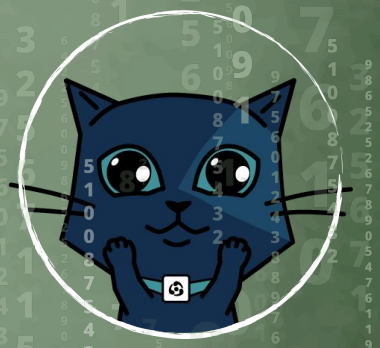
exactpro

ISTQB® AI Testing course

Chapter 3. Machine Learning (ML) – Overview

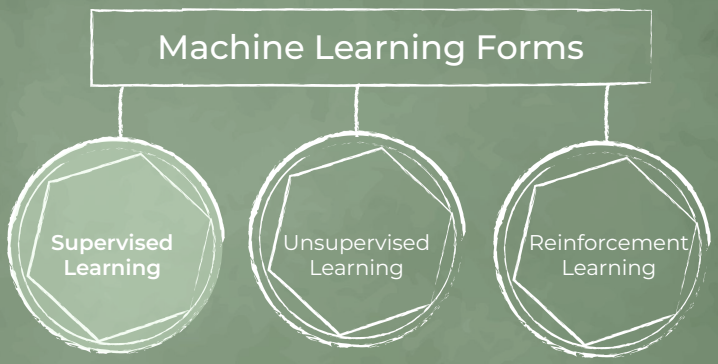


Iosif Itkin, Iuliia Emelianova, Dmitrii Degtiarenko

Contents

3.1 Forms of Machine Learning

3.1.1 Supervised Learning

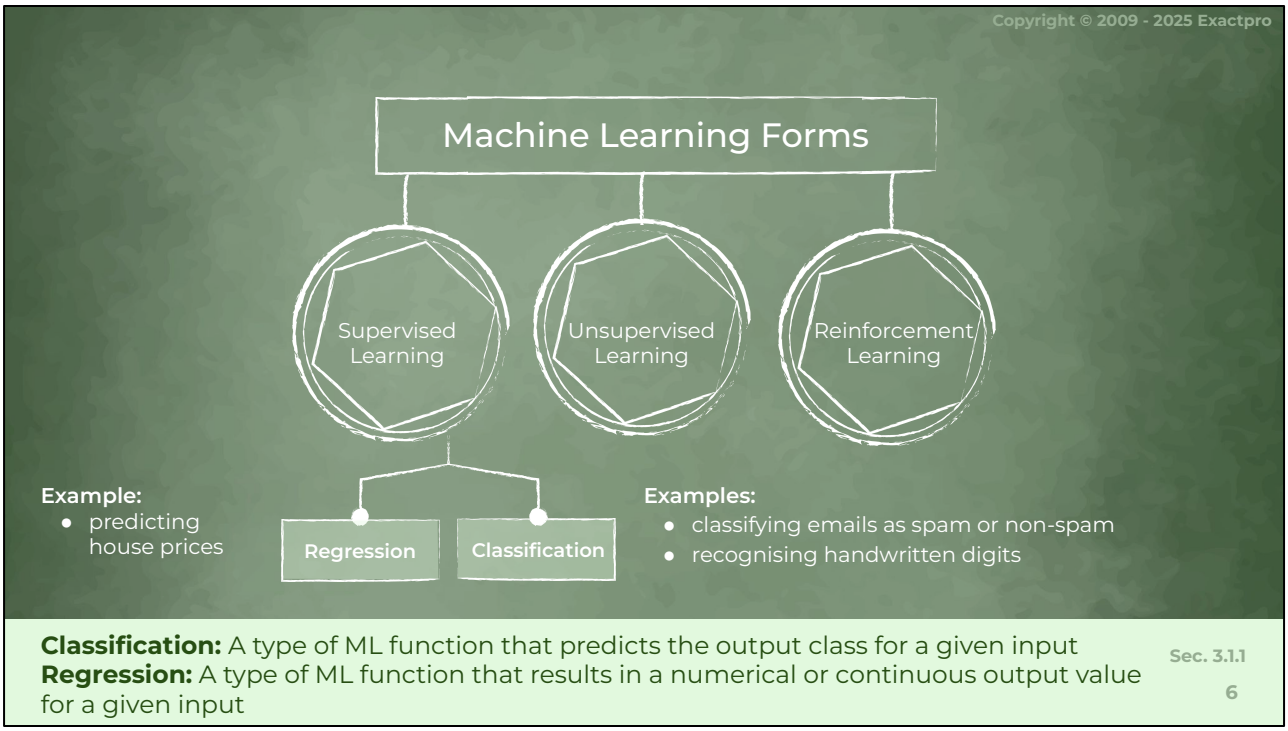


- **Goal:** To learn a mapping function that can accurately predict the output labels for new input data
- **Training phase: (model training/fitting)** The algorithm analyses the labelled examples and adjusts its internal parameters to minimise the difference between the predicted outputs and the true outputs
- **Testing/prediction phase:** A new data set is applied to the trained model to predict the output. The model is deployed once the output precision level is satisfactory

Supervised learning: Training an ML model from input data and its corresponding labels
Training dataset: A dataset used to train an ML model. It consists of input samples and their corresponding correct output labels

We shall begin this chapter with reiterating something we talked about in chapter 1 and that is that ML algorithms can be categorised as supervised learning, unsupervised learning, and reinforcement learning. **Supervised learning** is a machine learning technique where an algorithm learns to map input data to the corresponding desired output labels, given a set of labelled examples.

Here, a training dataset is provided, which consists of input samples and their corresponding correct output labels. The goal of the algorithm is to learn a mapping function that can accurately predict the output labels for new input data. The process of supervised learning involves two main components: the training phase and the prediction (or testing) phase. During the training phase, the algorithm analyses the labelled examples and adjusts its internal parameters to minimise the difference between the predicted outputs and the true outputs. This process is often referred to as ***model training*** or ***model fitting***.

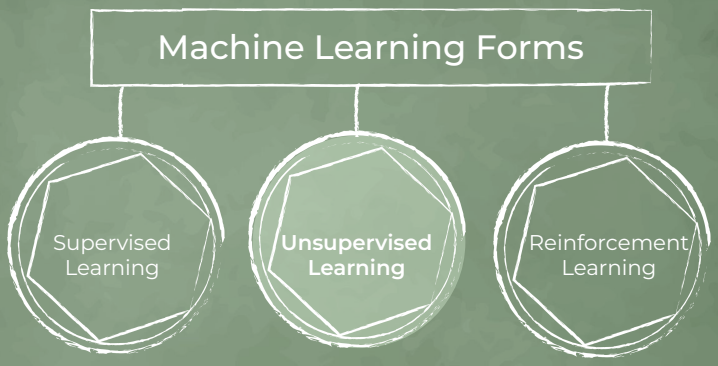


Classification: A type of ML function that predicts the output class for a given input
Regression: A type of ML function that results in a numerical or continuous output value for a given input

Supervised learning aids us with issues of:

- **classification**, which is when the problem requires an input to be classified into one of a few predefined classes (for example, classifying emails as spam or non-spam, or recognising handwritten digits).
- **regression**, which is when the problem requires the ML model to predict a numeric output using regression (for instance, in predicting house prices, the algorithm can learn from a dataset containing features such as the size, number of bedrooms, and location of houses, along with their corresponding sale prices. It can then predict the price of a new house based on its features).

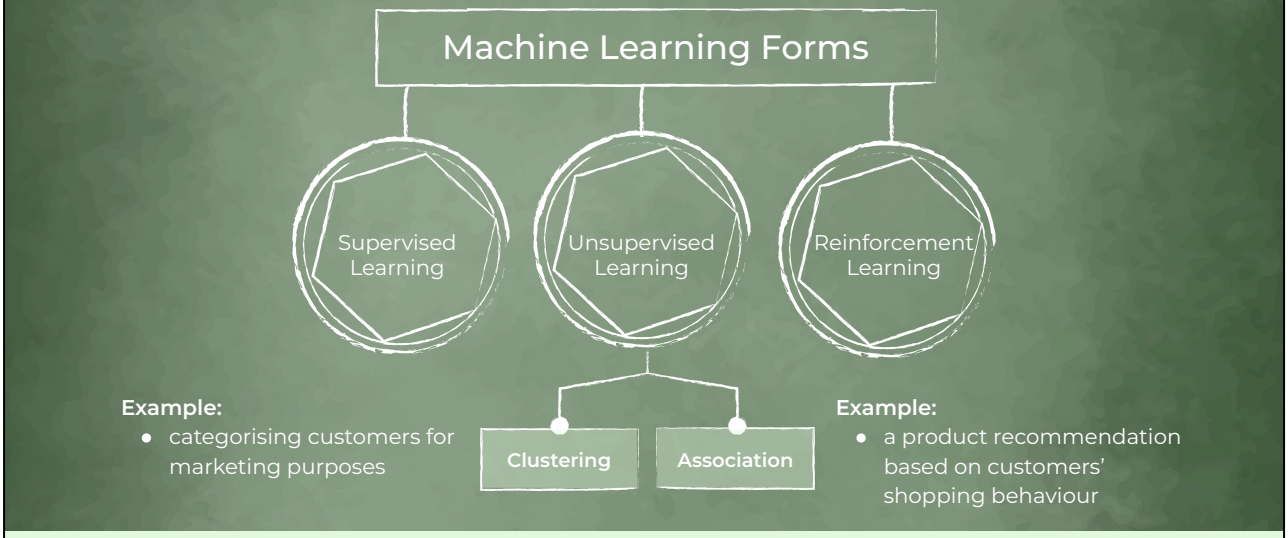
3.1.2 Unsupervised Learning



- **Goal:** To discover hidden patterns, structures, or relationships in the data
- **Training phase:** There is no predetermined correct answer or target variable to guide the learning process. An ML model is created from the unlabelled data used to deduce patterns in the input data
- **Testing/prediction phase:** A new data set is applied to the trained model to predict the output. The model is deployed once the output precision level is satisfactory

Unsupervised learning: Training an ML model from input data using an unlabelled dataset

Unsupervised learning is a type of machine learning where an algorithm learns patterns and structures in unlabelled data without any specific output labels. Unlike supervised learning, there is no predetermined correct answer or target variable to guide the learning process. The goal of unsupervised learning is to discover hidden patterns, structures, or relationships in the data. It allows the algorithm to explore the data and identify similarities, clusters, or patterns that may not be immediately apparent.

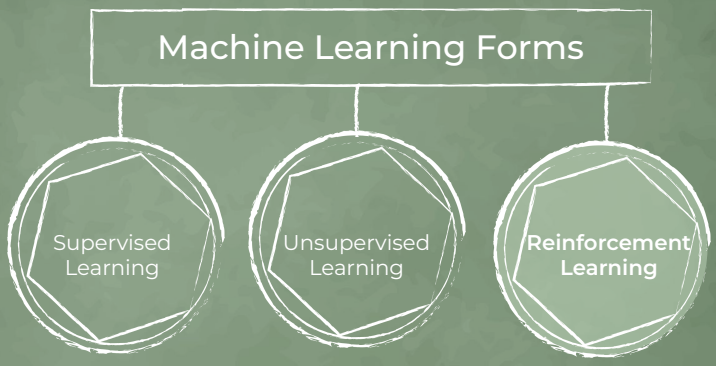


Clustering: A type of ML function that groups similar data points together
Association: An unsupervised learning technique that identifies relationships and dependencies between samples

Unsupervised learning helps us with the following issues:

- **clustering,** which is when the problem requires the identification of similarities in input data points that allows them to be grouped based on common characteristics or attributes (for example, categorising customers for marketing purposes).
- **association,** which is when the problem requires interesting relationships or dependencies to be identified among data attributes (for example, a product recommendation based on customers' shopping behaviour).

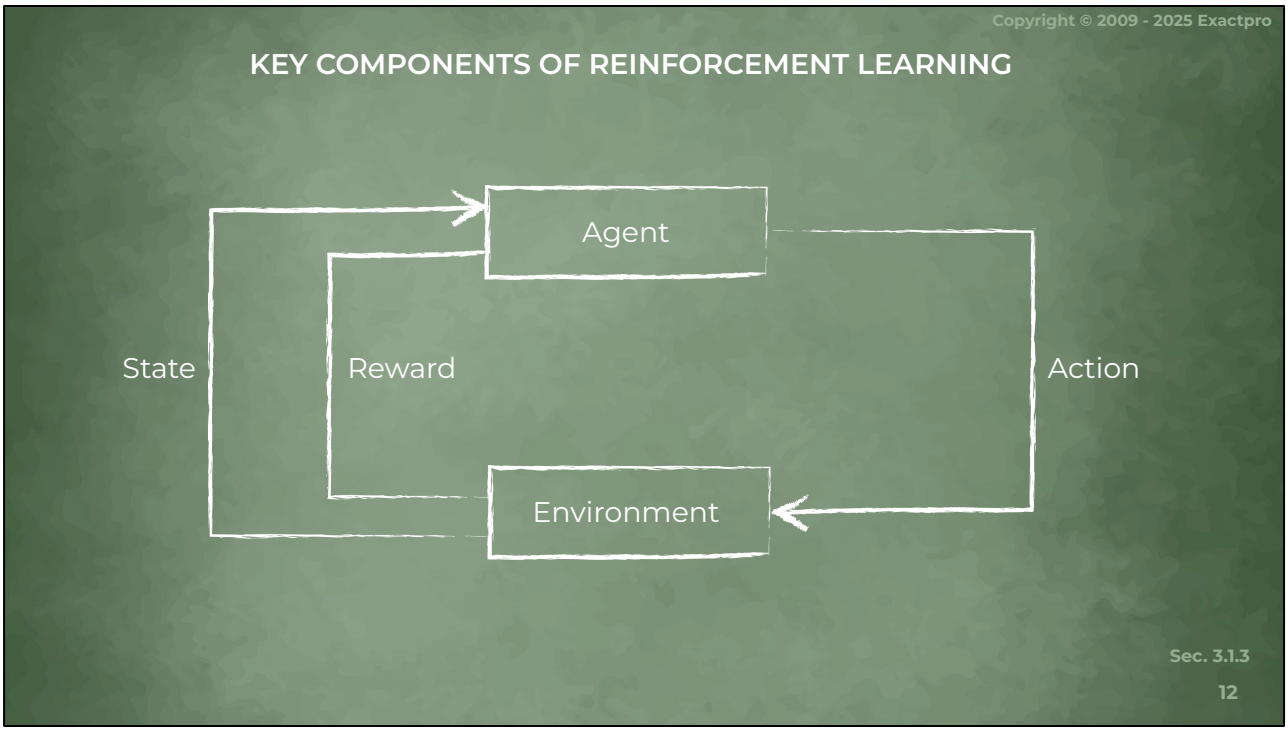
3.1.3 Reinforcement Learning



- **Goal:** For training agents, to make sequential decisions in an environment to maximise a cumulative reward
- **Training phase:** No training data. The system learns through a trial-and-error process by interacting with the environment
- **Examples:** Robotics, autonomous vehicles, chatbots

Reinforcement learning: The activity of building an ML model using a process of trial and reward to achieve an objective

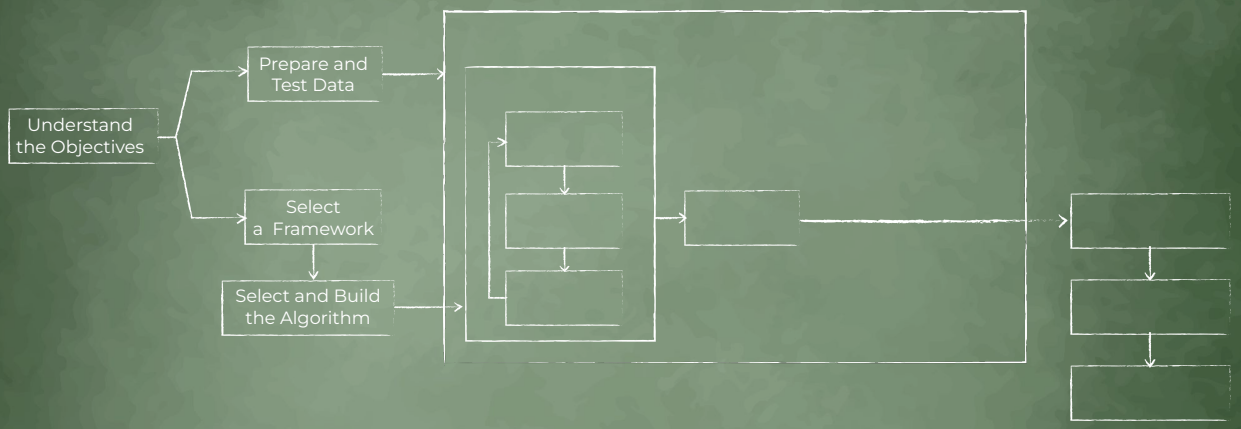
Reinforcement learning is a type of machine learning that focuses on training agents to make sequential decisions in an environment to maximise a cumulative reward, because the agent is rewarded when it makes a correct decision and penalised when it makes an incorrect decision. Unlike supervised learning and unsupervised learning, reinforcement learning does not rely on labelling or the absence of thereof. Instead, the agent learns through a trial-and-error process by interacting with the environment.
Examples of reinforcement learning could be found in robotics, autonomous vehicles, and chatbots.



Key components of reinforcement learning include:

1. **Agent**: The learner that interacts with the environment. It takes actions based on its current state and receives feedback in the form of rewards.
2. **Environment**: The external system with which the agent interacts. It provides feedback to the agent and transitions to new states based on the agent's actions.
3. **State**: The current representation of the environment at a particular time. It gives us information about how the agent understands the environment, which helps it make decisions.
4. **Action**: The choices available to the agent at each state. The agent selects actions based on its policy and the current state.
5. **Reward**: The feedback from the environment after the agent takes an action. It indicates the desirability of the agent's actions and is used to guide the learning process.

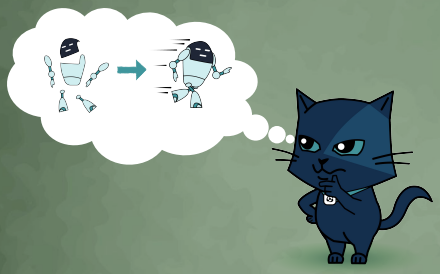
3.2 ML Workflow



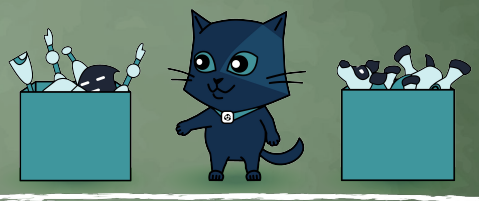
ML workflow: A sequence of activities used to manage the development and deployment of an ML model
ML framework: A tool or library that supports the creation of an ML model
ML algorithm: An algorithm used to create an ML model from a training dataset
Data preparation: The activities of data acquisition, data pre-processing and feature engineering in the ML workflow

Let’s go over the workflow of machine learning. First of all, we need to **understand the objectives.** All stakeholders need to agree on the purpose and acceptance criteria of the ML model to ensure alignment with business priorities. Second of all, a suitable AI development **framework should be selected** based on the aforementioned objectives, acceptance criteria, and business priorities. Then we **select and build an ML algorithm**. It may be coded manually, but it is often retrieved from a library of pre-written code. The algorithm is then compiled to prepare it for training the model, if needed. After that, **data preparation and testing** is required.

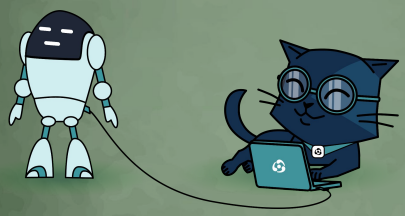
UNDERSTAND THE OBJECTIVES



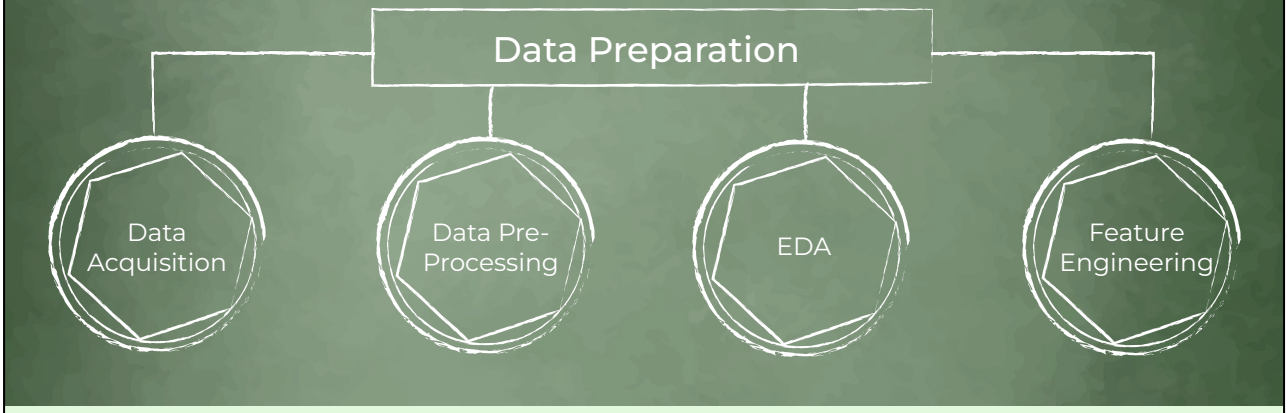
SELECT A FRAMEWORK



SELECT AND BUILD THE ALGORITHM

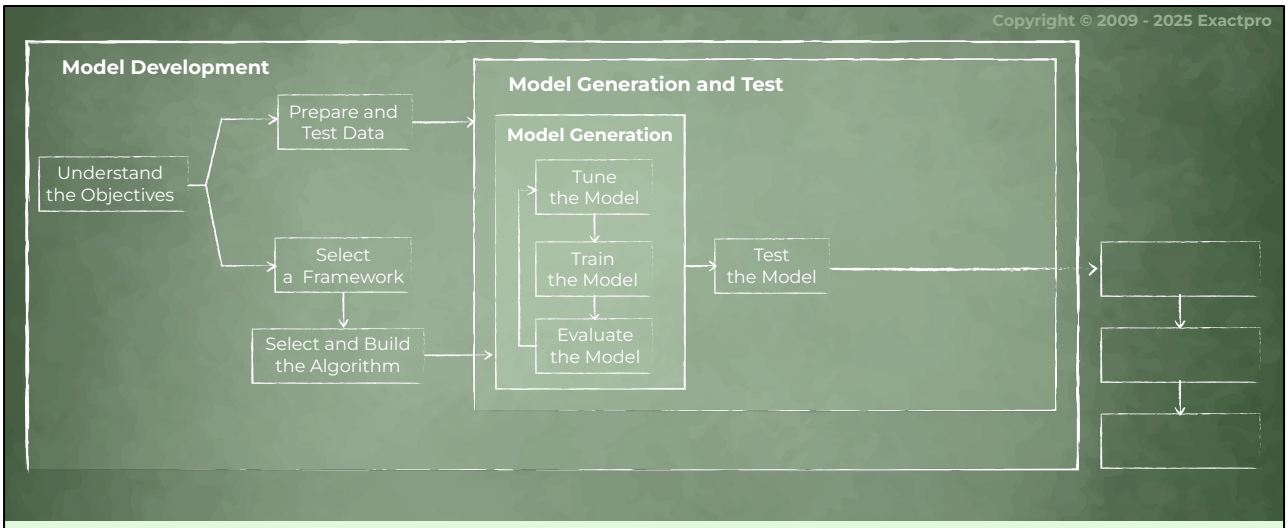


PREPARE TEST DATA



Data acquisition: The activity of acquiring data relevant to the business problem to be solved by an ML model
Data pre-processing: The activities of data cleaning, data transformation, data augmentation, and data sampling in the ML workflow
Exploratory data analysis (EDA): The interactive, hypothesis-driven and visual exploration of data used to support all data preparation activities (data acquisition, data pre-processing and feature engineering)
Feature engineering: The activity in which those attributes in the raw data that best represent the underlying relationships that should appear in the ML model are identified for use in the training data

After that, test **data preparation** is required. This process consists of **data acquisition**, **data pre-processing**, **exploratory data analysis** (EDA) and **feature engineering**.

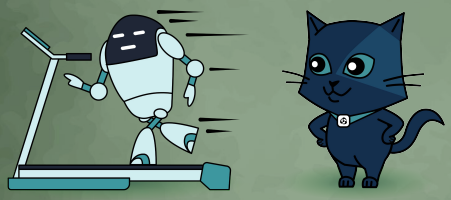


ML model training: The process of applying the ML algorithm to the training dataset to create an ML model
ML model evaluation: The process of comparing achieved ML functional performance metrics with required criteria and those of other ML models
ML model tuning: The process of testing hyperparameters to achieve optimum performance
ML model testing: The process where the performance of a fully trained ML model is evaluated on an independent testing dataset

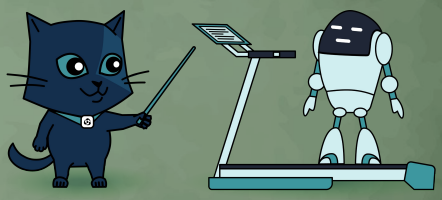
The data used by the model will be based on the set objectives and used by all the activities in the “model generation and test” phase. For example, if we have a real-time trading system, we will require market data which must be representative. It is also possible to use pre-gathered datasets for the initial training. Otherwise, raw data typically will demand some pre-processing, feature engineering, and testing. The ML algorithm uses data to **train the model**. Some algorithms, such as those generating a neural network, have to read the **training dataset** several times. Each iteration is called an **epoch**. Parameters defining the structure and controlling the training are passed to the model and they are called model **hyperparameters**. Then we **evaluate the model.** The ML functional performance metrics are assessed, using the **validation dataset**, thus trying to improve the model. After that we **tune the model.** The results from the model evaluation are used to adjust the settings to fit the data and improve its performance. The model may also be tuned by adjusting hyperparameters, where the training activity is modified by changing the number of training steps, or the amount of data used for training, or by updating attributes of the model. Model evaluation and tuning should resemble a carefully conducted scientific experiment with controlled conditions and clear documentation. In practice, several models are typically created and trained

using different algorithms, and then the best one is chosen. The three activities of training, evaluation and tuning help to generate the model. Then we **test the model.** Once a model has been generated, it should be tested against an independent test dataset to ensure that the agreed ML functional performance criteria are met. If the performance of the model with independent data is significantly lower than what it was during evaluation, it may be necessary to select a different model. In addition to functional performance tests, non-functional tests, such as estimating the time to train the model, and the time and resource usage taken to provide a prediction, also need to be run. Typically, these tests are performed by data scientists, but testers with sufficient knowledge of the domain can help here as well.
All the considered ML activities define the **model development** phase.

TRAIN THE MODEL

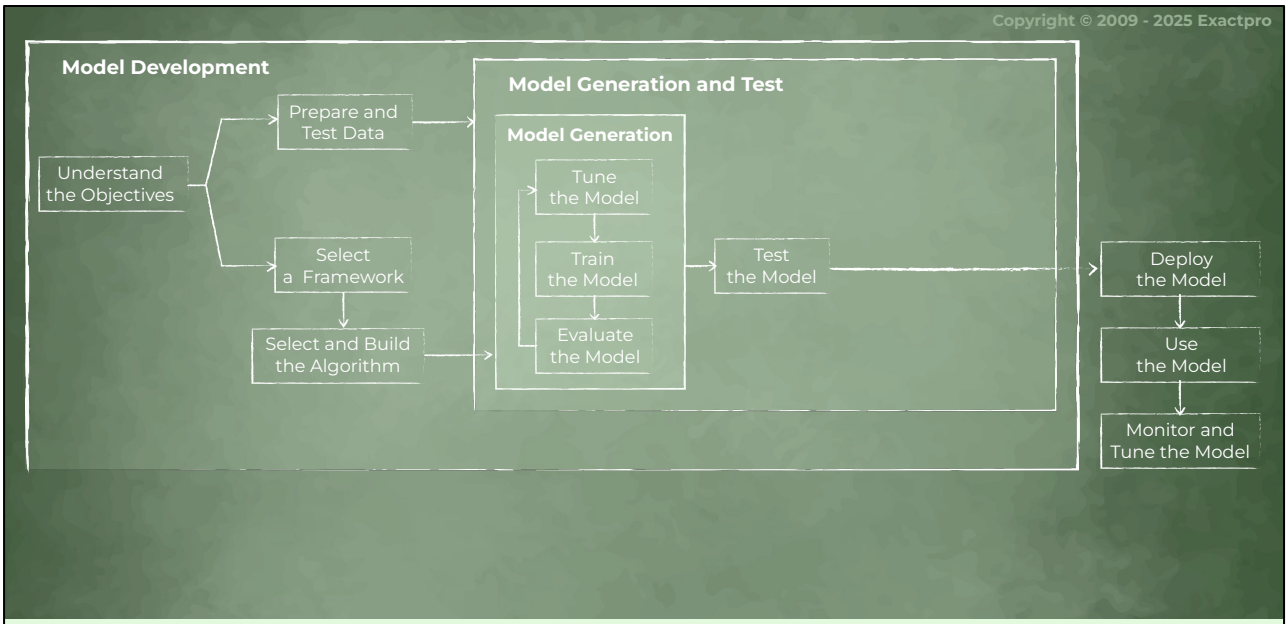


EVALUATE THE MODEL



TUNE THE MODEL

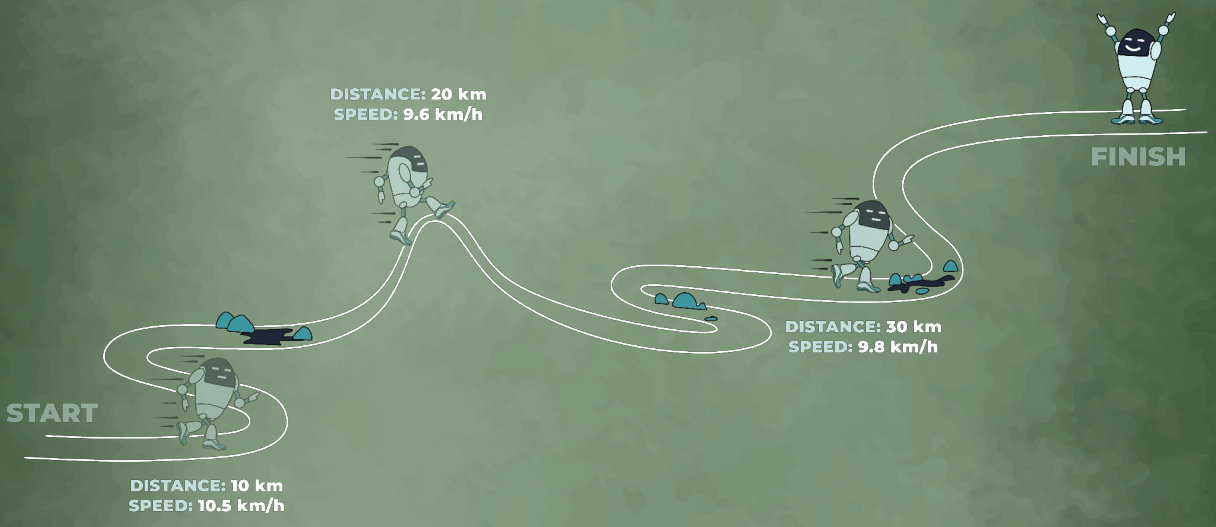
TEST THE MODEL



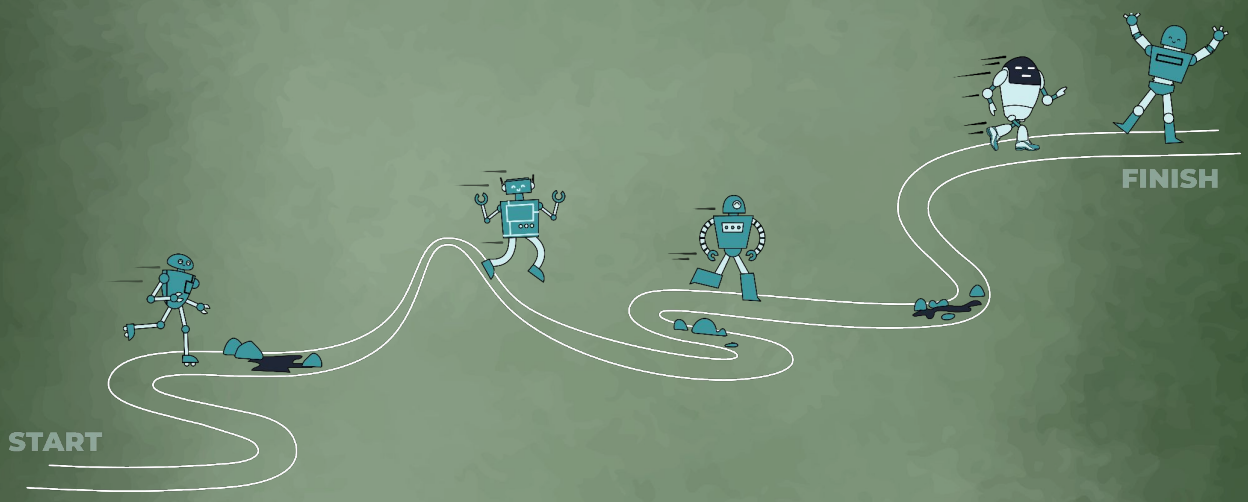
ML model deployment: The process of placing a finished ML model into a live environment where it can be used for its intended purpose

The next step is **model deployment.** Once development is complete, the tuned model typically needs to be re-engineered for deployment along with its related resources, including the relevant data pipeline. This is normally achieved through the framework. Targets might include embedded systems and the cloud, where the model can be accessed via a web API. After that we **use the model**. Once deployed, the model becomes a part of a larger AI-based system. It may perform scheduled batch predictions at specified time intervals or run on request in real time. And lastly, we **monitor and tune the model**. While the model is being used, its conditions may evolve and the model may drift away from what's intended. To ensure that any drift is identified and managed, the operational model should be regularly evaluated against its acceptance criteria. It may require the adjustment of the model's settings to address the drift or re-training with new updated data to create a more accurate and robust model. The new model may then be compared against the existing model, for example, using a form of A/B testing.

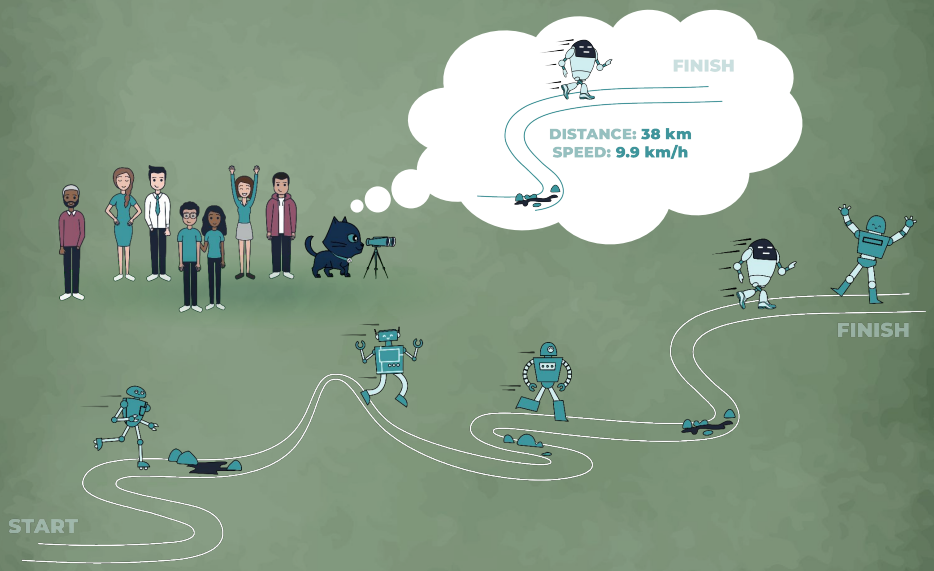
DEPLOY THE MODEL



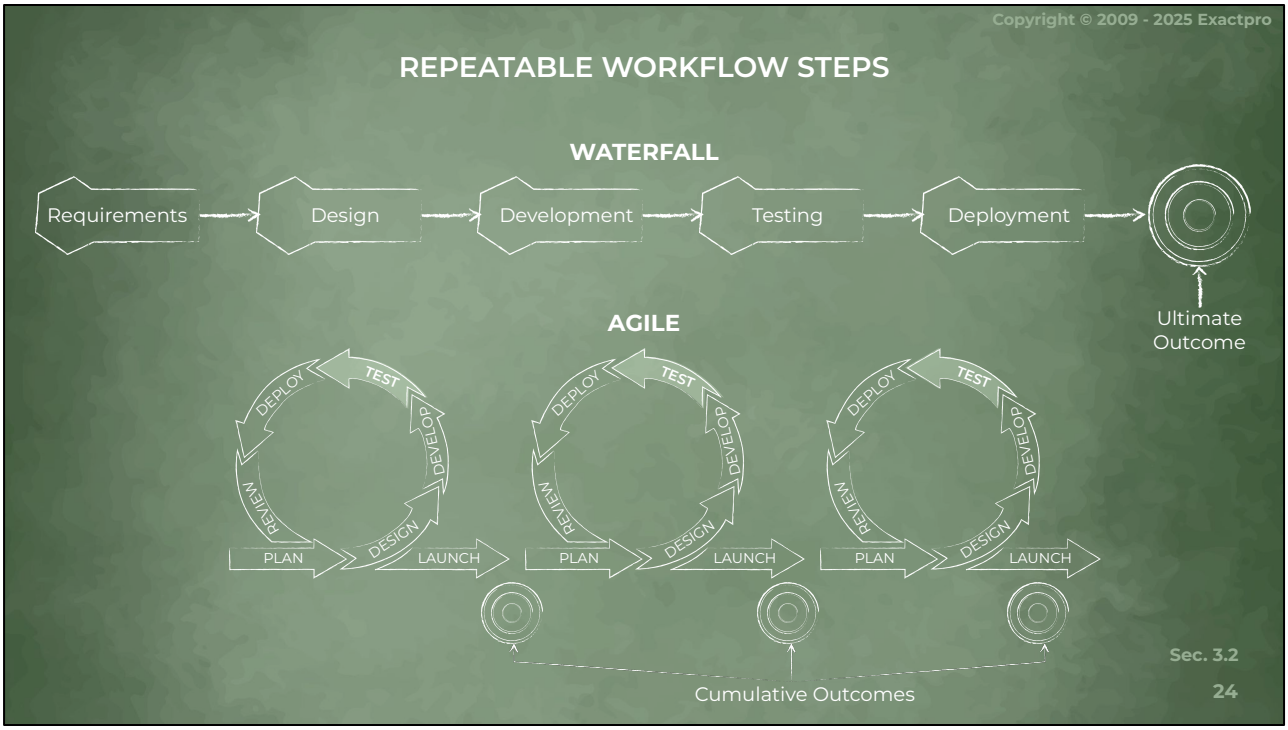
USE THE MODEL



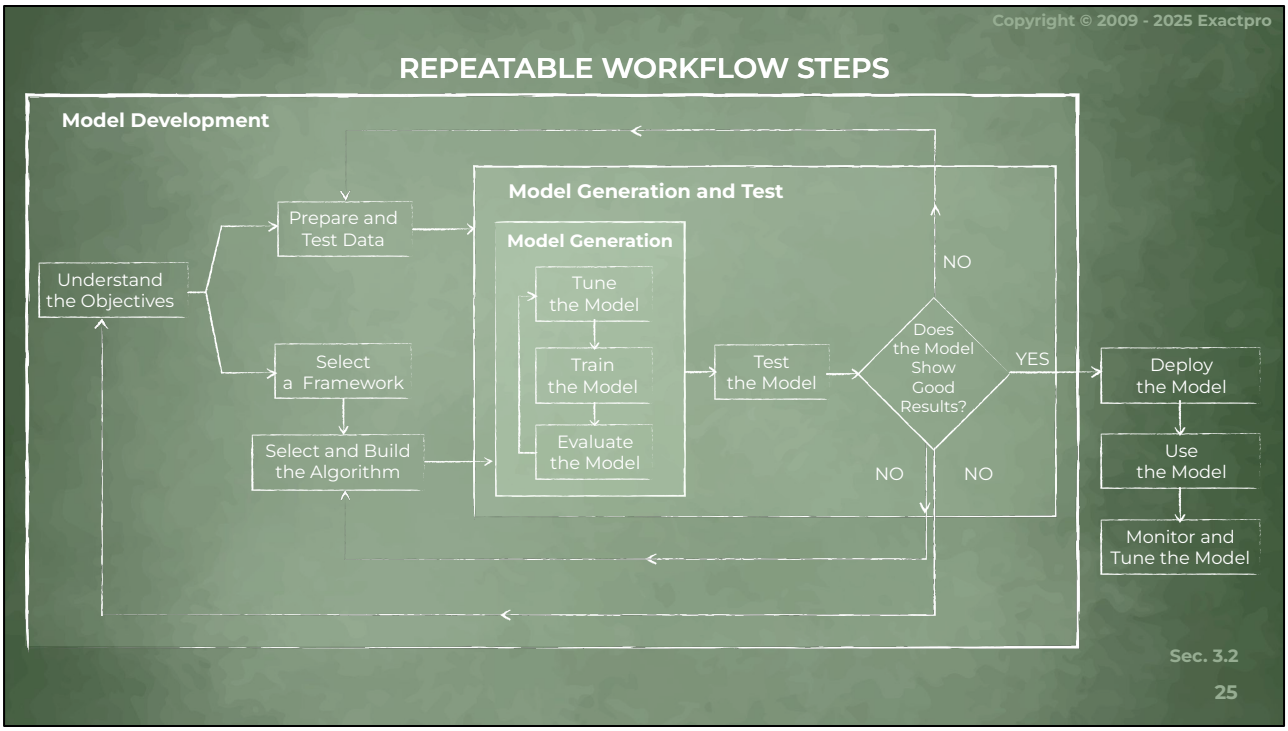
MONITOR THE MODEL



TUNE THE MODEL

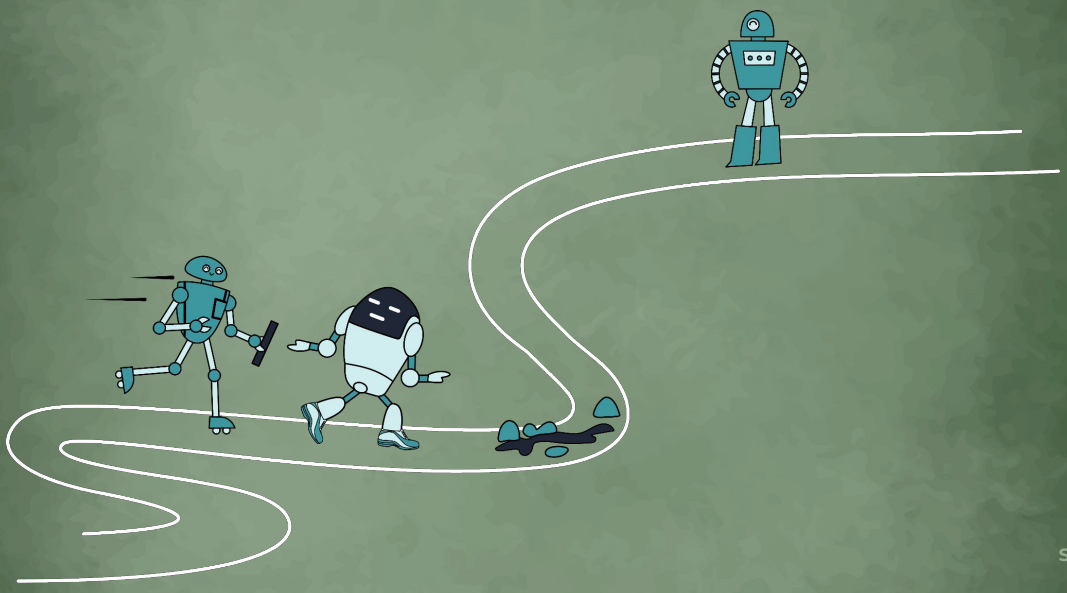


The ML workflow shown here is a logical sequence just like the **waterfall** model. In practice, the workflow steps are repeated **iteratively**, just like with iterative lifecycle models such as **Agile** when we receive feedback and generate preliminary cumulative outcomes. Usually, the key point here lies in the **testing phase** of the model.

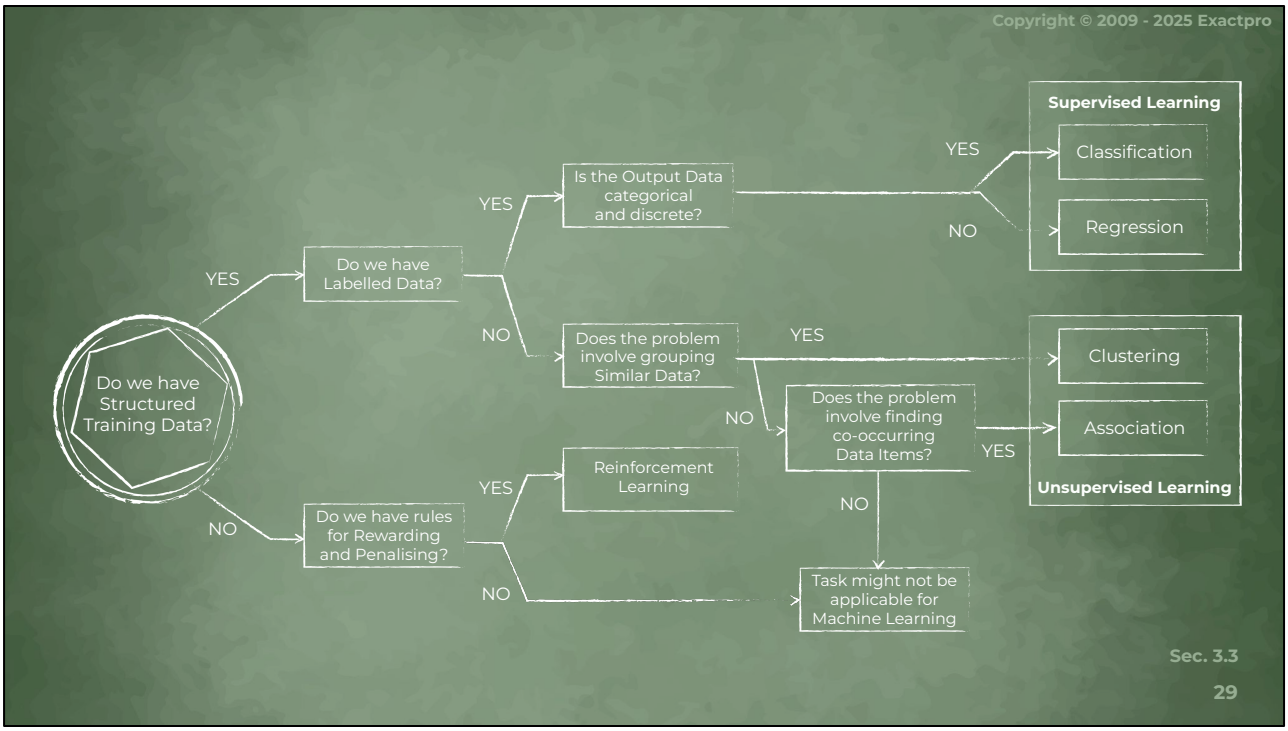


If the model shows **low or insufficient results at this stage**, a decision may be made to change the model (return to the **"select and build an ML algorithm"** step). It may also be that the data was originally not sufficiently examined and prepared, and not all of the model features useful for learning have been selected. If so the specialist will return to the "data preparation" step and the whole process will be repeated and this may happen several times. There could be a possibility that the objectives have been misunderstood, then the specialist will return to the “understand the objectives” step. Please mind that the phases shown here do not include the integration of the ML model with the non-ML parts of the overall system. If the model is part of a larger AI-based system, it will need to be integrated into this system prior to deployment. In this case, integration, system and acceptance test levels may be performed.

INTEGRATION



3.3 Selecting a Form of ML



Let's look into how one can select an appropriate ML approach, based on the nature of the task at hand:

- The first question we need to ask ourselves is do we have structured training data? If not, the next question is do we have **rules for rewarding and penalising**? If our answer is again no, our task might not be applicable for machine learning at all. But if those rules do exist then **reinforcement learning** may be applied.
- Let's go back to the question about the structured data. Now, if our answer is yes, then it takes us to another question. Do we have labelled data? If the answer is no, we have to ask ourselves if the problem involves grouping similar data. If the answer is yes, it may be **clustering**. If the answer is no, then we ask if the problem involves **finding co-occurring data items**, which may be an **association** in case of a positive answer, otherwise our task might not be applicable for machine learning in case of a negative answer**.** Both clustering and association point to **unsupervised learning**.
- Let's go back to the question about **labelled data.** If our answer is yes, then we ask a question if the output data is categorical. If the output is discrete and categorical, it may be **classified**. If the output is numeric

- and continuous in nature, it may be **regression**. Regression and classification suggest there is an output label which means it may be **supervised learning.**

EXERCISE 1

PROJECT GOAL

Sort emails as *spam/not spam* based on their content and other relevant features

TRAINING DATA

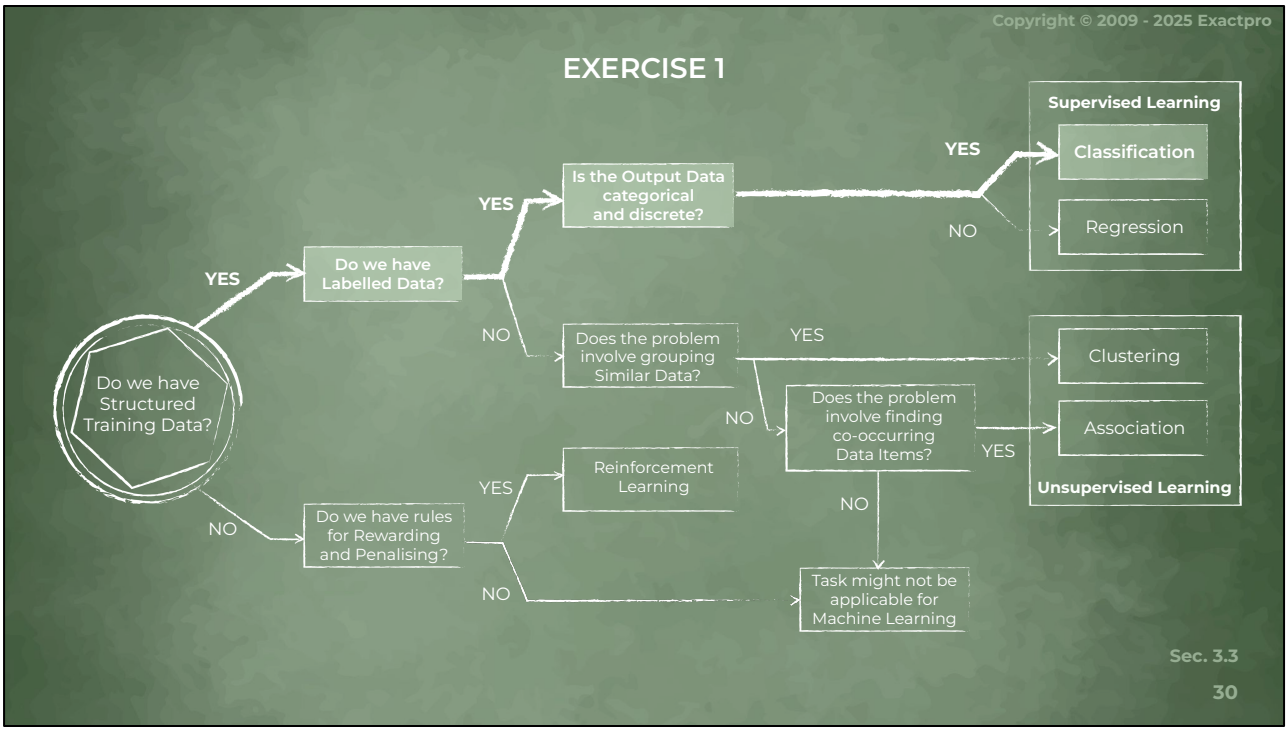
Dataset of emails, each labelled as spam or not spam, along with features extracted from the email text, such as the presence of certain keywords, the length of the email, and the number of exclamation marks

	A	B	C	D	E
1	**EMAIL #**	**Keywords**	**Length (in words)**	**Exclamation Marks**	**Class**
2	1	discount, sale	150	2	spam
3	2	meeting, agenda	200	0	not spam
4	3	free, win	100	3	spam
5	4	report, analysis	180	0	not spam

QUESTION

What is an appropriate ML approach in this situation?

Now let's review the application of this workflow.
Suppose we need to sort emails as either spam or not spam based on their content and other relevant features.
For the training purposes we have a dataset of emails, each labelled as spam or not spam, along with features extracted from the email text, such as the presence of certain keywords, the length of the email, and the number of exclamation marks. What is an appropriate ML approach that we can use to build our model?



Let's look at our workflow diagram and answer several questions. Do we have structured training data? Yes. Do we have labelled data? Yes. Is the output data categorical and discrete? Yes. Then we have **classification** as an appropriate ML approach for the considered problem with the emails.

EXERCISE 1

OBJECTIVE

Build a **CLASSIFICATION MODEL** that can accurately predict whether a new email is spam or not spam

LEARNING ALGORITHM

Logistic regression or support vector machines (SVM)

MODEL TRAINING

On the labelled dataset to learn the patterns and relationships between the input features and the target class labels (spam or not spam). The ML model finds the decision boundary that separates the two classes

MODEL PREDICTIONS

The class of new, unseen emails. For instance, given the features of a new email:

EMAIL #	Keywords	Length (in words)	Exclamation Marks
1034	offer, buy	120	1

the classification model can predict whether it is spam or not spam

The objective is to build a classification model that can accurately predict whether a new email is spam or not. Using a classification algorithm like logistic regression or support vector machines (SVM), we can train a model on the labelled dataset to learn the patterns and relationships between the input features and the target class labels (spam or not spam). By training a classification model on this data, the model will learn the patterns that find spam emails based on the input features. It will find the decision boundary that separates the two classes. Once the model is trained, it can be used to predict the class of new, unseen emails. For instance, given the features of a new email (e.g., keywords: "offer," "buy," length: 120 words, exclamation marks: 1), the classification model can predict whether it is spam or not spam.

EXERCISE 2

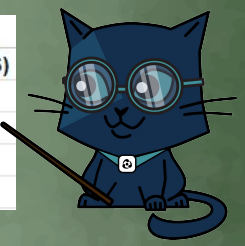
PROJECT GOAL

Predict house prices based on various features such as size, number of rooms, location, and other relevant factors

TRAINING DATA

Dataset of houses with their corresponding features and their actual sale prices

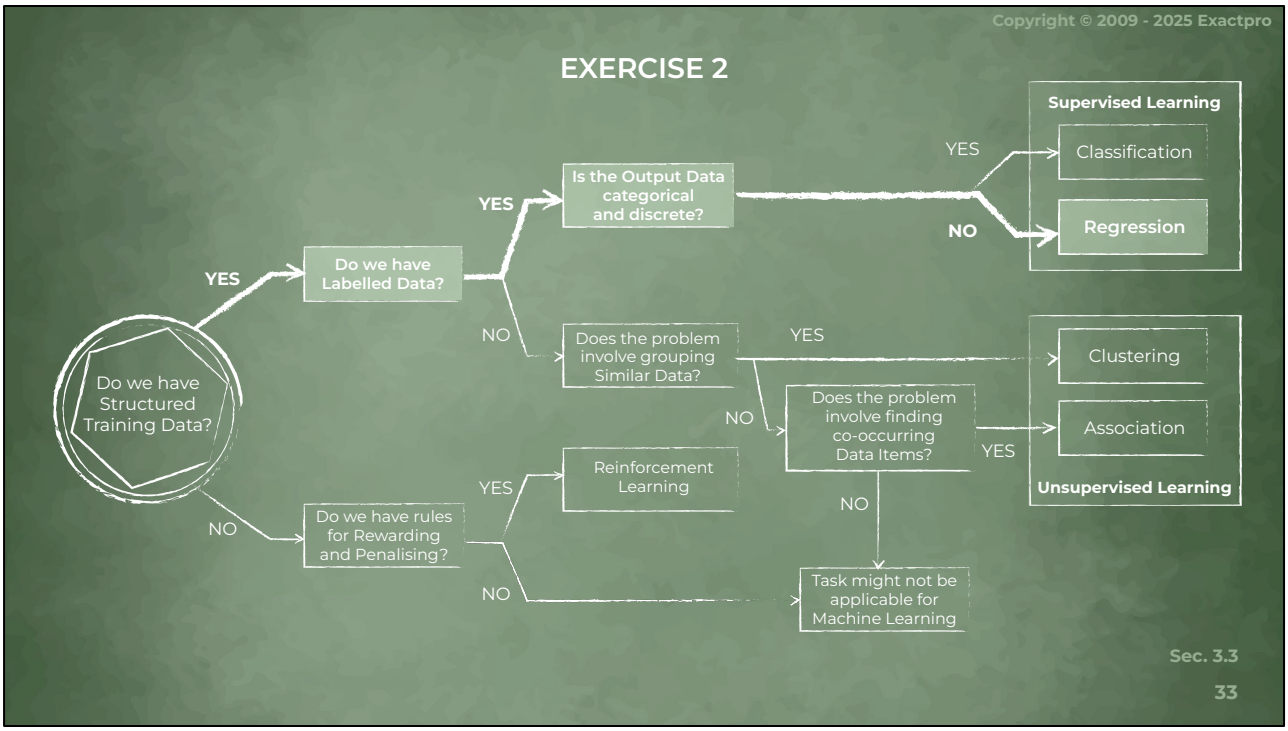
	A	B	C	D	E
1	**House #**	**Size (m^2)**	**Number of Rooms**	**Location**	**Sale Price ($)**
2	1	140	3	Suburban	250,000
3	2	185	4	Urban	350,000
4	3	170	3	Suburban	300,000
5	4	200	5	Urban	400,000



QUESTION

What is an appropriate ML approach in this situation?

Let’s go to the next task. We have to predict house prices based on various features such as size, number of rooms, location, and other relevant factors. Suppose we have a simplified scenario, when a dataset of houses with their corresponding features and their actual sale prices is available for the training purposes.This dataset might look like this…
What is an appropriate ML approach to build our model?



Let's look at the workflow diagram and answer the questions one by one. Do we have structured training data? Yes. Do we have labelled data? Yes. Is the output data categorical and discrete? No. It is numerical. Hence we need to use **regression** as an appropriate ML approach for our price prediction problem.

EXERCISE 2

OBJECTIVE

Develop a **REGRESSION MODEL** that can accurately predict the sale price of a house based on its features

LEARNING ALGORITHM

Linear regression

MODEL TRAINING

On the labelled dataset to learn the patterns and relationships between the input features and the target sale prices

MODEL PREDICTIONS

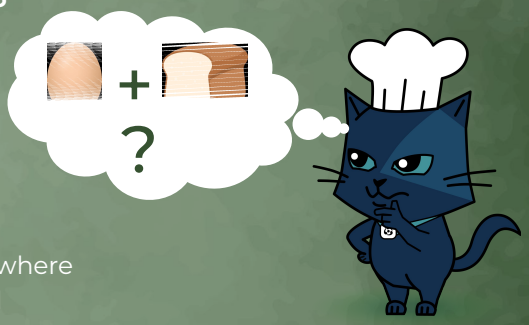
The sale price of new houses based on their features: size, number of rooms and location. For instance, given the features of a new house:

House #	Size (m^2)	Number of Rooms	Location
10485	175	3	Urban

the regression model might predict a sale price of around $320,000

The objective is to develop a regression model that can accurately predict the sale price of a house based on its features.
By using a regression algorithm such as linear regression, we can train a model on the labelled dataset to learn the patterns and relationships between the input features and the target sale prices. By training a regression model on this data, the ML model will learn to predict house prices based on the given features. The trained model can then be used to predict the sale price of new houses based on their features: size, number of rooms and location. For example, if we have a new house with a size of 175 m2, 3 rooms, and located in an urban area, the regression model might predict a sale price of around $320,000.

EXERCISE 3



PROJECT GOAL

The market basket analysis, which aims to discover relationships between items that are frequently purchased together in a transactional dataset

TRAINING DATA

Dataset of customer transactions in a grocery store, where each transaction consists of a list of items purchased

	A	B	C	D	E	F	G
1	**Transaction ID**	**Bread**	**Milk**	**Eggs**	**Diapers**	**Beer**	**Chips**
2	1	1	1	1	0	0	0
3	2	1	0	0	1	1	0
4	3	0	1	0	1	1	1
5	4	1	1	0	1	1	0

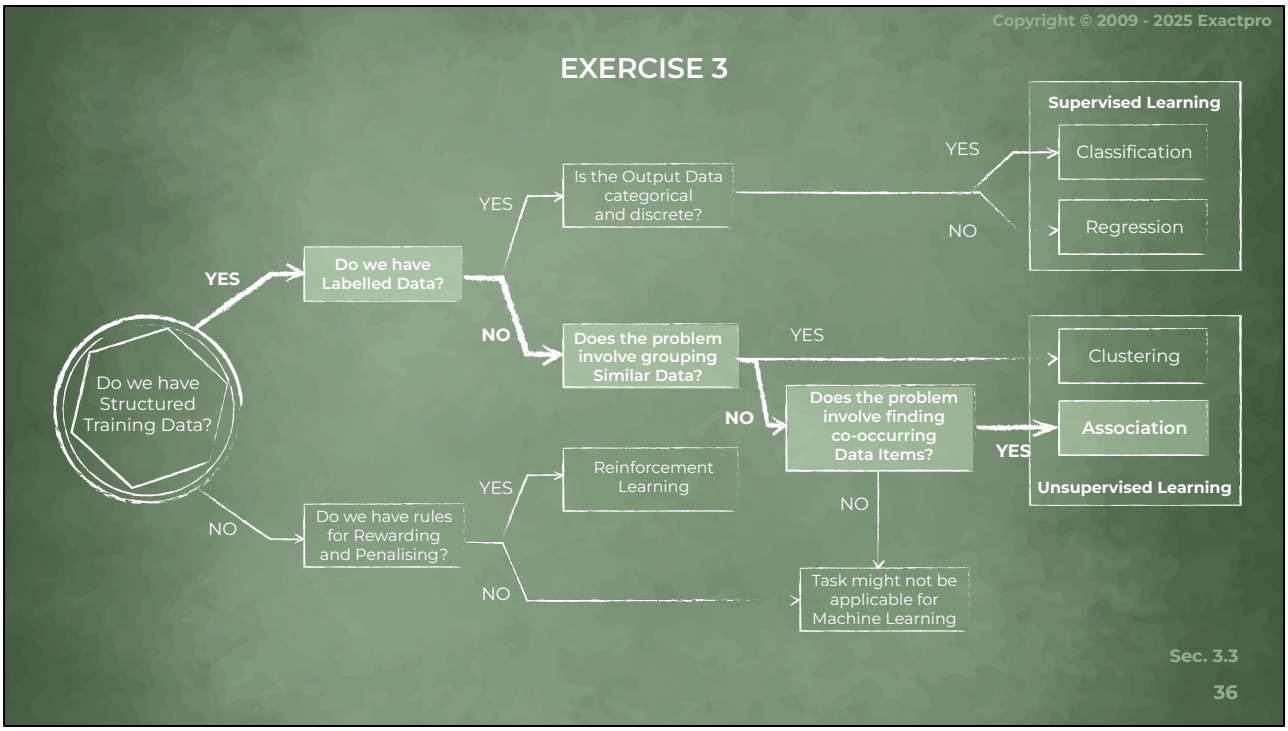
QUESTION

What is an appropriate ML approach in this situation?

Let’s go to the next exercise. Say, we need to conduct the market basket analysis, which aims to discover relationships or associations between items that are frequently purchased together in a transactional dataset. We will consider a simplified scenario again. Suppose, we have a dataset of customer transactions in a grocery store, where each transaction consists of a list of items purchased. The value 1 means the presence of the item in the corresponding transaction, and the value 0 represents the absence of an item in that transaction. For example, the dataset might contain transactions like:

- Transaction 1: {Bread, Milk, Eggs}
- Transaction 2: {Bread, Diapers, Beer}
- Transaction 3: {Milk, Diapers, Beer, Chips}
- Transaction 4: {Bread, Milk, Diapers, Beer}

What is an appropriate ML approach in this situation?



Let's look again at our workflow diagram and answer the questions. Do we have structured training data? Yes. Do we have labelled data? No. Does the problem involve grouping similar data? No.
Does the problem involve finding co-occurring data items? Yes. Hence **association** may be applied as an appropriate ML approach for the market basket analysis.

EXERCISE 3

OBJECTIVE

Identify associations between items in customer transactions in a grocery store, such as which items tend to be bought together

LEARNING ALGORITHM

Association rule mining algorithm, such as the Apriori algorithm

MODEL TRAINING

On the unlabelled dataset to learn the patterns and relationships between items based on their co-occurrence in transactions, to discover frequent itemsets and association rules

$$\text{Support (A, B)} = \frac{\text{number of transactions with A and B}}{\text{total number of transactions}}$$

where A and B are separate itemsets that occur at the same time in a transaction

The objective is to identify associations between items in customer transactions in a grocery store, such as which items tend to be bought together.
In this case, we can apply an association rule mining algorithm, such as the Apriori algorithm, to discover frequent itemsets and association rules. The algorithm will identify patterns and relationships between items based on their co-occurrence in transactions.
Support is an indication of how frequently the itemset appears in the dataset. The support of the itemset containing both items A and B equals the number of transactions in the dataset that contain the itemset A and B divided by the total number of transactions in the dataset. In simpler terms, the support of an itemset is the proportion of transactions that contain that specific itemset out of all transactions in the dataset.

EXERCISE 3

- The higher the support, the more frequently the itemset occurs, and it is considered more significant
- Those itemsets whose support exceeds a predefined minimum support threshold are considered frequent
- Frequent itemsets are used to generate association rules

Minimum Support Threshold = 50%

if Antecedent then Consequent	Support (%)
if buy bread, then buy milk	75
if buy milk, then buy diapers	50
if buy bread, then buy beer	50

Association Rules

Association Rule	Support (%)
Bread => Milk	75
Milk => Bread	75
Milk => Diapers	50
Diapers => Milk	50
Bread => Beer	50
Beer => Bread	50

MODEL PREDICTIONS

Items which might be bought together with items in a new transaction of a customer in a grocery store

The insights into the relationships between items might be used for various purposes, such as product placement, cross-selling, or targeted marketing

The higher the support, the more frequently the itemset occurs. During the Apriori algorithm execution, the support is calculated for candidate itemsets, and those itemsets whose support exceeds a predefined minimum support threshold are considered frequent. Frequent itemsets are then used to generate association rules.Let's say we set a minimum support threshold of 50% (itemset must appear in at least 50% of transactions to be “frequent”). Applying the algorithm, we might discover the following associations:
Frequent Item sets: {Bread, Milk}: 75% (appears in 3 out of 4 transactions), {Milk, Diapers}: 50%, {Bread, Beer}: 50%
Association Rules:{Bread} => {Milk}: 75%, {Milk} => {Bread}: 75%, {Milk} => {Diapers}: 50%, {Diapers} => {Milk}: 50%, {Bread} => {Beer}: 50%, {Beer} => {Bread}: 50%
These results indicate that bread and milk are frequently purchased together, as they appear together in 75% of transactions. The same goes for all 50% outcomes.The trained association model can be used to predict which items might be bought together with items in a new transaction of a customer in a grocery store.The insights into the relationships between items might be used for various purposes, such as product placement, cross-selling, or targeted marketing.

EXERCISE 4

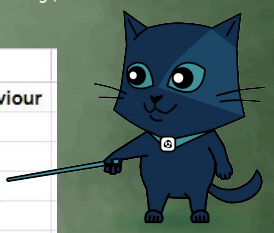
PROJECT GOAL

Create an algorithm that helps grouping customers based on their characteristics and past behaviour

TRAINING DATA

Dataset containing information about customers in an e-commerce business. Each customer is described by features such as age, income, browsing history, and purchase behaviour

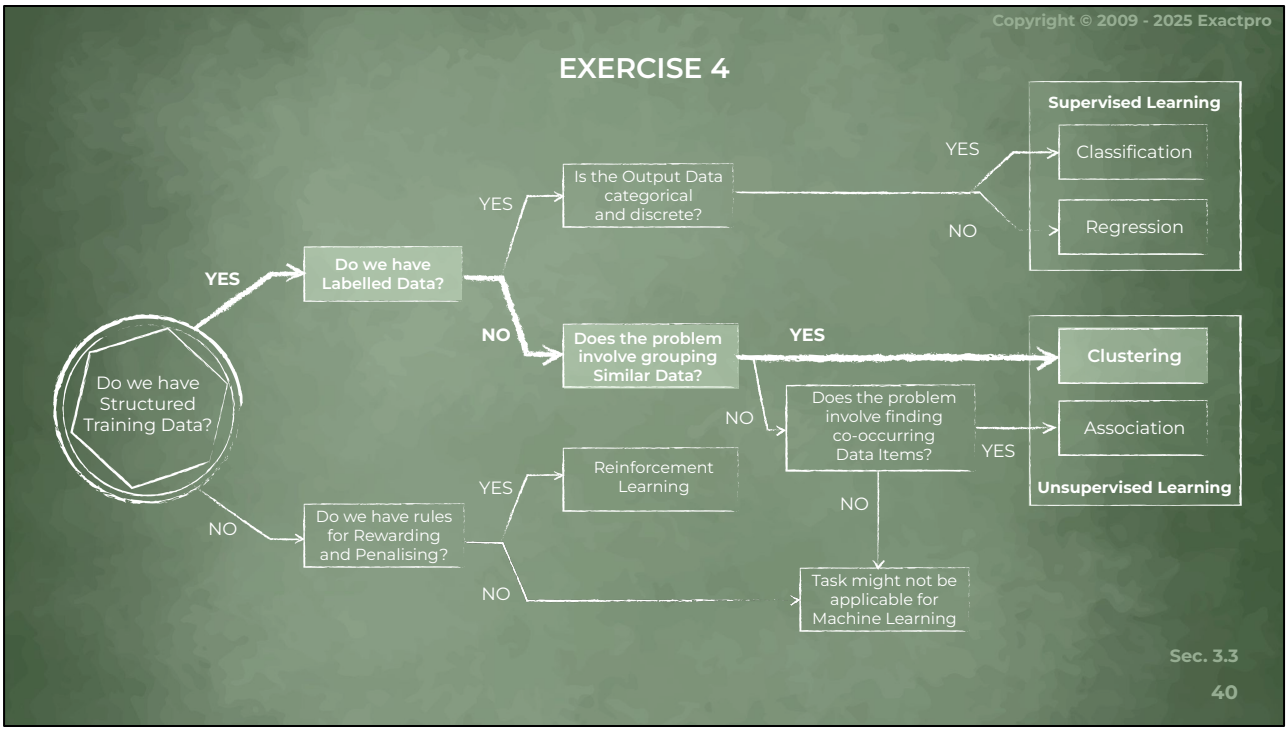
	A	B	C	D	E
1	**Customer #**	**Age**	**Income ($)**	**Browsing History (minutes)**	**Purchase Behaviour**
2	1	30	50,000	5	Frequent
3	2	45	70,000	15	Moderate
4	3	25	35,000	3	Occasional
5	4	40	60,000	10	Frequent



QUESTION

What is an appropriate ML approach in this situation?

The next project example is to create an algorithm that helps grouping customers based on their characteristics and past behaviour.
Let's consider a simplified scenario. Suppose you have a dataset containing information about customers in an e-commerce business. Each customer is described by features such as age, income, browsing history, and purchase behaviour. What is an appropriate ML approach that we can use to build our model?



Let's look at the workflow diagram and answer several questions. Do we have structured training data? Yes. Do we have labelled data? No. Does the problem involve grouping similar data? Yes. Then we have **clustering** as an appropriate ML approach for our problem of customer segmentation.

EXERCISE 4

OBJECTIVE

To **CLUSTER** the customers into distinct groups to gain insights and make targeted marketing strategies

LEARNING ALGORITHM

K-means

MODEL TRAINING

On the unlabelled dataset with unknown true values. The algorithm might group the customers based on similarities in their feature values into three clusters:

#1	Frequent shoppers with moderate to high income	**Customer 1, Customer 4**
#2	Moderate shoppers with higher income and longer browsing history	**Customer 2**
#3	Occasional shoppers with lower income and shorter browsing history	**Customer 3**

The objective is to cluster the customers into distinct groups to gain insights and make targeted marketing strategies.Using a clustering algorithm like K-means, we can perform customer segmentation. By training the model on the unlabelled customer dataset, the algorithm will automatically group customers based on similarities in their feature values. For example, let's say we choose to cluster the customers into three groups. Our ML clustering model might group the customers as follows:

- Cluster 1(frequent shoppers with moderate to high income): Customer 1, Customer 4
- Cluster 2 (moderate shoppers with higher income and longer browsing history): Customer 2
- Cluster 3 (occasional shoppers with lower income and shorter browsing history): Customer 3

EXERCISE 4

MODEL PREDICTIONS

New customer assignment to some customer segment

The insights into different customer segments help the business to tailor marketing strategies and promotions based on the characteristics of each cluster.

For example, Cluster 1 customers may be targeted with loyalty programs or personalised recommendations to encourage more frequent purchases, while Cluster 3 customers could be engaged with targeted discounts or incentives to increase their engagement.

One can interpret the clusters to gain insights into different customer segments, based on their income and various spending scores. Keep in mind that in a real-world scenario, you may have more features and a larger dataset for a more meaningful segmentation.

The trained model can then be used to assign new customers to some customer segments based on their features.
The resulting clusters provide insights into these different customer segments. The business can tailor marketing strategies and promotions based on the characteristics of each cluster. For example, Cluster 1 customers may be targeted with loyalty programs or personalised recommendations to encourage more frequent purchases, while Cluster 3 customers could be engaged with targeted discounts or incentives to increase their engagement.
One can interpret the clusters to gain insights into different customer segments, based on their income and various spending scores. Keep in mind that in a real-world scenario, you may have more features and a larger dataset for a more meaningful segmentation.

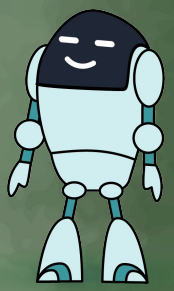
EXERCISE 5

PROJECT GOAL

Have an autonomous robot navigating through a maze to reach a target location by taking actions such as moving forward, turning left, turning right, or staying in place. The optimal sequence of actions should be found that lead the robot to the target location while avoiding obstacles

TRAINING DATA

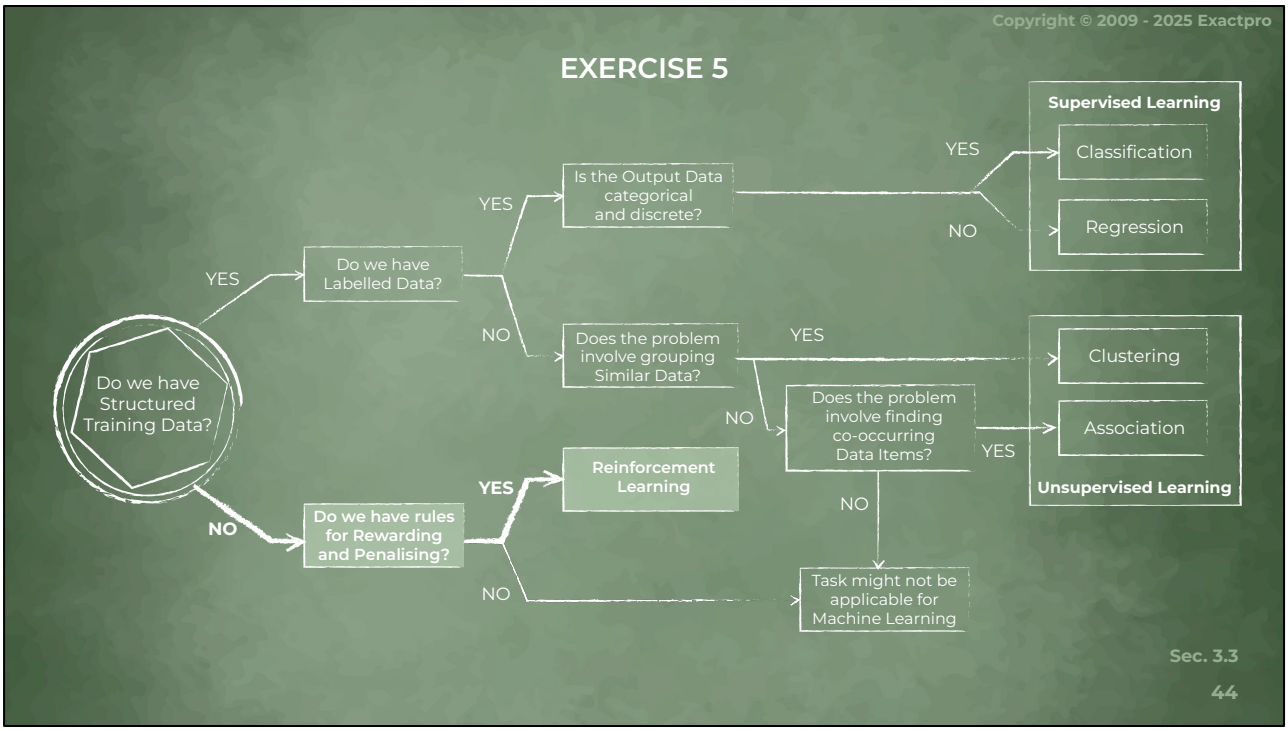
The robot starts with no prior knowledge of the maze or the optimal path to the target. The current configuration of the robot and its surroundings, which includes its position in the maze, is available. The environment provides the feedback signal to the agent based on its actions. The reward can be positive for reaching the target or negative for hitting an obstacle



QUESTION

What is an appropriate ML approach in this situation?

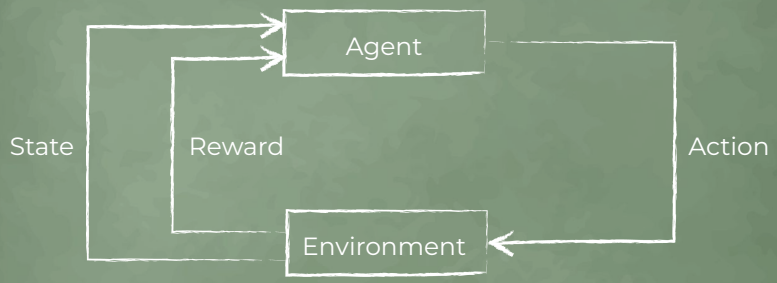
Let's say we need to have an autonomous robot that is able to learn how to navigate through a maze to reach a target location by taking actions such as moving forward, turning left, turning right, or staying in place. The optimal sequence of actions should be found that lead the robot to the target location while avoiding obstacles.
The robot starts with no prior knowledge of the maze or the optimal path to the target. The current configuration of the robot and its surroundings, which includes its position in the maze, is available. The environment provides the feedback signal to the agent based on its actions. The reward can be positive for reaching the target or negative for hitting an obstacle. What is an appropriate ML approach in this situation?



Let's look again at our workflow diagram and answer the questions. Do we have structured training data? No. Do we have rules for rewarding and penalising? Yes. So **reinforcement learning** can be applied.

EXERCISE 5

KEY COMPONENTS OF REINFORCEMENT LEARNING



Agent:	The autonomous robot that learns to navigate the maze
Environment:	The maze itself, which provides feedback to the agent based on its actions
State:	The current configuration of the robot and its surroundings, which includes its position in the maze
Action:	The possible moves the robot can take at any given state (up, down, left, right)
Reward:	The feedback signal provided by the environment to the agent based on its actions. The reward can be positive for reaching the target or negative for hitting an obstacle

As we have learned before, the reinforcement learning process needs the following key components: Agent, Environment, State, Action and Reward. Consider them separately within our problem. **Agent** is the autonomous robot that learns to navigate the maze. **Environment** is the maze itself, which provides feedback to the agent based on its actions. **State** is the current configuration of the robot and its surroundings, which includes its position in the maze. **Action** is the possible moves the robot can take at any given state (up, down, left, right). And finally, **Reward** is the feedback signal provided by the environment to the agent based on its actions. The reward can be positive for reaching the target or negative for hitting an obstacle.

EXERCISE 5

OBJECTIVE

Build a **REINFORCEMENT LEARNING MODEL** that can be used to teach the robot to navigate the maze efficiently

LEARNING ALGORITHM

Q-learning or policy gradients

MODEL TRAINING

STEP 1: The agent starts exploring the maze by taking random actions, at first, inefficiently and randomly

STEP 2: The agent progresses and receives feedback from the environment learning to associate certain actions with higher rewards

STEP 3: Through trial and error, the agent discovers the actions that lead to positive outcomes and avoids actions that result in negative rewards

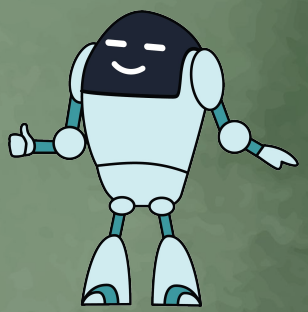
The project objective is to build a reinforcement learning model that can be used to teach the robot to navigate the maze efficiently.
To apply reinforcement learning, the agent starts exploring the maze by taking random actions. Initially, the agent's actions are likely to be inefficient and random. However, as the agent progresses and receives feedback from the environment, it learns to associate certain actions with higher rewards.
Through trial and error, the agent discovers the actions that lead to positive outcomes and avoids actions that result in negative rewards.

EXERCISE 5

STEP 4: The agent can employ a learning algorithm, such as Q-learning where it learns a quality value for each state-action pair. The Q-value represents the expected cumulative reward the agent will receive starting from that state, taking a particular action, and then following the optimal policy

STEP 5: The algorithm iteratively updates its Q-values based on the reward received from the environment

The agent learns from the rewards it receives to navigate the maze effectively avoiding obstacles through associating actions with the highest expected rewards in different states



To improve its performance, the agent employs a learning algorithm, such as Q-learning or policy gradients. This algorithm updates the agent's policy or value function based on the rewards received and the expected future rewards. The agent repeats steps 1 to 5 for multiple episodes, gradually improving its performance by learning from the rewards it receives. Over time, through repeated iterations, the agent learns to navigate the maze effectively, avoiding obstacles and reaching the goal more consistently. It learns the optimal strategy by associating actions with the highest expected rewards in different states.

3.4 Factors Involved in ML Algorithm Selection

FACTORS INVOLVED IN ML ALGORITHM SELECTION

- The required functionality
- The required quality characteristics, such as
  - accuracy
  - constraints on available memory
  - the speed of training (and retraining) the model
  - the speed of prediction
  - transparency, interpretability and explainability requirements
- The type of data available for training the model
- The amount of data available for training and testing the model
- The number of attributes in the input data expected to be used by the model
- The expected number of groups for clustering
- Previous experience
- Trial and error method

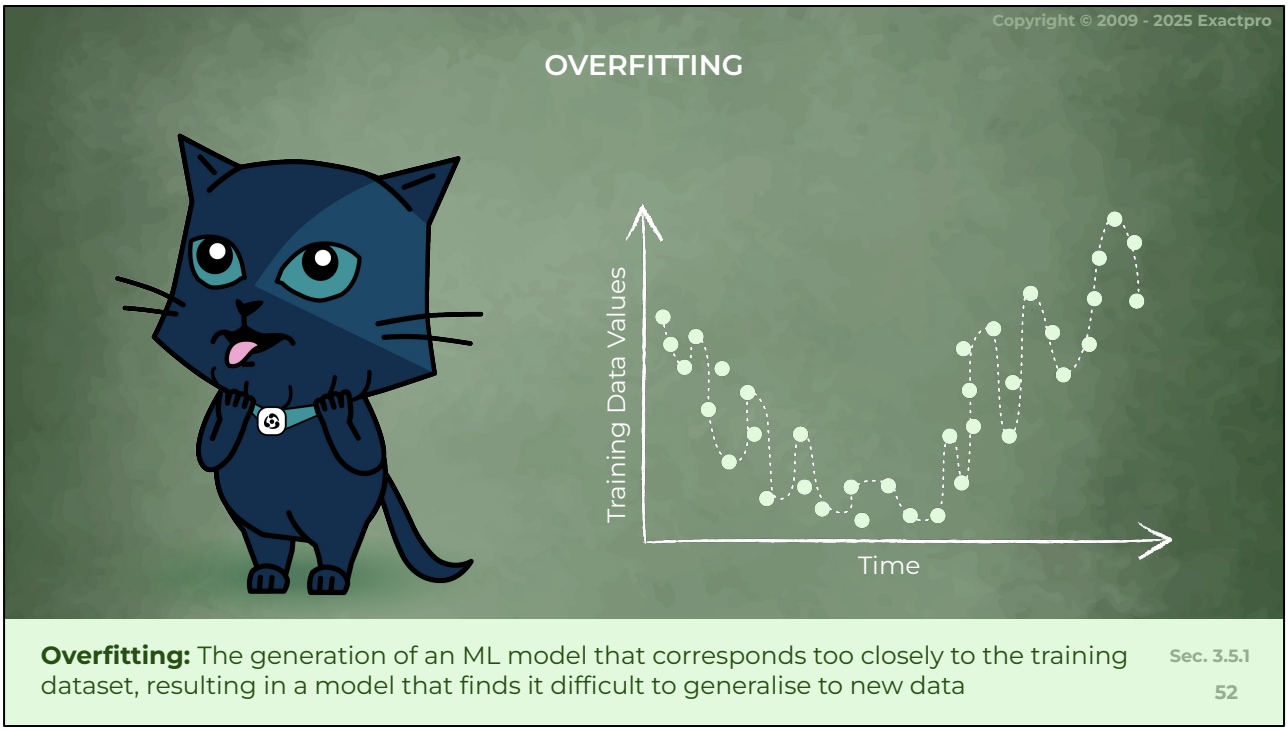
There is no definitive approach on how to choose an optimal ML algorithm, model settings and model hyperparameters. In practice, this set is chosen based on a mix of the following factors:

- the required functionality;
- the required quality characteristics, such as:
  - accuracy;
  - constraints on available memory;
  - the speed of training (and retraining) the model;
  - the speed of prediction;
  - transparency, interpretability and explainability requirements;
- the type of data available for training the model;
- the amount of data available for training and testing the model;
- the number of attributes in the input data expected to be used by the model;
- the expected number of groups for clustering;
- previous experience;
- trial and error method.

3.5 Overfitting and Underfitting

As we have learned earlier, when a machine learning expert chooses the best model, he or she looks at quality indicators, among other things, which may lead to several possible scenarios.

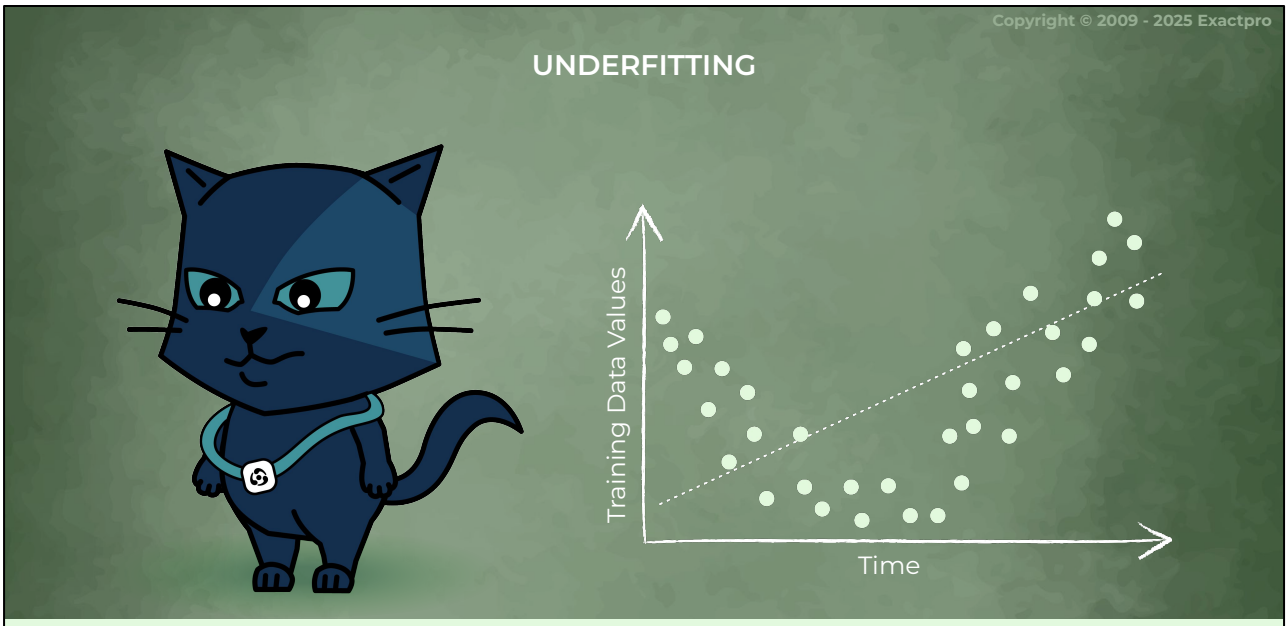
3.5.1 Overfitting



Overfitting: The generation of an ML model that corresponds too closely to the training dataset, resulting in a model that finds it difficult to generalise to new data

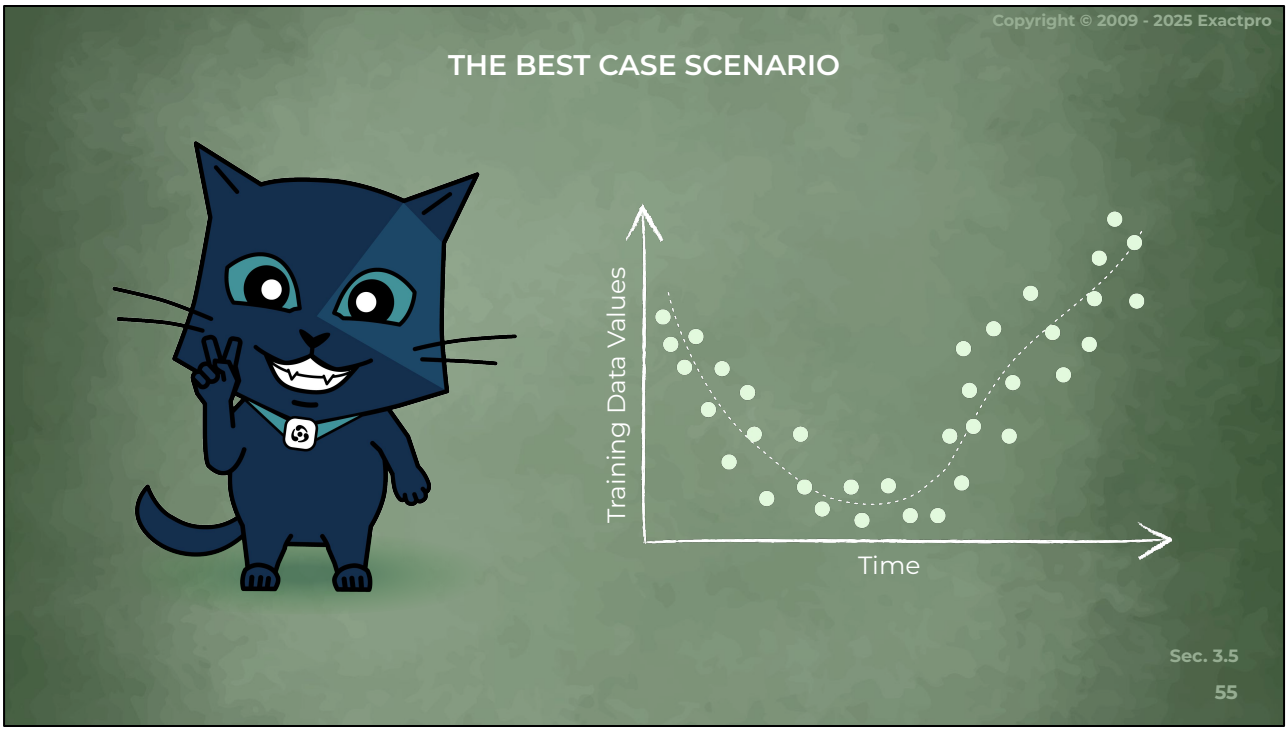
The first situation is **overfitting**. Overfitting is when a model performs well on a training dataset, but as soon as new data becomes available, the quality of the prediction deteriorates significantly. It can also occur when insufficient data is provided in the training dataset. *Example: We have a dataset of 100 images, each labelled "cat" or "dog." Each image is represented by a set of features such as the height, width, and colour intensity. We want to train an ML model to classify these images as either cats or dogs. We use a decision tree algorithm. We split our dataset into a training set (80%) and a test set (20%) and we train our decision tree on the training set and evaluate it on the test set. Initially, the model achieves an accuracy of 80% on the test set, which is ok. But we notice that the model is not performing well on images of cats with closed eyes. We add more training data focusing on cats with their eyes closed. We collect 20 more images of this type and add them to the training set. We retrain our decision tree using the augmented dataset and evaluate its performance on the test set. The model achieves a perfect accuracy of 100% on the test set. But because overfitting, the decision tree has become too specific to the training data. It has learned to recognise the specific cats with their eyes closed as well as other unique characteristics that might be present only in our limited dataset. This hyper-specificity prevents the model from generalising well to unseen data. When we try to use this overfitted model in the real world, it performs poorly and misclassifies most of the new images. The model has memorised the training data instead of learning the underlying patterns that generalise to a broader range of data.*

3.5.2 Underfitting

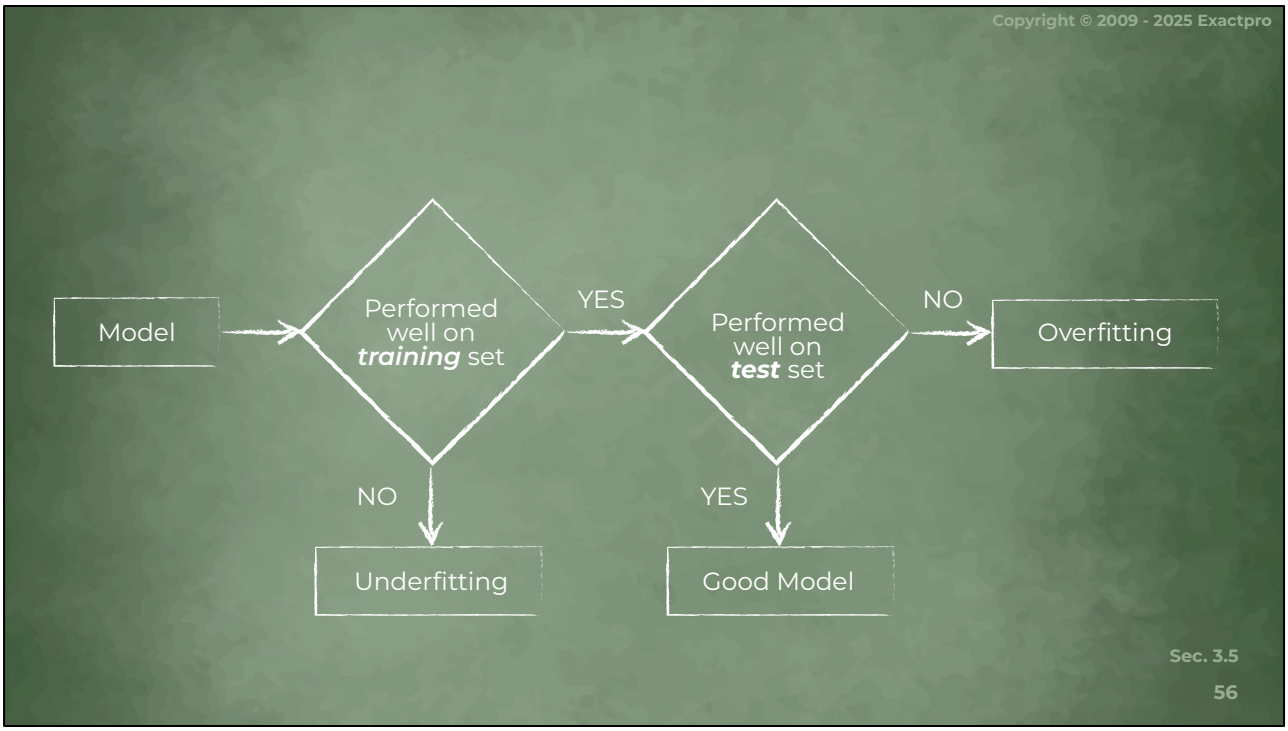


Underfitting: The generation of an ML model that does not reflect the underlying trend of the training dataset, resulting in a model that finds it difficult to make accurate predictions

The second situation occurs when the model performs poorly on both the training data and the new data. This situation is called **underfitting**. Underfitting occurs when the model is not sophisticated enough to accurately fit to the patterns in the training data. *Example: We have a dataset with 2 classes: "cats" and "not cats." The goal is to build an ML model that can accurately tell one from another. We train the model using a simple algorithm with limited capacity which makes it hard to learn the complex features that distinguish cats from non-cat images effectively. As a result, the model might underfit the training data. It fails to capture the intricate details in cat images, leading to low accuracy on both the training and test datasets. The model's performance may be poor as it struggles to generalise to new, unseen cat images. The model might make overly simplistic assumptions, such as classifying any image with four legs as a cat, which is inadequate. To address underfitting, we can try using more powerful models with deeper architectures specifically designed for image classification. Additionally, augmenting the dataset with more diverse cat images can also help mitigate underfitting.*



And the third situation is when the model is equally good at predicting both training and new data. This, of course, is **the best case scenario**.



Let's go over it one more time. Unfortunately cases of overfitting and underfitting are quite common. **Overfitting** occurs when a model fails to generalise to new data but performs exceedingly well on the training data. This often happens when a model becomes too complex or has too many parameters specific to the available training data. Signs of overfitting include performing well on a training set but failing on a test set. An **underfit** model, on the other hand, exhibits high errors on both the training and test data. It occurs when a model is too simplistic or lacks the capacity to capture the underlying data patterns. This may result in a model that is too generalised and fails to capture important data trends. Overcoming overfitting and underfitting is essential in developing good and effective machine learning models. The goal is to find the right balance where the model generalises well to new data by accurately capturing the underlying patterns and relationships in the data.

THANK YOU