc":[],"Pre":[],"Rec":[]},
           "SVM":{"Acc":[],"Pre":[],"Rec":[]}}
```

- A for loop is used to go through the different sets of data:

```
for i in range(0,len(X_sets)):
    pred_NB = model_NB.predict(X_sets[i])
    metrics["NB"]["Acc"].append(accuracy_score(Y_sets[i], \
                                pred_NB))
    metrics["NB"]["Pre"].append(precision_score(Y_sets[i], \
                                pred_NB))
    metrics["NB"]["Rec"].append(recall_score(Y_sets[i], \
                                pred_NB))
    pred_tree = model_tree.predict(X_sets[i])
    metrics["DT"]["Acc"].append(accuracy_score(Y_sets[i], \
                                pred_tree))
    metrics["DT"]["Pre"].append(precision_score(Y_sets[i], \
                                pred_tree))
    metrics["DT"]["Rec"].append(recall_score(Y_sets[i], \
                                pred_tree))
    pred_svm = model_svm.predict(X_sets[i])
    metrics["SVM"]["Acc"].append(accuracy_score(Y_sets[i], \
                                 pred_svm))
    metrics["SVM"]["Pre"].append(precision_score(Y_sets[i], \
                                 pred_svm))
    metrics["SVM"]["Rec"].append(recall_score(Y_sets[i], \
                                 pred_svm))
```

Accuracy, Precision, and Recall

- Print the metrics, as follows:

```
print(metrics)
```

- The output is as follows:

```
{'NB': {'Acc': [0.7970975544208546, 0.7902978200798281, 0.8084126496776174],
  'Pre': [0.6683725690890481, 0.6816901408450704, 0.6873239436619718],
  'Rec': [0.3123405612244898, 0.2980295565024633, 0.3223249669749093]},
 'DT': {'Acc': [0.9723960532882866, 0.8114829597789377, 0.8234571691740866],
  'Pre': [0.9827856025039123, 0.6316489361702128, 0.622641509433962],
  'Rec': [0.9011479591836735, 0.584975369458128, 0.6103038309114928]},
 'SVM': {'Acc': [0.8024724536414942, 0.7958243782622045, 0.8099478047282775],
  'Pre': [0.7411210954214805, 0.7474747474747475, 0.7429577464788732],
  'Rec': [0.2761479591836734, 0.2733990147783251, 0.2787318361955086]}}
```

- From the preceding snippets, the following results are obtained:

		Naïve Bayes	Decision Tree	SVM
Accuracy	Training sets	0.7971	0.9724	0.8025
	Validation sets	0.7902	0.8115	0.7958
	Testing sets	0.8084	0.8235	0.8099
Precision	Training sets	0.6684	0.9828	0.7411
	Validation sets	0.6817	0.6316	0.7475
	Testing sets	0.6873	0.6226	0.7429
Recall	Training sets	0.3123	0.9011	0.2761
	Validation sets	0.2980	0.5849	0.2734
	Testing sets	0.3223	0.6103	0.2787

Summary

- Using the knowledge from previous lessons, we started this lesson by performing an analysis of the Census Income dataset, with the objective of understanding the data that's available and making decisions about the pre-processing process.

- Three supervised learning classification algorithms—the Naïve Bayes algorithm, the Decision Tree algorithm, and the SVM algorithm—were explained, and were applied to the previously pre-processed dataset to create models that generalized to the training data.

5. Supervised Learning – Key Steps

Supervised Learning – Key Steps

Overview

- In this lesson, we will dive deep into the concept of neural networks and describe the processes of forward and backpropagation.
- We will solve a supervised learning classification problem using a neural network and analyze the results of the neural network by performing error analysis.

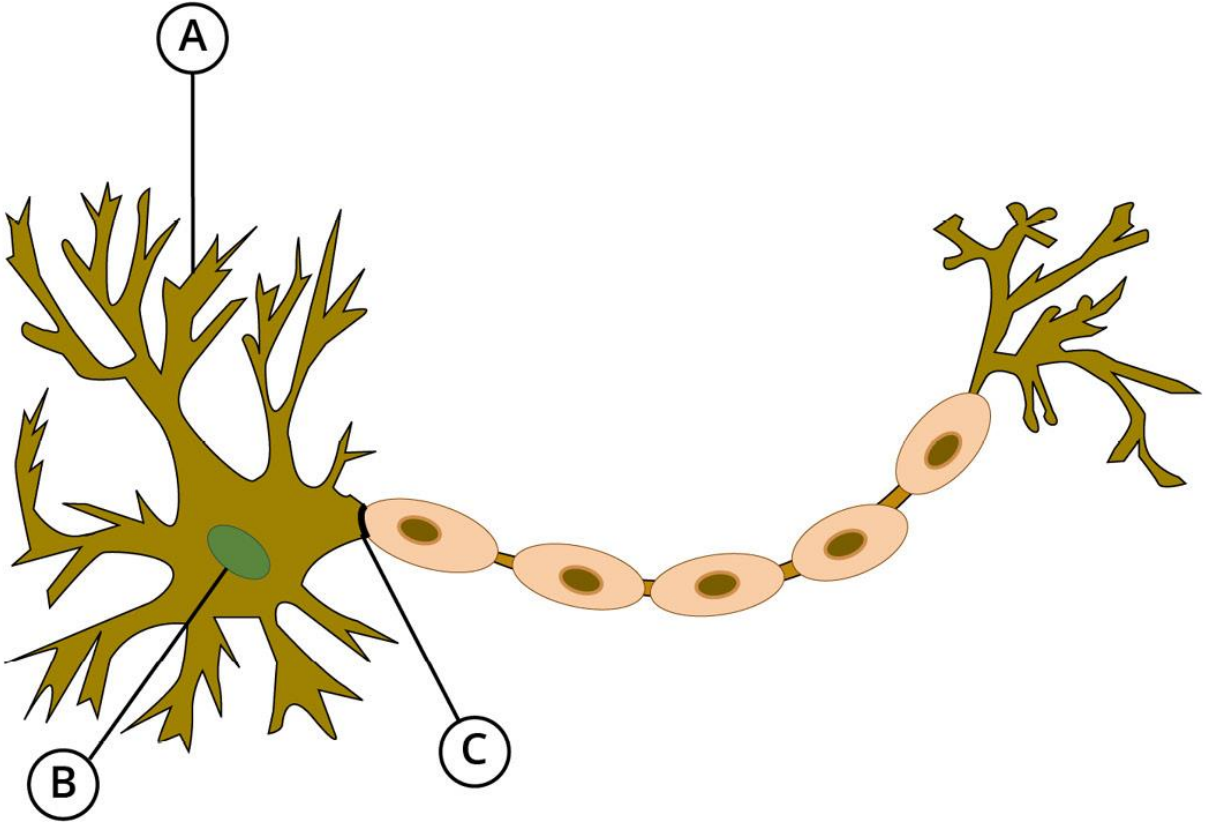
Introduction

- In the preceding lesson, we explored three machine learning algorithms to solve supervised learning tasks, either for classification or regression.
- In this lesson, we will explore one of the most popular machine learning algorithms nowadays, artificial neural networks, which belong to a subgroup of machine learning called deep learning.
- Artificial neural networks (ANNs), also known as Multilayer Perceptrons (MLPs), have become increasingly popular mostly because they present a complex algorithm that can approach almost any challenging data problem.

Artificial Neural Networks

- Although there are several machine learning algorithms available to solve data problems, as we have already stated

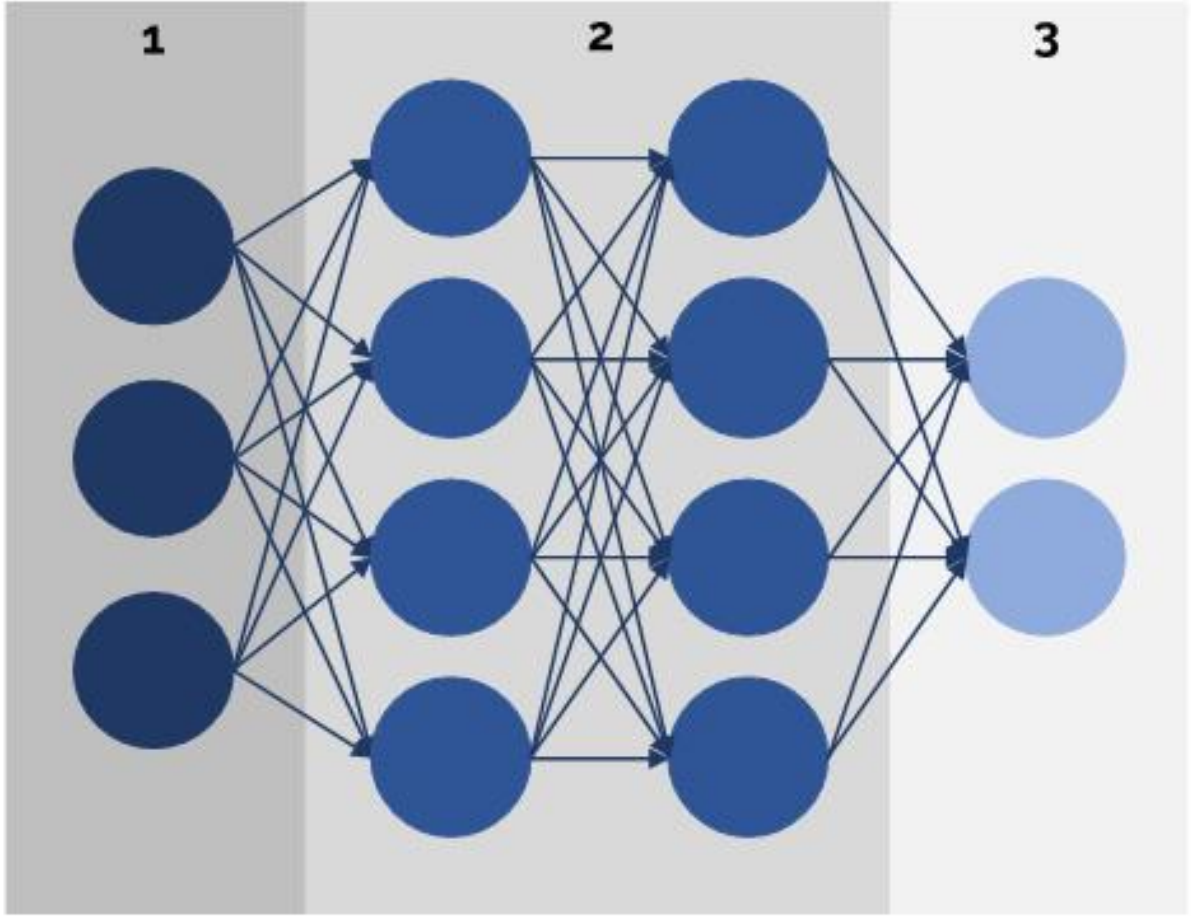
- ANNs have become increasingly popular among data scientists, on account of their ability to find patterns in large and complex datasets that cannot be interpreted by humans.


A
B
C

How Do ANNs Work?

Before we dive into the process that is followed by an ANN, let's start by looking at its main components:

- Input layer
- Hidden layers
- Output layer


1
2
3

Forward Propagation

- The input layer feeds the initial information to the ANN.
- The processing of the data is done by propagating data bits through the depth (number of hidden layers) and width (number of units in each layer) of the network.
- The information is processed by each neuron in each layer using a linear function, coupled with an activation function that aims to break the linearity, as follows:

$$Z_1 = W_1 * X + b_1$$

$$A_1 = \sigma(Z_1)$$

Forward Propagation

- The preceding two formulas are calculated for each neuron in each layer, where the value of X for the hidden layers (other than the input layer) is replaced by the output of the previous layer (An), as follows:

$$Z_2 = W_2 * A_1 + b_2$$

$$A_2 = \sigma(Z_2)$$

Cost Function

- Considering that the final objective of the training process is to build a model based on a given set of data that maps an expected output, it is particularly important to measure the model's ability to estimate a relation between X and Y by comparing the differences between the predicted value (Y_hat) and the ground truth (Y).
- This is accomplished by calculating the cost function (also known as the loss function) to determine how poor the model's predictions are.

Cost Function

- For a binary classification task, that is, tasks with only two class output labels, the cross-entropy cost function is calculated as follows:

cost = -(y * log(yhat) + (1-y) *(1-yhat))

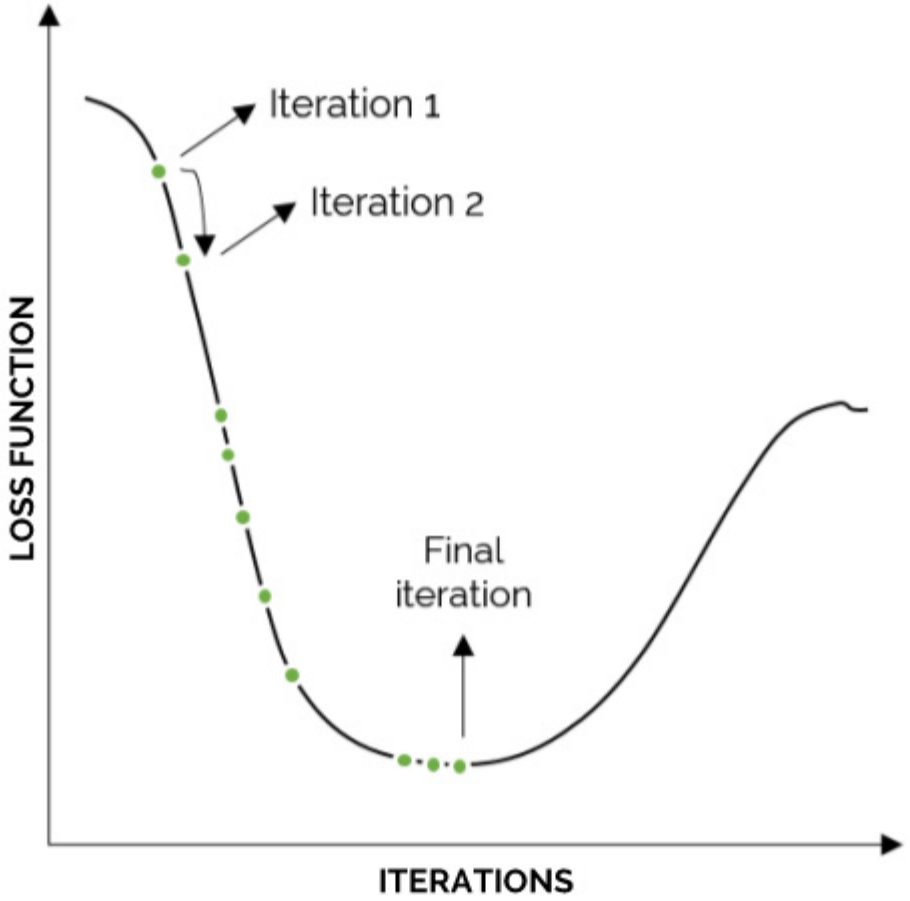
Cost Function

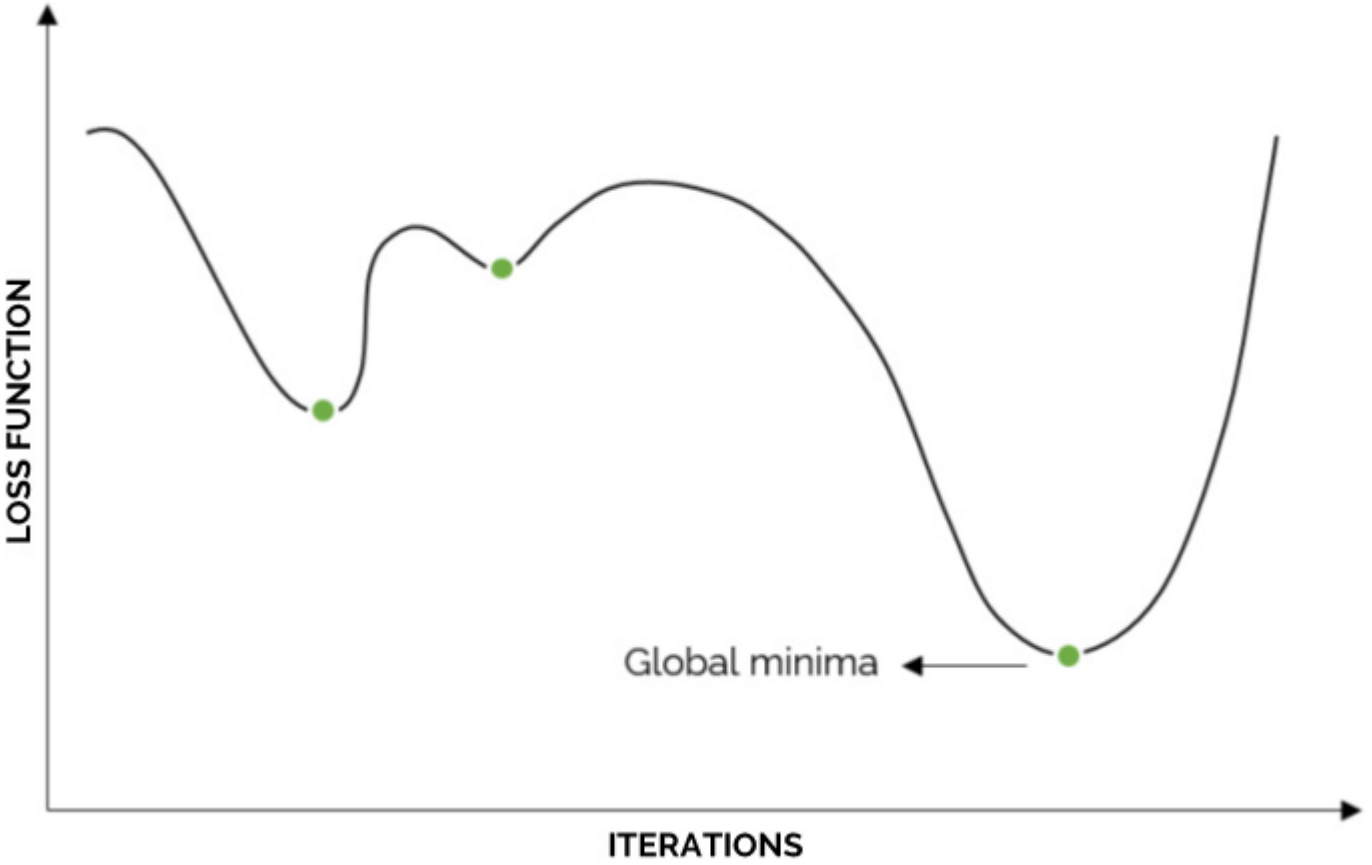
- For a multiclass classification task, the formula is as follows:

$$cost = -\sum_{c=1}^{M} y_c * \log(y_{hat,c})$$

Backpropagation

- The backpropagation procedure was introduced as part of the training process of ANNs to make learning faster.
- It basically involves calculating the partial derivatives of the cost function with respect to the weights and biases along the network.
- The objective of this is to minimize the cost function by changing the weights and the biases.


Iteration 1
Iteration 2
LOSS FUNCTION
Final
iteration
ITERATIONS


LOSS FUNCTION
Global minima
ITERATIONS

Updating the Weights and Biases

- Taking the derivatives' average that was calculated during backpropagation, the final step of an iteration is to update the values of the weights and biases.
- This process is done using the following formula for updating weights and biases:

New weight = old weight – derivative rate * learning rate
New bias = old bias – derivative rate * learning rate

Understanding the Hyperparameters

Hyperparameters, as you have seen so far, are parameters that can be fine-tuned to improve the accuracy of a model. For neural networks, hyperparameters can be classified into two main groups:

- Those that alter the structure of the network
- Those that modify the process to train it

Number of Hidden Layers and Units

- The number of hidden layers and the number of units in each layer can be set by the researcher, as mentioned previously.
- Again, there is no exact science to select this number, and, on the contrary, the selection of this number is part of the fine-tuning process to test different approximations.
- Nonetheless, when selecting the number of hidden layers, some data scientists lean toward an approach wherein multiple networks are trained, each with an extra layer.

Activation Functions

As mentioned previously, the activation function is used to introduce non-linearity to the model. The most commonly used activation functions are the following:

- ReLU
- Tanh
- Sigmoid
- Softmax

Regularization

- Regularization is a technique used in machine learning to improve a model that is suffering from overfitting, which means that this hyperparameter is mostly used when it is strictly required, and its main objective is to increase the generalization ability of the model.

- There are different regularization techniques, but the most common ones are the L1, L2, and dropout techniques.

Batch Size

- Another hyperparameter to be tuned during the construction of an ANN is the batch size.
- This refers to the number of instances to be fed to the neural network during an iteration, which will be used to perform a forward and a backward pass through the network.
- For the next iteration, a new set of instances will be used.

Learning Rate

- The learning rate, as explained previously, is introduced to help determine the size of the steps that the model will take to get to the local or global minima in each iteration.

- The lower the learning rate, the slower the learning process of the network, but this results in better models

Number of Iterations

- A neural network is trained through an iterative process, as mentioned previously.
- Therefore, it is necessary to set the number of iterations that the model will perform.
- The best way to set up the ideal number of iterations is to start low, between 200 and 500, and increase it, in the event that the plot of the cost function over each iteration shows a decreasing line.

Applications of Neural Networks

- In addition to the preceding architecture, a number of new architectures have emerged over time, thanks to the popularity of neural networks.

- Some of the most popular ones are convolutional neural networks, which can handle the processing of images by using filters as layers, and recurrent neural networks, which are used to process sequences of data such as text translations.

Limitations of Neural Networks

Some of the limitations of training neural networks are as follows:

- The training process takes time. Regardless of the hyperparameters used, they generally take time to converge.
- They need very large datasets in order to work better. Neural networks are meant for larger datasets, as their main advantage is their ability to find patterns within millions of values.

Applying an Artificial Neural Network

- Now that you know the components of an ANN, as well as the different steps that it follows to train a model and make predictions, let's train a simple network using the scikit-learn library.

- In this topic, scikit-learn's neural network module will be used to train a network using the datasets used in the previous lesson's exercises and activities (that is, the Fertility Dataset and the Processed Census Income Dataset).

Scikit-Learn's Multilayer Perceptron

- An MLP is a supervised learning algorithm that, as the name indicates, uses multiple layers (hidden layers) to learn a non-linear function that translates the input values into output, either for classification or regression.
- As we explained previously, the job of each unit of a layer is to transform the data received from the previous layer by calculating a linear function and then applying an activation function to break the linearity.

Complete Exercise 5.01: Applying the MLP Classifier Class

Complete Activity 5.01: Training an MLP for Our Census Income Dataset

Performance Analysis

- In the following section, we will first perform error analysis using the accuracy metric as a tool to determine the condition that is affecting (in greater proportion) the performance of the algorithm.
- Once the model is diagnosed, the hyperparameters can be tuned to improve the overall performance of the algorithm.
- The final model will be compared to those that were created during the previous lesson in order to determine whether a neural network outperforms the other models.

Error Analysis

- Using the accuracy score calculated in previous Activity, Training an MLP for Our Census Income Dataset, we can calculate the error rates for each of the sets and compare them against one another to diagnose the condition that is affecting the model.

- To do so, a Bayes error equal to 1% will be assumed, considering that other models in the previous lesson were able to achieve an accuracy level of over 97%.

Error Analysis

	Accuracy score	Error rate	Difference
Bayes error		0.01	
Training sets	0.8465	0.1535	0.1435
Validation sets	0.8246	0.1754	0.0219
Testing sets	0.8415	0.1585	-0.0169

- Considering that both the number of iterations and the size of the network (number of layers and units) should be changed using a trial-and-error approach, the following experiments will be performed:

	Default values	Experiment 1	Experiment 2	Experiment 3
Number of iterations	200	500	500	500
Number of hidden layers	1	1	2	3
Number of units per layer	100	100	100,100	100,100,100

Hyperparameter Fine-Tuning

- Similar to adding the random_state argument to the initialization of the MLP, the change in the values of the number of iterations and the size of the network can be achieved using the following code, which shows the values for Experiment 3:

```
from sklearn.neural_network import MLPClassifier
model = MLPClassifier(random_state=101, max_iter = 500, \
                      hidden_layer_sizes=(100,100,100))
model = model.fit(X_train, Y_train)
```

Hyperparameter Fine-Tuning

- The accuracy scores from running the preceding experiments can be seen in the following table:

	Initial model	Experiment 1	Experiment 2	Experiment 3
Training sets	0.8465	0.8243	0.8673	0.8494
Validation sets	0.8246	0.8031	0.8311	0.8299
Testing sets	0.8415	0.8222	0.8520	0.8428

- Nonetheless, in order to test the width of the hidden layers, the following experiments will be considered, using the selected values for the number of iterations and the number of hidden layers of Experiment 2, but varying the number of units in each layer:

	Initial model (Experiment 2)	Experiment 2.1	Experiment 2.2
Number of iterations	500	500	500
Number of hidden layers	2	2	2
Number of units per layer	100,100	50,50	150,150

Hyperparameter Fine-Tuning

- The accuracy score of the two experiments is shown, followed by an explanation of the logic behind them:

	Initial model (Experiment 2)	**Experiment 2 .1**	**Experiment 2.2**
Training sets	0.8673	0.8523	0.8590
Validation sets	0.8311	0.8289	0.8219
Testing sets	0.8520	0.8443	0.8455

Model Comparison

- When more than one model has been trained, the final step related to the process of creating a model is a comparison between the models in order to choose the one that best represents the training data in a generalized way, so that it works well over unseen data.
- The comparison, as mentioned previously, must be done by using only the metric that was selected to measure the performance of the models for the data problem

Complete Activity 5.02: Comparing Different Models to Choose the Best Fit for the Census Income Data Problem

Summary

- This lesson mainly focused on ANNs (the MLP, in particular), which have become increasingly important in the field of machine learning due to their ability to tackle highly complex data problems that usually use extremely large datasets with patterns that are impossible to see with the human eye.
- The main objective is to emulate the architecture of the human brain by using mathematical functions to process data.

6. Building Your Own Program

Building Your Own Program

Overview

- In this lesson, we will present all the steps required to solve a problem using machine learning. We will take a look at the key stages involved in building a comprehensive program.

- We will save a model in order to get the same results every time it is run and call a saved model to use it for predictions on unseen data.

Introduction

- In the previous lessons, we covered the main concepts of machine learning, beginning with the distinction between the two main learning approaches (supervised and unsupervised learning), and then moved on to the specifics of some of the most popular algorithms in the data science community.
- This lesson will talk about the importance of building complete machine learning programs, rather than just training models.
- This will involve taking the models to the next level, where they can be accessed and used easily.

Program Definition

- The following section will cover the key stages required to construct a comprehensive machine learning program that allows easy access to the trained model so that we can perform predictions for all future data.
- These stages will be applied to the construction of a program that allows a bank to determine the promotional strategy for a financial product in its marketing campaign.

Building a Program – Key Stages

- At this point, you should be able to pre-process a dataset, build different models using training data, and compare those models in order to choose the one that best fits the data at hand.

- These are some of the processes that are handled during the first two stages of building a program, which ultimately allows the creation of the model.

Preparation

Preparation consists of all the procedures that we have developed thus far, with the objective of outlining the project in alignment with the available information and the desired outcome. The following is a brief description of the three processes in this stage (these have been discussed in detail in previous lessons):

- Data Exploration
- Data Pre-processing
- Data Splitting

Creation

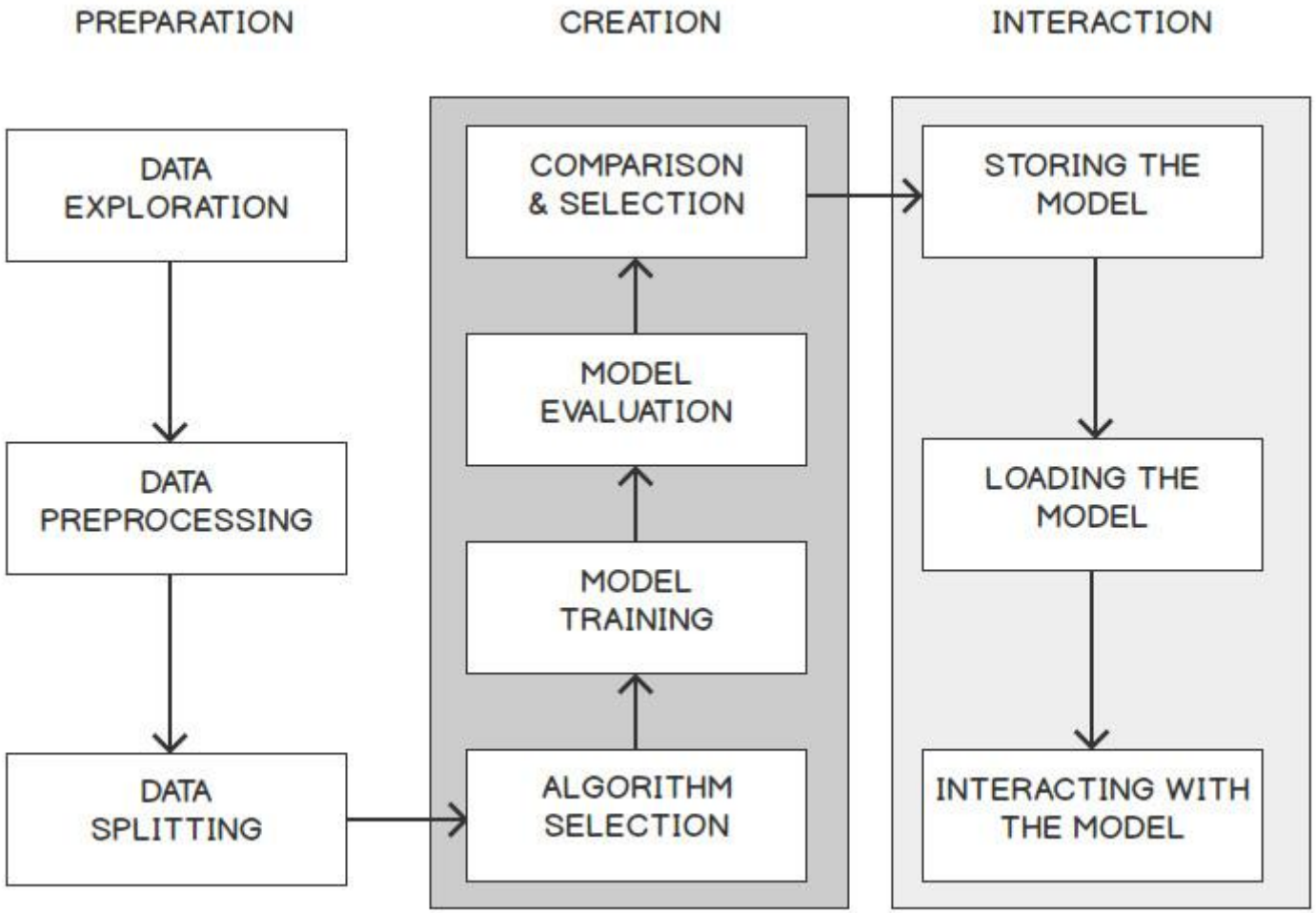
This stage involves all of the steps that are required to create a model that fits the data that is available. This can be done by selecting different algorithms, training and tuning them, comparing the performance of each, and, finally, selecting the one that generalizes best to the data (meaning that it achieves better overall performance). The processes in this stage will be discussed briefly, as follows:

- Algorithm Selection
- Training Process
- Model Evaluation
- Model Comparison and Selection

Interaction

The final stage in building a comprehensive machine learning program consists of allowing the final user to easily interact with the model. This includes the process of saving the model into a file, calling the file that holds the saved model, and developing a channel through which users can interact with the model:

- Storing the Final Model
- Loading the Model
- Channel of Interaction.


PREPARATION
CREATION
INTERACTION
DATA EXPLORATION
DATA PREPROCESSING
DATA SPLITTING
ALGORITHM SELECTION
MODEL TRAINING
MODEL EVALUATION
COMPARISON & SELECTION
STORING THE MODEL
LOADING THE MODEL
INTERACTING WITH THE MODEL

Understanding the Dataset

- To learn how to implement the processes in the Interaction section, we will build a program that's capable of predicting whether a person will be interested in investing in a term deposit, which will help the bank target its promotion efforts.

- A term deposit is money that is deposited into a banking institution that cannot be withdrawn for a specific period of time.

Program Definition

Once you have accessed the link of the UC Irvine Machine Learning repository, follow these steps to download the dataset:

- First, click on the Data Folder link.
- Click the bank hyperlink to trigger the download

Program Definition

- Import the required libraries:

```
import pandas as pd
import numpy as np
```

- As we have learned thus far, the dataset can be loaded into a Jupyter Notebook using Pandas:

```
data = pd.read_csv("bank-full.csv")
data.head()
```

Program Definition

- The DataFrame will look as follows:

	age;"job";"marital";"education";"default";"balance";"housing";"loan";"contact";"day";"month";"duration";"campaign";"pdays";"previous";"poutcome";"y"
0	58;"management";"married";"tertiary";"no";2143...
1	44;"technician";"single";"secondary";"no";29;"...
2	33;"entrepreneur";"married";"secondary";"no";2...
3	47;"blue-collar";"married";"unknown";"no";1506...
4	33;"unknown";"single";"unknown";"no";1;"no";"n...

Program Definition

- This can be fixed by adding the delimiter parameter to the read_csv function and defining the semicolon as the delimiter, as shown in the following code snippet:

```
data = pd.read_csv("bank-full.csv", delimiter = ";")
data.head()
```

Program Definition

- After this step, the data should look as follows:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	unknown	no
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unknown	no
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	0	unknown	no
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1	0	unknown	no
4	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	1	-1	0	unknown	no

Program Definition

- To aid the process of dealing with missing values, all unknown values will be replaced by NaN using Pandas' replace function, as well as NumPy, as follows:

```
data = data.replace("unknown", np.NaN)
data.head()
```

Program Definition

- By printing the head of the data variable, the output of the preceding code snippet is as follows:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
0	58	management	married	tertiary	no	2143	yes	no	NaN	5	may	261	1	-1	0	NaN	no
1	44	technician	single	secondary	no	29	yes	no	NaN	5	may	151	1	-1	0	NaN	no
2	33	entrepreneur	married	secondary	no	2	yes	yes	NaN	5	may	76	1	-1	0	NaN	no
3	47	blue-collar	married	NaN	no	1506	yes	no	NaN	5	may	92	1	-1	0	NaN	no
4	33	NaN	single	NaN	no	1	no	no	NaN	5	may	198	1	-1	0	NaN	no

Program Definition

- Finally, the edited dataset is saved in a new .csv file so that it can be used for the activities throughout this lesson.

- You can do this by using the to_csv function, as follows:

```
data.to_csv("bank-full-dataset.csv")
```

- The following table displays brief descriptions of all 16 features.
- This will help you determine the relevance of each feature to the study, and will provide an idea of some of the steps required to pre-process the data:

Name	Type	Description
age	Quantitative (continuous)	The age of the individual.
job	Qualitative (nominal)	The type of job the individual currently has. For instance: "blue-collar".
marital	Qualitative (nominal)	The marital status of the individual.
education	Qualitative (ordinal)	The highest education level achieved by the individual.
default	Qualitative (nominal - binary)	Whether the individual has credit by default
balance	Quantitative (continuous)	Average yearly balance of the individual in euros.
housing	Qualitative (nominal - binary)	Whether the individual has any housing loan.
loan	Qualitative (nominal - binary)	Whether the individual has any personal loan.
contact	Qualitative (nominal)	The mode of communication used to contact the individual for the current campaign.
day	Quantitative (discrete)	The day of the month when the individual was last contacted for the current campaign.
month	Qualitative (nominal)	The month of the year when the individual was last contacted for the current campaign.
duration	Quantitative (continuous)	The duration, in seconds, of the last contact with the individual for the current campaign.
campaign	Quantitative (continuous)	The number of times the individual was contacted during the promotion campaign.
pdays	Quantitative (continuous)	The number of days that passed by after the individual was contacted for a previous campaign. The value -1 means that the client was not contacted for a previous campaign.
previous	Quantitative (continuous)	The number of times the individual was contacted for previous campaigns.
poutcome	Qualitative (nominal)	The outcome obtained from the previous campaign.

Complete Activity 6.01: Performing the Preparation and Creation Stages for the Bank Marketing Dataset

Saving and Loading a Trained Model

- Although the process of manipulating a dataset and training the right model is crucial for developing a machine learning project, the work does not end there.

- Knowing how to save a trained model is key as this will allow you to save the hyperparameters, as well as the values for the weights and biases of your final model, so that it remains unchanged when it is run again.

Saving a Model

- The process of saving a model is also called serialization, and it has become increasingly important due to the popularity of neural networks that use many parameters (weights and biases) that are randomly initialized every time the model is trained

- As well as due to the introduction of bigger and more complex datasets that make the training process last for days, weeks, and sometimes months.

Complete Exercise 6.01: Saving a Trained Model

Loading a Model

- The process of loading a model is also known as deserialization, and it consists of taking the previously saved file, deserializing it, and then loading it into code or Terminal so that you can use the model on new data.

- The pickle module is also used to load the model.

Complete Exercise 6.02: Loading a Saved Model

Complete Activity 6.02: Saving and Loading the Final Model for the Bank Marketing Dataset

Interacting with a Trained Model

- Once the model has been created and saved, it is time for the last step of building a comprehensive machine learning program: allowing easy interaction with the model.

- This step not only allows the model to be reused, but also introduces efficiency to the implementation of machine learning solutions by allowing you to perform classifications using just input data.

Interacting with a Trained Model

Interacting with a Trained Model

- The backpropagation procedure was introduced as part of the training process of ANNs to make learning faster.
- It basically involves calculating the partial derivatives of the cost function with respect to the weights and biases along the network.
- The objective of this is to minimize the cost function by changing the weights and the biases.

Complete Exercise 6.03: Creating a Class and a Channel to Interact with a Trained Model

Complete Activity 6.03: Allowing Interaction with the Bank Marketing Dataset Model

Summary

- This lesson wraps up all of the concepts and techniques that are required to successfully train a machine learning model based on training data.

- In this lesson, we introduced the idea of building a comprehensive machine learning program that not only accounts for the stages involved in the preparation of the dataset and creation of the ideal models

THANK YOU !!