Hands on Machine Learning with scikit-Learn,Keras &TensorFlow chapter 1

LAMIA KHALID ALARIQI

INTRODUCTION

Machine learning, a subset of artificial intelligence, has gained significant attention in recent years due to its ability to enable computers to learn and make predictions without being explicitly programmed. It has found applications in various domains, ranging from autonomous vehicles to personalized recommendations.

What is Machine Learning ?

Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed." - Arthur Samuel, 1959. "And a more engineering-oriented one: A computer program is said to learn from **experience** E with respect to some task T and some **performance**

measure P if its performance on T, as measured by P, improves with experience E ."Let's make it clear: The common example of Machine Learning is a Spam Filter. A spam filter is a Machine Learning program that, given examples of regular emails (non-spam, also called 'ham') and irregular emails (spam), learns to detect and flag spam messages. The training set is the set of examples that the system uses to learn, and each example in the training set is called a training instance (or sample). The part of a Machine Learning system that learns and makes predictions is called a model. Neural networks and random forests are examples of models. In the case of a spam filter, the task T is to flag spam for new emails, the experience E is the training data, and the performance measure P needs to be defined. For example, the accuracy, which is the ratio of correctly classified emails, can be used as a performance measure. Accuracy is often used in classification tasks.

Why Use Machine Learning?

You may ask yourself why we need to use machine learning and what's the difference between it and traditional programming. Let's use the same example to illustrate the difference. If you were to write the spam filter using traditional programming techniques:

- First of all, you would have to examine the most common words in each message, which can be done by checking the most frequent words. Consequently, a long list of words would have to be created.
- You would write a detection algorithm for each of the patterns that you noticed, and your program would flag emails as spam if a number of these patterns were detected

You may think this would be enough to efficiently detect email spam, but in fact, it is not. If the spammer discovers your approach, they will directly try to change those words to their synonyms, and certainly, your program will not be able to detect those new words. As a result, you would have to update your program each time. In contrast, a spam filter based on Machine Learning techniques automatically learns which words and phrases are good predictors of spam by detecting unusually frequent patterns of words in the spam examples compared to the ham examples. If the spammer decides to change the words, the program can notice that by itself .

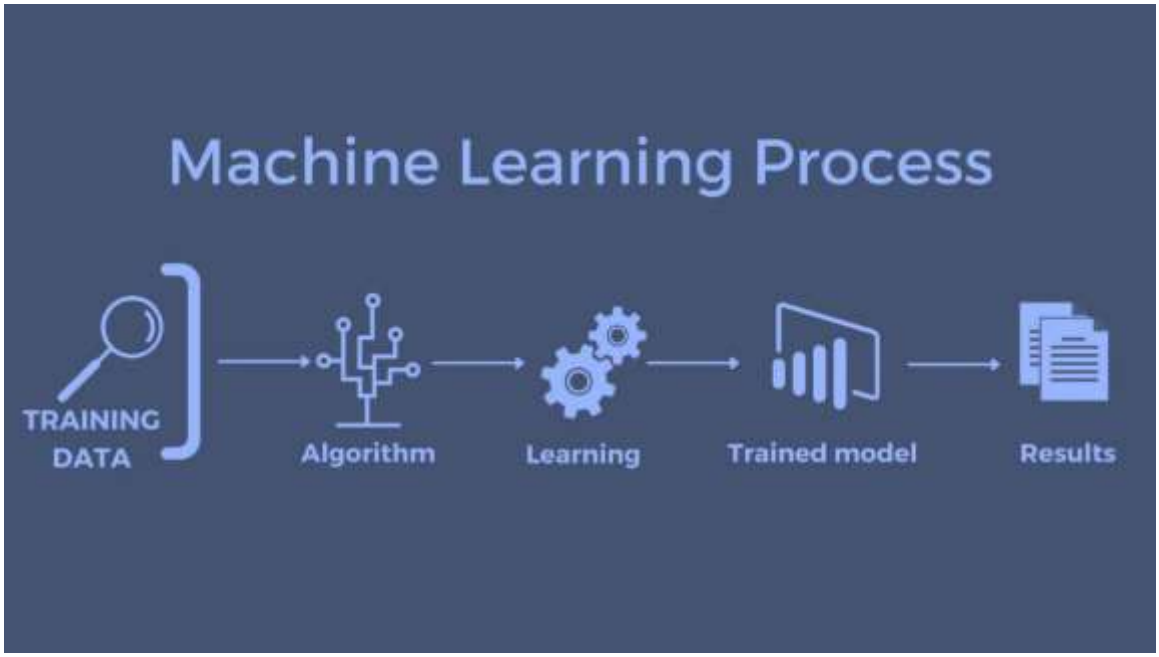

Figure 1: how does machine learning work

Machine Learning is great for solving complex problems or problems without known solutions. For example, let's consider speech recognition. If you want to create a program that can distinguish between the words "one" and "two," a simple approach like detecting high-pitch sounds may not work well when dealing with lots of different words spoken by many people in various languages and noisy environments. Instead, the best solution is to develop an algorithm that can learn on its own. By providing the algorithm with many examples of recorded speech for each word, it can automatically learn to differentiate between them. Machine Learning models, like neural networks, are designed to extract patterns and features from data, enabling them to accurately recognize different words. Machine Learning can also help humans learn. By examining the models, we can gain insights into what they've learned. For instance, a spam filter trained on a large dataset of spam emails can reveal the words or combinations of words it identifies as strong indicators of spam. This can uncover unexpected correlations or trends and lead to a better understanding of the problem.

To summarize, Machine Learning is great for: Problems for which existing solutions require a lot of fine-tuning or long lists of rules: one Machine Learning model can often simplify code and perform better than the traditional approach. Complex problems for which using a traditional approach yields no good solution: the best Machine Learning techniques can perhaps find a solution. Fluctuating environments: a Machine Learning system can easily be retrained on new data, always keeping it up to date. Getting insights about complex problems and large amounts of data.

Examples of Applications:

Machine learning finds diverse applications across various domains. Figure 2 illustrate some examples.

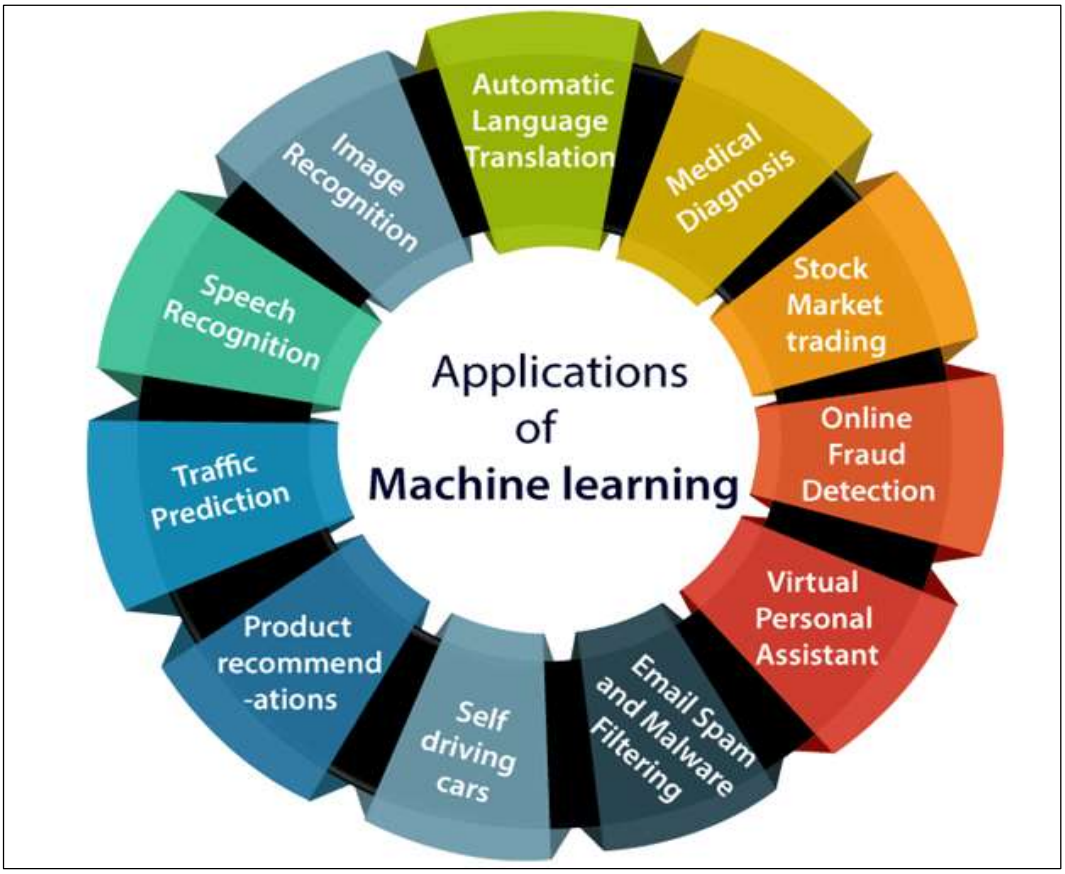

Figure 2: ML Application

Some other examples:

- Making your app react to voice commands
- Detecting credit card fraud
- Forecasting your company's revenue next year, based on many performance metrics
- Automatically flagging offensive comments on discussion forums
- Summarizing long documents automatically
- Building an intelligent bot for a game

- Recommending a product that a client may be interested in, based on past purchases
- Representing a complex, high-dimensional dataset in a clear and insightful diagram
- Detecting credit card frau

Types of Machine Learning Systems

There are so many different types of Machine Learning systems that it is useful to classify them in broad categories, based on the following criteria:

- How they are supervised during training (supervised, unsupervised, semi-supervised, self-supervised, and others)
- Whether or not they can learn incrementally on the fly (online versus batch learning)
- Whether they work by simply comparing new data points to known data points, or instead by detecting patterns in the training data and building a predictive model, much like scientists do (instancebased versus model-based learning)

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

1) Training Supervision

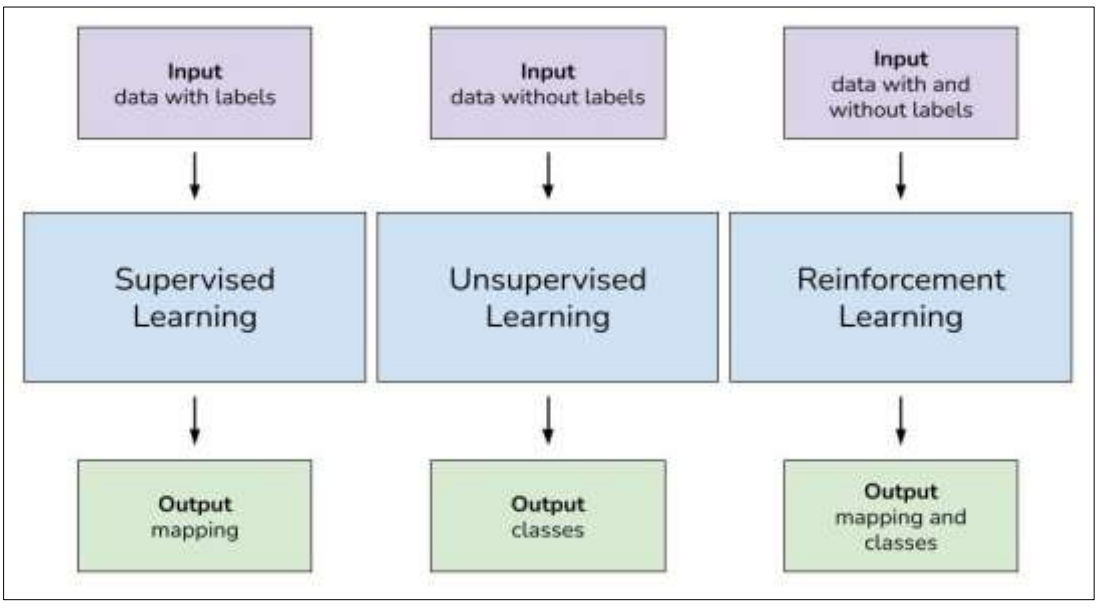

Figure 3:training supervision types

1.1 Supervised learning: In this approach, the machine learning system is trained using labeled data, where the input data is paired with corresponding desired outputs. The system learns to map inputs to outputs based on the provided labeled examples. A typical supervised learning task is classification. The spam filter is a good example of this: it is trained with many example emails along with their class (spam or ham), and it must learn how to classify new emails.

1.2 Unsupervised learning: Unlike supervised learning, unsupervised learning does not use labeled data. The system learns to find patterns, structures, and relationships in the input data without explicit guidance. It aims to discover hidden insights or groupings within the data. For instance, let's consider a dataset containing information about customers of an e-commerce website. This dataset includes features such as age, purchase history, and browsing behavior. By applying an unsupervised learning algorithm like k-means clustering, we can group the customers into distinct clusters based on their similarities. The algorithm would analyze the patterns and relationships among the features and automatically group the customers into clusters. Each cluster would consist of customers who exhibit similar behaviors or characteristics. This can help in understanding customer segments, targeted marketing, or personalized recommendations. Visualization algorithms are also good examples of unsupervised learning: you feed them a lot of complex and unlabeled data, and they output a 2D or 3D representation of your data that can easily be plotted

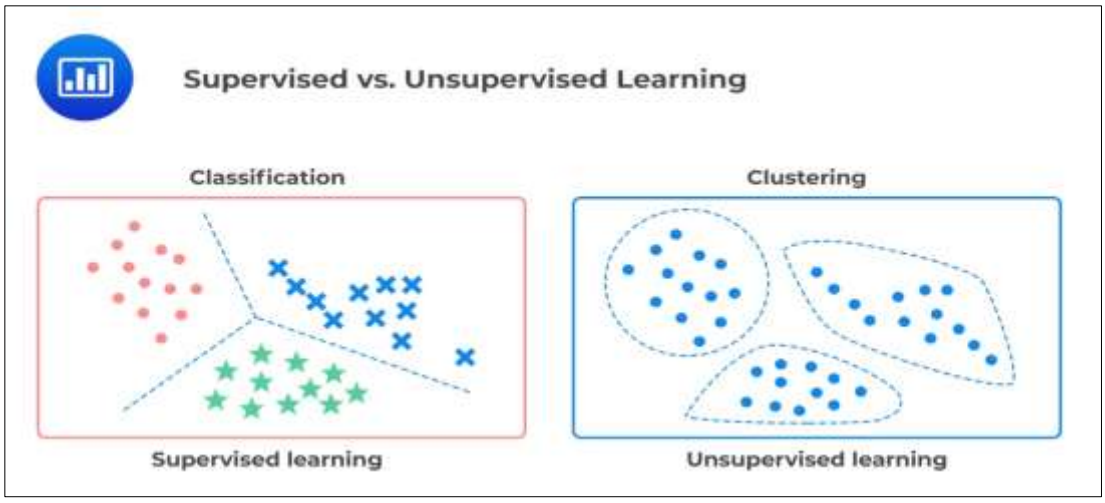

Figure 4:supervised Vs unsupervised

1.3 Self-supervised learning: Self-supervised learning is a type of unsupervised learning where the system learns from the data itself by creating surrogate tasks. It leverages the inherent structure or properties of the data to generate labels or targets for training. or example, suppose that what you really want is to have a pet classification model: given a picture of any pet, it will tell you what species it belongs to. If you have a large dataset of unlabeled photos of pets, you can start by training an image-repairing model using self-supervised learning. Once it performs well, it must be able to distinguish different pet species: indeed, when it repairs an image of a cat whose face is masked, it knows it must not add a dog's face. Assuming your model's architecture allows it (and most neural network architectures do), it is then possible to tweak the model so that it predicts pet species instead of repairing images. The final step consists of fine-tuning the model on a labeled dataset: the model already knows what cats, dogs and other pet species look like, so this step is only needed so the model can learn the mapping between the species it already knows and the labels we expect from it. Some people consider self-supervised learning to be a part of unsupervised learning, since it deals with fully unlabeled datasets. But self-supervised learning uses (generated) labels during training, so in that regard it's closer to supervised learning. And the term "unsupervised learning" is generally used when dealing with tasks like clustering,

dimensionality reduction or anomaly detection, whereas self-supervised learning focuses on the same tasks as supervised learning: mainly classification and regression. In short, it's best to treat self-supervised learning as its own category.

1.4 Semi-supervised learning: This approach combines elements of both supervised and unsupervised learning. It uses a small amount of labeled data along with a larger amount of unlabeled data to train the system. The labeled data helps guide the learning process, while the unlabeled data aids in capturing additional patterns and generalization. For instance: Some photo-hosting services, such as Google Photos, are good examples of this. Once you upload all your family photos to the service, it automatically recognizes that the same person A shows up in photos 1, 5, and 11, while another person B shows up in photos 2, 5, and 7. This is the unsupervised part of the algorithm (clustering). Now all the system needs is for you to tell it who these people are. Just add one label per person and it is able to name everyone in every photo, which is useful for searching photos. Most semi-supervised learning algorithms are combinations of unsupervised and supervised algorithms

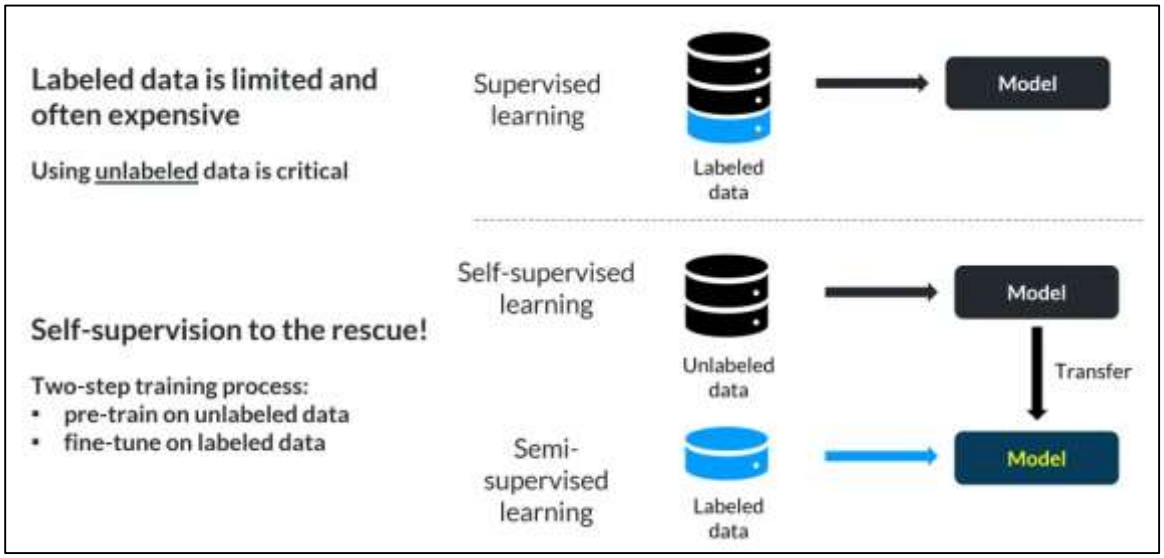

Figure 5:self-supervised vs semi supervised

1.5 Reinforcement learning: The learning system, referred to as an agent, has the ability to observe its environment, make decisions, take actions, and receive rewards or penalties in return. The rewards can be in the form of positive values, while penalties are represented by negative rewards or punishments. The objective of the agent is to learn on its own the optimal strategy, known as a policy, to maximize the cumulative reward over time. The policy guides the agent in determining the appropriate action to take in a given situation.

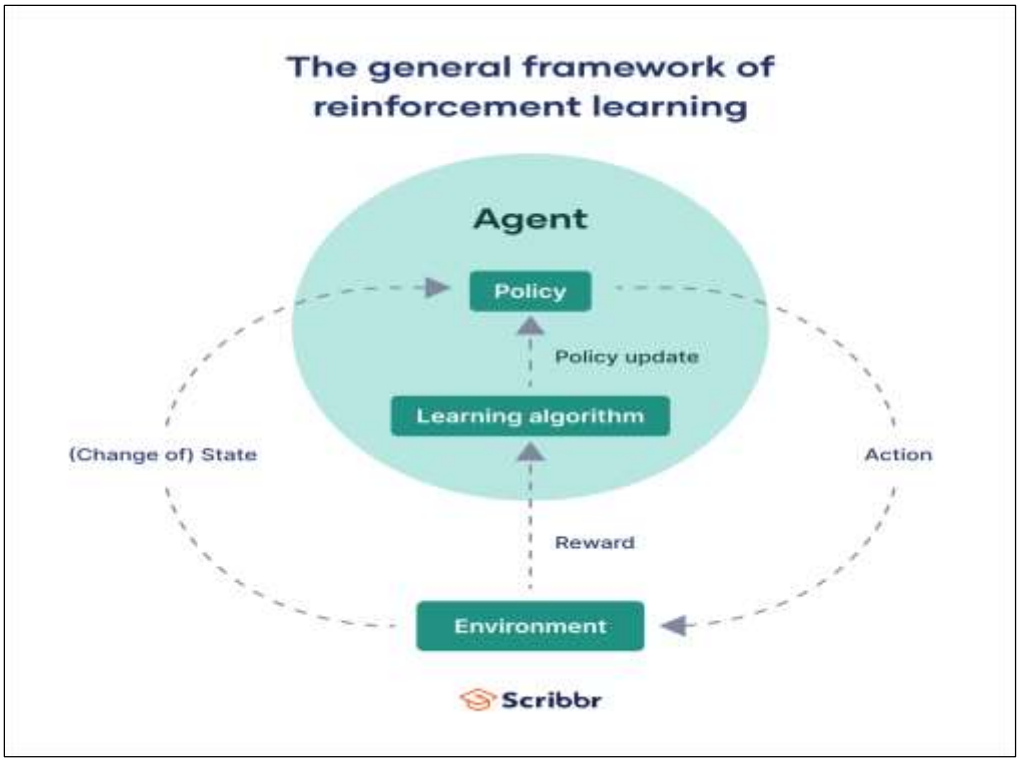

Figure 6:Reinforcement Learning

2) Batch versus Online Learning

Another criterion used to classify Machine Learning systems is whether or not the system can learn incrementally from a stream of incoming data.

2.1 Batch Learning

In batch learning, the system learns using all available data and cannot learn incrementally. This process is typically time-consuming and resource-intensive, so it is commonly done offline. After training, the system is deployed for production and operates without further learning, applying what it has already learned. This is known as offline learning. However, over time, the performance of a model may gradually decline due to the changing nature of the world, while the model remains unchanged. This phenomenon is referred to as model rot or data drift. To address this, it is necessary to regularly retrain the model with up-to-date data. The frequency of retraining depends on the specific use case. For instance, a model classifying pictures of cats and dogs may experience slow performance decay, while a model making predictions in a fast-evolving financial market may experience faster decay.

2.2 Online learning

In online learning, you train the system incrementally by feeding it data instances sequentially, either individually or in small groups called minibatches. Each learning step is fast and cheap, so the system can learn about new data on the fly, as it arrives. One important parameter of online learning systems is how fast they should adapt to changing data: this is called the learning rate. If you set a high learning rate, then your system will rapidly adapt to new data, but it will also tend to quickly forget the old data (you don’t want a spam filter to flag only the latest kinds of spam it was shown). Conversely, if you set a low learning rate, the system will have more inertia; that is, it will learn more slowly, but it will also be less sensitive to noise in the new data or to sequences of non-representative data points (outliers).

Figure 7 shows the different between these two kind.

	Batch Machine Learning	Online Machine Learning
Complexity	Is complex because models need to be trained frequently. Also, difficult to evaluate.	Less complex because training data is more complex, and models are easier to evaluate.
Compute	Requires larger compute because you need to keep training models.	Requires less compute because models are trained on batches.
Maintenance	Easy to maintain in production because the prediction changes less frequently.	Difficult to maintain in production because the prediction model keeps on changing.
Application	Suitable for scenarios where data is changing less frequently.	Suitable for scenarios where data is changing constantly.

Figure 7:Online Learning Vs Batch Learning

3) Instance-Based Versus Model-Based Learning

Another way to categorize Machine Learning systems is based on their ability to generalize. In most Machine Learning tasks, the objective is to make accurate predictions. This entails training the system on a set of examples and then evaluating its ability to generalize and make predictions for unseen instances. While having good performance on the training data is important, the ultimate goal is to perform well on new, unseen data. There are two main approaches to generalization:

3.1 Instance-based learning

Possibly the most trivial form of learning is simply to learn by heart. If you were to create a spam filter this way, it would just flag all emails that are identical to emails that have already been flagged by users—not the worst solution, but certainly not the best. Instead of just flagging emails that are identical to known spam emails, your spam filter could be programmed to also flag emails that are very similar to known spam emails. This requires a measure of similarity between two emails. A (very basic) similarity measure between two emails could be to count the number of words they have in common. The system would flag an email as spam if it has many words in common with a known spam email. This is called instance-based learning: the system learns the examples

by heart, then generalizes to new cases by using a similarity measure to compare them to the learned examples (or a subset of them).

To summarize: In this approach, the system learns by memorizing the training examples and their associated outcomes. When a new instance needs to be predicted, the system compares it to the stored instances and uses the closest or most similar examples to make a prediction. This method relies on the assumption that similar instances have similar outcomes.

3.2 Model-Based Learning

the system learns the underlying patterns and relationships in the training data to build a general model. The model captures the essential features and characteristics of the data and can be used to make predictions for new instances. This approach focuses on extracting general rules or representations from the training data.

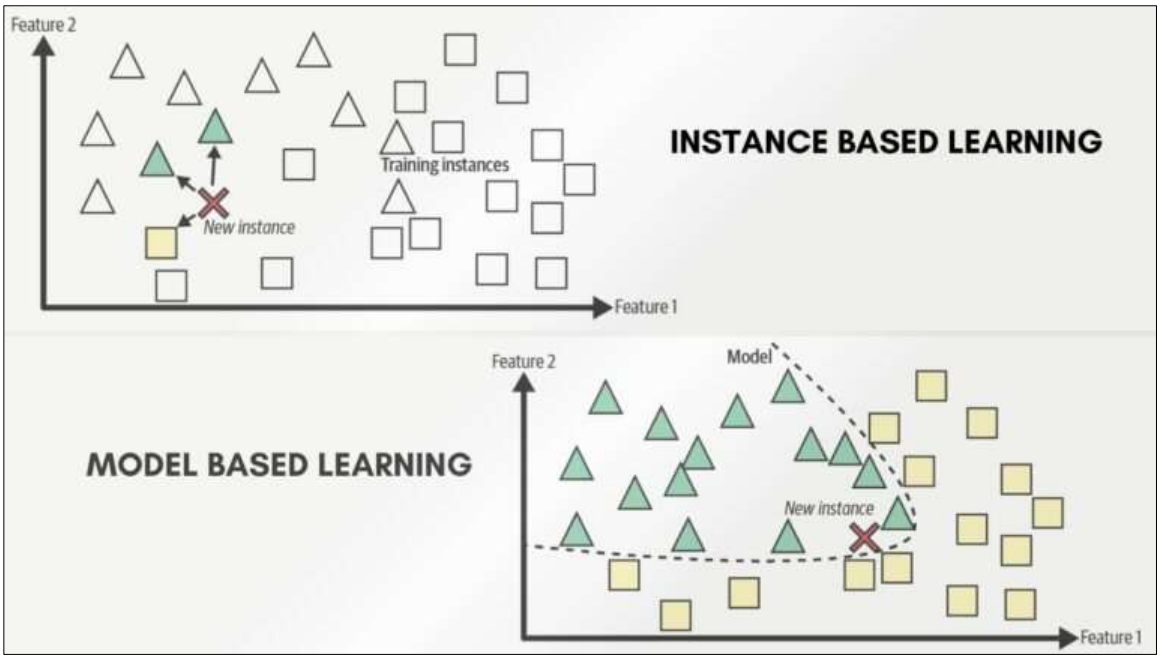

Figure 8: Instance-Based Vs Model-Based Learning

Before using a model, it is necessary to define the parameter values (θ and θ). To determine the optimal parameter values that make the model perform best, a performance measure needs to be specified. This can be achieved by defining a utility function or a cost function. In the case of Linear Regression problems, a cost function is typically used to measure the distance between the model's predictions and the training examples, with the objective of minimizing this distance. The Linear Regression algorithm is employed to train the model by feeding it the training examples. The algorithm finds the parameter values (θ = 3.75 and θ = 6.78 × 10^−6) that make the linear model best fit the data. The term "model" can refer to the type of model, the fully specified model architecture, or the final trained model with specific parameter values. Model selection involves choosing the type of model and specifying its architecture. Training the model involves running an algorithm to find the parameter values that optimize its fit to the training data and enable accurate predictions on new data. Once the model is trained and closely fits the training data, it can be used to make predictions.

In summary:

- You studied the data.
- You selected a model.
- You trained it on the training data

Main Challenges of Machine Learning

Since your main task is to select a model and train it on some data, the two things that can go wrong are “bad model” and “bad data.” Let’s start with examples of bad data.

- **Insufficient Quantity of Training Data**

Machine Learning algorithms require a significant amount of data to perform effectively. Even for relatively simple problems, thousands of examples are typically needed, while complex tasks like image or speech recognition may require millions of examples. However, it is worth noting that in some cases, parts of an existing model can be reused, which can help mitigate the data requirements. Nonetheless, compared to the remarkable learning abilities of

a toddler, Machine Learning still has a long way to go in terms of efficiency and the amount of data needed for training.

- **Non-representative Training Data**

Non-representative training data refers to the concept where the data used for training in machine learning does not accurately represent the problem at hand. It can occur when the training data is not randomly sampled, biased, or lacks proper balance between different classes or scenarios. Non-representative training data can lead to poor performance and inaccurate results when the model encounters new or unfamiliar cases. To address this issue, it is important to collect training data that is representative of the problem, ensuring random sampling, addressing biases, and maintaining balance between different classes. Regular updates to the data may also be necessary to keep up with changes in the environment or conditions.

- **Poor-Quality Data**

When training data contains errors, outliers, or noise, it becomes more challenging for the system to identify underlying patterns, resulting in poorer performance. Therefore, it is beneficial to invest time in cleaning up training data. Data scientists often allocate a significant portion of their time to this task. There are specific scenarios where cleaning the data is necessary:

1) Outliers: If some instances are clearly outliers, it might be useful to either discard them or manually correct the errors associated with them.
2) Missing features: In cases where some instances lack a few features (e.g., 5% of customers did not provide their age), several options can be considered. You can choose to ignore the attribute altogether, remove those instances, fill in the missing values (e.g., using the median age), or train separate models with and without the feature.

Cleaning up training data helps improve the quality and reliability of the model's predictions. By removing errors, outliers, and addressing missing values, the system can better learn the underlying patterns and perform more effectively

- **Irrelevant Features**

Irrelevant features refer to the attributes or variables in the dataset that do not contribute useful information to the problem being solved. These features are not related to the target variable or do not have a meaningful impact on the model's predictions. Identifying and handling irrelevant features is crucial in machine learning as they can introduce noise, increase computational complexity, and potentially lead to overfitting. Overfitting occurs when a model becomes too specialized to the training data, resulting in poor performance on new, unseen data. To address irrelevant features, various techniques can be employed. One approach is feature selection, where relevant features are selected based on their importance or correlation with the target variable. Another technique is feature extraction, which involves transforming the original features into a new set of features that capture the essential information. By eliminating irrelevant features, the model becomes more focused on the meaningful aspects of the data, leading to improved performance, reduced dimensionality, and enhanced interpretability.

- **Overfitting the Training Data**

Overgeneralization, both by humans and machines, can lead to inaccurate conclusions. In machine learning, this is known as overfitting, where a model performs well on the training data but fails to generalize effectively. Overfitting can occur when complex models detect patterns in noise or irrelevant features, which do not hold true for new instances. Regularization is a technique used to constrain models and reduce the risk of overfitting. By imposing constraints on the model's parameters, such as keeping them small, regularization strikes a balance between fitting the training data and maintaining simplicity for better generalization. The provided example illustrates overfitting with a polynomial life satisfaction model that performs better on the training data than a simple linear model. However, the high-degree polynomial model may capture noise or chance patterns, making its predictions less reliable. Including uninformative features, like a country's name, can lead to spurious patterns that do not generalize to new instances.To demonstrate the impact of regularization, three models are shown. The first model fits only the circular data points, the second model fits all data points (circles and squares), and the third model applies

regularization. The regularized model has a smaller slope, indicating less accurate fit to the training data (circles), but it has better generalization performance on new examples (squares) that were not part of the training set.

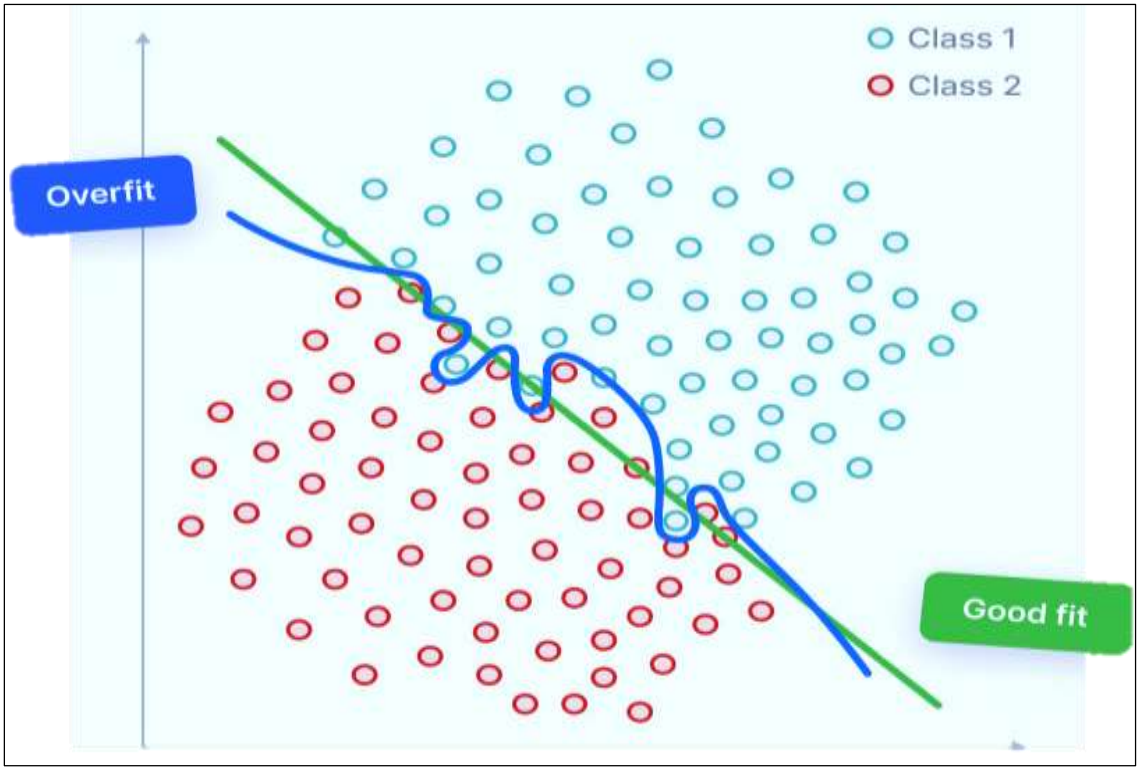

Figure 9: Overfitting the Training Data

- **Underfitting the Training Data**

Underfitting is the opposite of overfitting and happens when a model is too simplistic to capture the underlying patterns in the data

To address underfitting, there are several options available:

1. Choose a more powerful model that can capture greater complexity by having more parameters.
2. Improve the features provided to the learning algorithm through feature engineering, which involves selecting or creating informative features.
3. Relax the constraints on the model by reducing regularization, which can be achieved by decreasing the regularization hyperparameter.

This allows the model to have more flexibility in fitting the training data.

- **Testing and Validating**

To assess how well a model will perform on new cases, it is necessary to evaluate it on unseen data. One way to achieve this is by putting the model into production and monitoring its performance. However, if the model performs poorly, user complaints may arise, making this approach less desirable. A better alternative is to divide the available data into two sets: the training set and the test set. The model is trained using the training set, and its performance is then assessed using the test set. The error rate on the test set is known as the generalization error or out-of-sample error, and it provides an estimate of how well the model will perform on new, unseen instances. If the model exhibits a low error rate on the training set but a high generalization error, it indicates that the model is overfitting the training data. This means that the model is overly specialized to the training set and fails to generalize effectively to new cases. Evaluating the model on the test set helps identify such issues and allows for adjustments to improve its generalization performance.

- **Hyperparameter Tuning and Model Selection**

When deciding between different types of models or determining the optimal hyperparameters for a model, evaluating the model using a test set alone may lead to unreliable results. To address this, a common solution is to use holdout validation. Holdout validation involves splitting the training set into two parts: the reduced training set and the validation set. Multiple models with different hyperparameters are trained using the reduced training set, and their performance is evaluated on the validation set. The model that performs best on the validation set is selected as the preferred model.

After the holdout validation process, the chosen model is trained on the full training set, including the validation set. This final model is then assessed on the separate test set to estimate its generalization error, providing an indication of how well it is expected to perform on new, unseen data. By using holdout validation, the risk of overfitting to the test set and obtaining overly optimistic results is reduced. It ensures that the model's performance is evaluated on an independent set of data, allowing for a more reliable estimation of its ability to generalize to new instances.

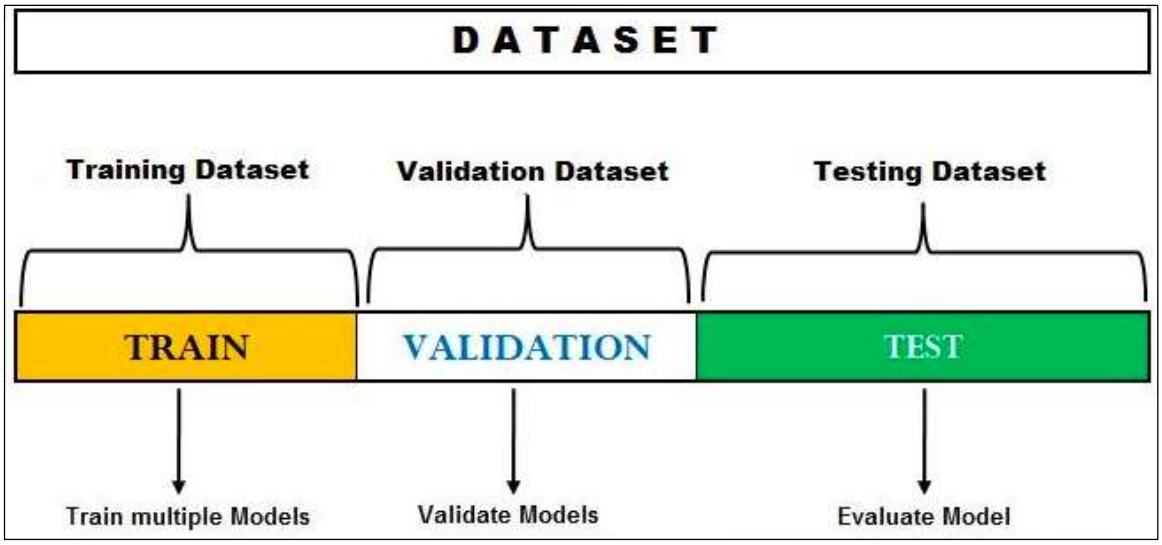

Figure 10: holdout validation process

While holdout validation is generally effective, it can have limitations depending on the size of the validation set. If the validation set is too small, the evaluations of models may be imprecise, leading to the possibility of mistakenly selecting a suboptimal model. On the other hand, if the validation set is too large, the remaining training set becomes significantly smaller compared to the full training set. This is problematic because the final model will be trained on the full training set, making it less ideal to compare candidate models trained on a much smaller subset. To address these issues, a solution is to employ repeated cross-validation, which involves using multiple small validation sets. Each model is trained on the remaining data and then evaluated once per validation set.

By averaging the evaluations across all validation sets, a more accurate measure of a model's performance can be obtained. However, one drawback of this approach is that the training time increases proportionally to the number of validation sets used. Although repeated cross-validation requires more computational time, it provides a more robust and reliable assessment of a model's performance, mitigating the potential biases or uncertainties caused by the size of the validation set in traditional holdout validation.

- **Data Mismatch**

In situations where a large amount of training data is available but may not be representative of the data used in production, it is crucial to ensure that both the validation set and the test set accurately reflect the production data. For instance, when developing a mobile app to classify flower species, millions of web images of flowers can be easily obtained, but they may not accurately represent the images taken using the app on a mobile device. In such cases, having a smaller set of representative pictures (e.g., 1,000 images taken with the app) is important. To address this, the validation set and test set should exclusively consist of representative pictures. These sets can be created by shuffling and dividing the representative pictures, ensuring that no duplicates or similar images are present in both sets. After training the model on the web pictures, if the model's performance on the validation set is disappointing, it becomes challenging to determine whether the issue is due to overfitting the training set or the mismatch between web pictures and app pictures. To resolve this, a train-dev set can be created by setting aside some training pictures from the web. After training the model on the training set (excluding the train-dev set), its performance can be evaluated on the train-dev set. If the model performs poorly on the train-dev set, it indicates overfitting, and steps like simplifying or regularizing the model, obtaining more training data, or cleaning up the training data can be taken. However, if the model performs well on the train-dev set, it can be evaluated on the dev set.

If the model performs poorly on the dev set, it suggests a data mismatch problem. In this case, preprocessing the web images to resemble app

pictures can be attempted, followed by retraining the model. Once the model performs well on both the train-dev set and the dev set, a final evaluation on the test set provides an estimate of its performance in production.

My understanding of Machine Learning:

Machine learning is a branch of artificial intelligence that focuses on creating algorithms and models that can automatically learn and make predictions or choices from data without being explicitly programmed. It entails training a computer system on a dataset, allowing it to learn patterns, correlations, and insights, and then using that information to generate accurate predictions or perform informed decisions on previously unknown data.

Suggesting a Machine Learning Project: Grammar Correction for English Text