**THE MACHINE LEARNING LANDSCAPE**

**Introduction**

* Each training example is called ***training instance*** or ***sample***.
* Only a machine becoming **better** at its tasks with **more data** comes under category of **machine learning**.
* Opposite term for **machine learning** is **traditional programming**.

**Spam Mail Detection**

* Opposite term for **spam** is ***ham***.
* Models learn the **pattern of words** used in mail for **filtering**.
* **Traditional approach** will take a lot of programming efforts with constant need to be updated, unlike **ML models** which learns on its own from data.
* **Data mining:** Applying ML techniques to **dig out patterns** that were **not** visible on ground level.

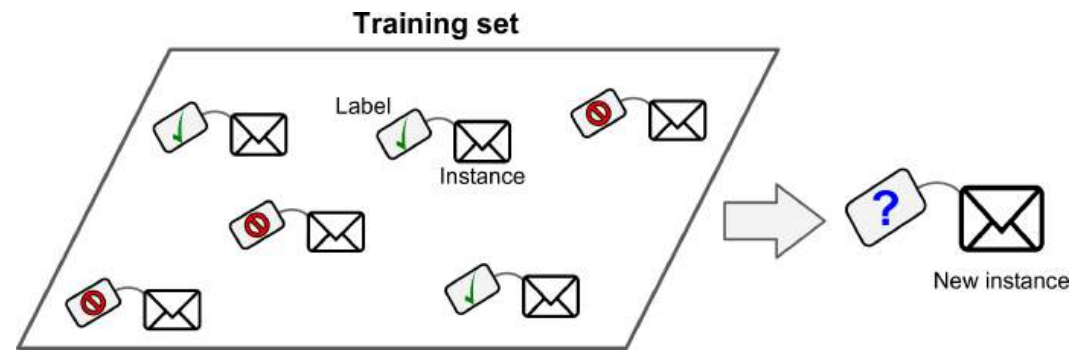
**Nature of Models**

* Supervised, unsupervised, semi-supervised or reinforcement.
* Batch or online.
* Instance-based or model-based learning

**Supervised & Unsupervised Learning**

Supervised learning:-

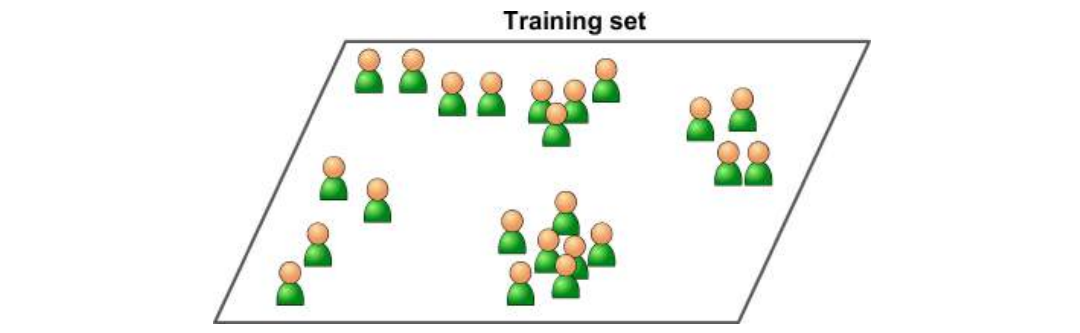
* The **training data** contains the **solution** with them, called ***labels***.
* These **labels** are what has to be **predicted** in **testing data** (like **price** of a car).



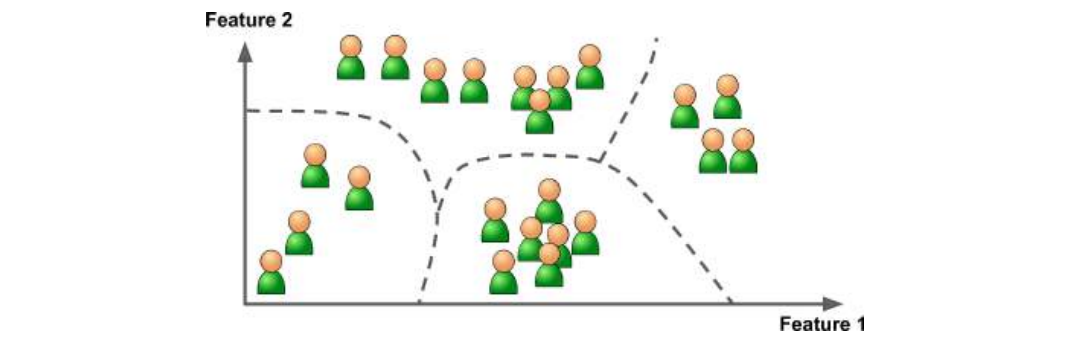
* Common uses are **classification** & **numeric prediction**.
* Predicting **numeric values** is called **regression**.
* The **set of features** which help in **predicting** are called ***predictors*** (like **mileage, age & brand** of a car etc).
* **Attribute:** Name of data category (like **mileage**).
* **Feature:** Name and value of a particular data (like **mileage = 15,000**).
* Some **regression types** can be used for **classification** too (like **logistic regression**).
* **Logistic regression:** Tells the probability of a value for belonging to a **category** (like **20% chance for being a spam**).
* **Supervised learning algorithms** to be covered are:
  + k-Nearest algorithm
  + Linear regression
  + Logistic regression
  + Support vector machines (SVMs)
  + Decision trees & random forests
  + Neural networks

Unsupervised learning:-

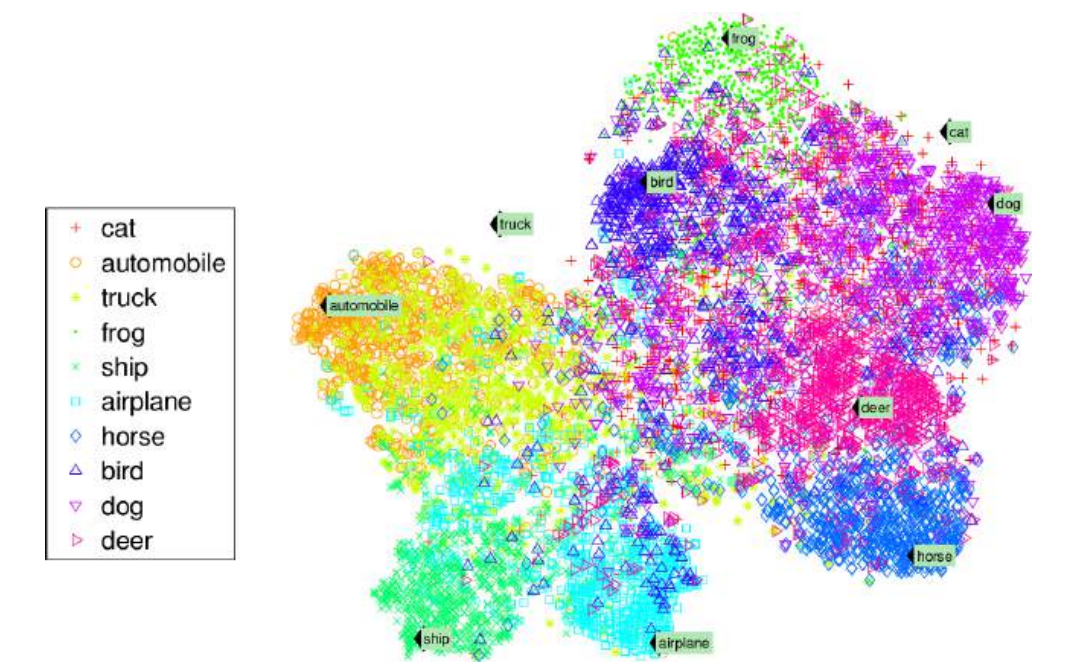
* System learns **without** any **label** or **guidance**.
* The training data are complete group of **strangers**.



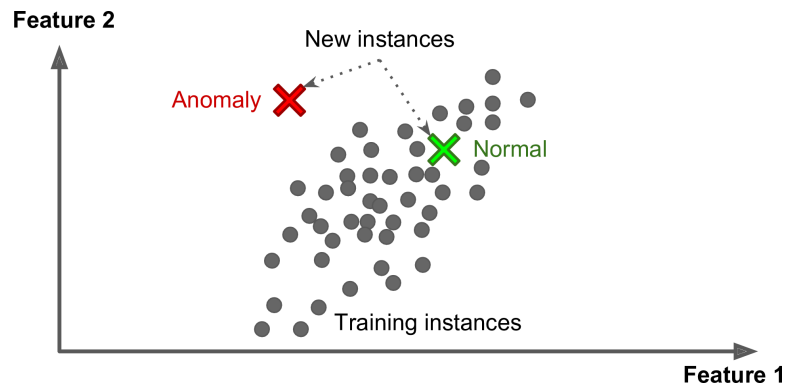
* **Unsupervised algorithms** to be covered are:
  + **Clustering**
    - K-Means
    - DBSCAN
    - Hierarchical cluster analysis
  + **Anomaly & novelty detection**
    - One-class SVM
    - Isolation forest
  + **Visualization & dimensionality reduction**
    - Principal component analysis (PCA)
    - Kernel PCA
    - Locally linear embedding (LLE)
    - t-distributed stochastic neighbour embedding (t-SNE)
  + **Association rule learning**
    - Apriori
    - Eclat
* **Clustering:** Dividing data into various **groups** or **clusters** without any help by the trainer.



* **Visualization algorithm:** Are fed with lot of complex **unlabelled** data & they provide **2D** or **3D** representation of that data (like the **t-SNE visualization** example below).



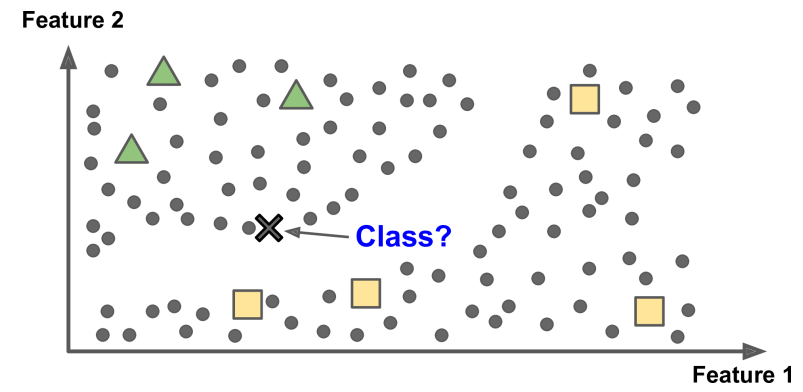
* **Dimensionality reduction:** Simplifying the data without losing too much of the information.
* Picking the same example of predicting **car’s price** through its **mileage, age & brand**, if **age** and **mileage** of the car are **related** to each other, then we will **merge** these into **one feature**.
* This is called ***feature extraction***.
* It often results in **better performance** & occupies **less memory space**.
* **Anomaly detection:** Detection of **unwanted** information or **outliers**.
* **Novelty detection:** These algorithms unlike **anomaly detection** algorithms, are **sensitive** to anomalies & **can’t** perform well when they are present in data.



* **Association rule learning:** Finding out **interesting relations** among attributes.
* Like finding out that people buying **bread** also buying **jam** & thus one can keep them close in the shop.

Semi-supervised learning:-

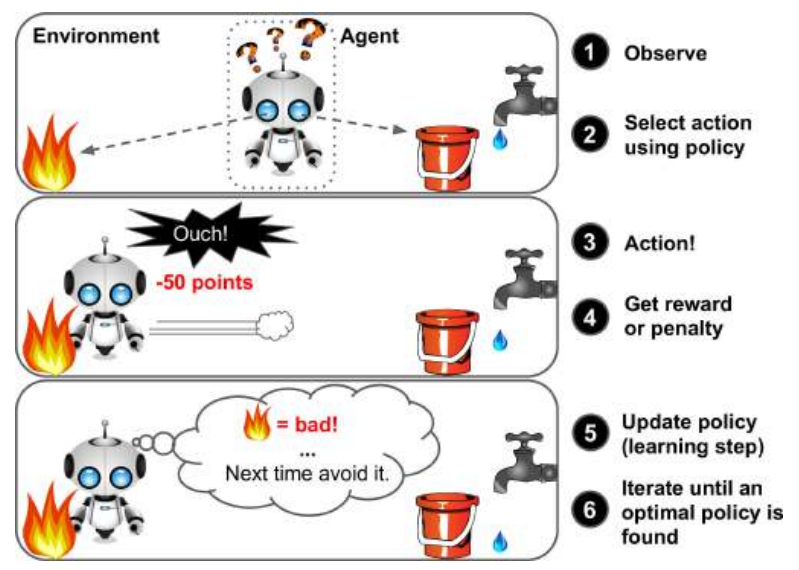
* Deals with **partially labelled** data & **mostly labelled** data.
* For example, **Google photos** recognizes same faces in various photos (**unsupervised**).
* But we have to externally tell it the name of each person in one photo, then it **labels** rest of the photos as per that input (**supervised**).
* Actually, its **unsupervised** part uses **clustering**.
* Sometimes user has to clear some clusters **manually** to prevent it detecting alike looking people into same cluster.



* ***Deep Belief Networks (DBNs)*** are based on ***Restricted Boltzmann Machines (RBMs)***.
* **RBMs** are trained as an **unsupervised algorithm** & later whole model is covered with some **supervised learning** techniques.

Reinforcement learning:-

* The model is **rewarded** or **penalised** with points & based on its learning it shapes its policy to avoid repeating its mistakes.
* This explains the whole concept of **robotics**.
* We call the **learning model/ system** as ***agent*** here.



**Batch & Online Learning**

* It is a **criterion** used to know if a model can learn **incrementally** from newly fed data.

Batch learning:-

* The machine is fed **huge amount of data** from starting & consumes a lot of computing time, energy & probably bandwidth too.
* Then it can perform the tasks it was made for **without** learning anymore.
* It is also known as ***offline learning***.
* But it requires whole learning process to be **repeated** from scratch when an **update** is required.

Online learning:-

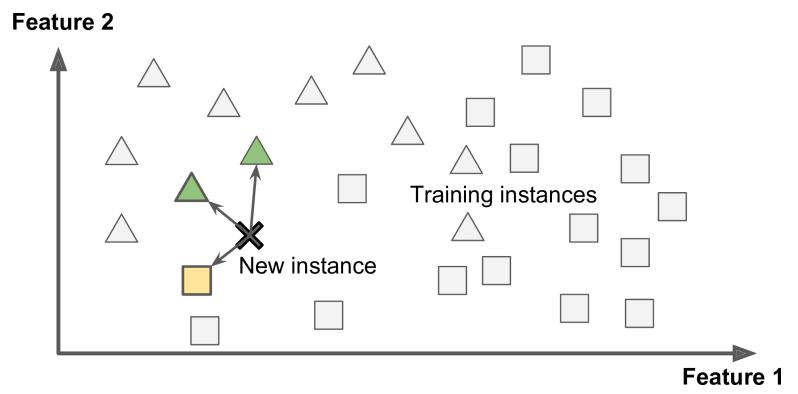
* System is trained by being fed data **sequentially** or in **small groups** of data (***mini-batches***).
* So, it is both **faster** and **cheaper**.
* For example, **stock prices** which are updated daily.
* **Online learning** also includes **discarding old data** which are **no** longer required.
* **Out-of-core learning:** Learning **externally** on data which **can’t** fit on a machine’s **main memory**.
* **Online learning** uses an important criterion called **learning rate**.
* If learning rate is **high**, the system **quickly adapts** to new data but **forgetting the old** ones.
* But when the learning rate is **low**, the system becomes **lazy** & computes data **slowly** along with being **insensitive/ tolerant** to **outliers** & **noisy data**.
* When **unnecessary** or **bad data** are fed to the system, its performance can **decline**.
* Like someone **spamming same search** in search engine to make it appear on the list of most searched searches (example of **anomaly**).
* At such time, the maintainers must switch its **learning mode off** before finding the problem & fixing it.
* Or better, **revert** to a date when the system was working perfectly.

**Instance Based & Model Based Learning**

* This criterion is used to see the model’s ability to ***generalize***.
* ***Generalization*** simply means being able to **categorize/ predict** the new examples well.

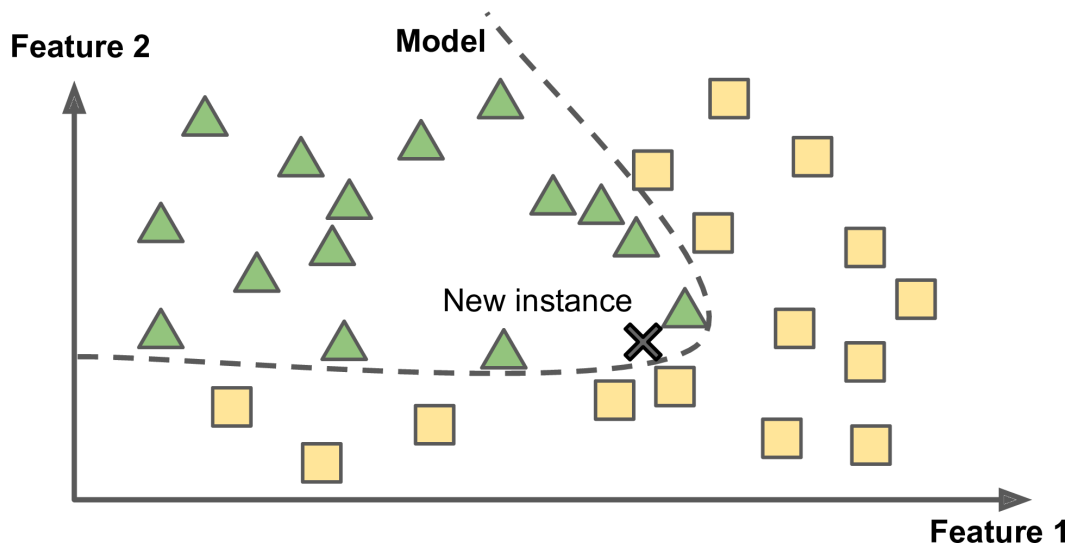
Instance based learning:-

* Learns from how user has **labelled** the **old data** to know how to label **new** **ones**.
* For example, **K-Nearest neighbour** groups nearby **datapoints** into same group.
* Like when a set of **spam emails** are given to the system, it finds out how user has **flagged** some emails as **spams** & then groups emails into **spams** and **hams**.

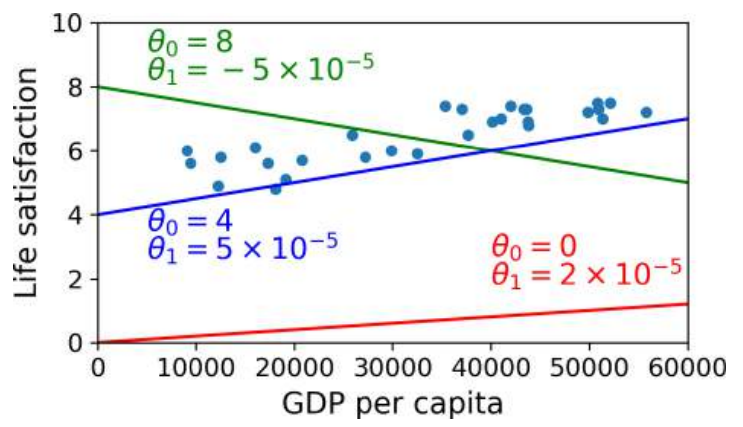


Model based learning:-

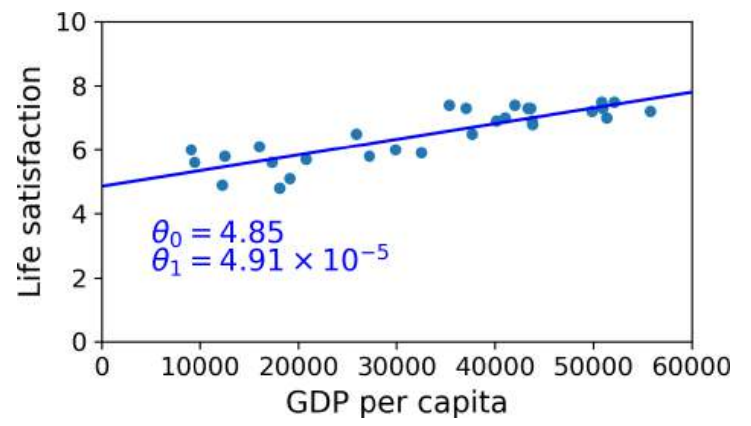
* Creating a model to predict what the target data would be (like **regression algorithms**).
* **Instance based learning** judges as per **past behaviour** & while **model based** learns the **algorithm** to predict.



* **Model selection:** Making a **linear function** to represent data on the graph.
* **Utility/ fitness function:** Used to measure the **performance** of a function.
* **Cost function:** Used to measure the **magnitude of distance** between training datapoints & function line on graph.
* We guess the **best fitting line** for our function & measure **how bad** it is using **cost function**.



* Then the **most fitting line** is selected.
* **Linear regression** easily solves this problem without **manually** guessing.



* Using **more attributes** can be helpful to achieve **accuracy**, if they are **correlated**.
* But it is preferred to perform **feature extraction** for better performance.
* Even a type of **regression curve** must be chosen carefully to get desired results from predictions (like sometimes **polynomial regression** might be better).
* **Inference:** Using a model to make **predictions**.

**Challenges Faced in ML (In a Nutshell)**

* Insufficient training data
* Non-representative training data
* Poor quality data
* Irrelevant features
* Overfitting the training data
* Underfitting the training data

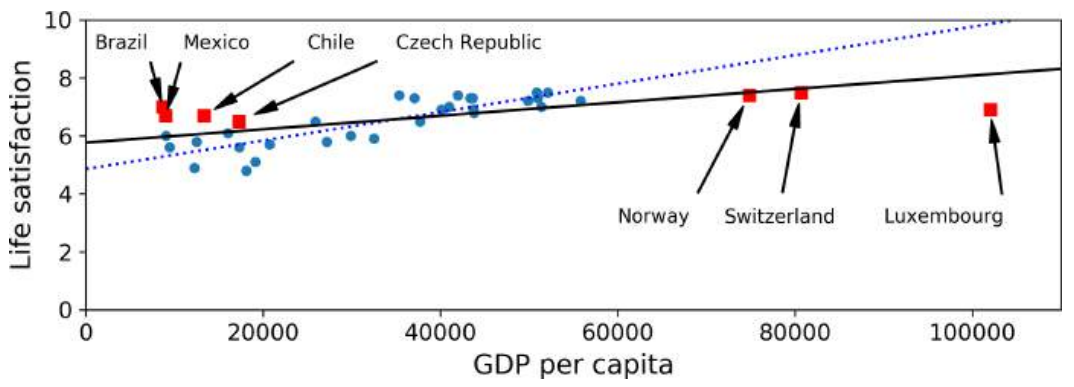
**Challenges Faced in ML (Brief)**

Insufficient training data:-

* Insufficient training data can cause problems.
* Even simple tasks require **thousands** of examples & complex tasks like **recognition systems** might require **millions** of examples!
* **Natural language disambiguation:** Distinguishing & writing similar words at appropriate places (like **to**, **too** & **two**).
* Having **more data** promises **more accuracy** when compared to writing **more efficient** algorithms.
* But that **doesn’t** mean to not focus on algorithms, as data are **expensive** to get.

Non-representative training data:-

* Means **not** including some part of data may alter the **nature of model**, or give **different/unexpected** insights.



* Just like how the graph above represents.
* The **dotted blue line** is old data & **solid black line** is after adding some extra data.
* Adding few more countries to dataset changes the **line’s angle** significantly.
* And we may come to know how some **poor countries** **are** **happier** than **some of the rich countries**, that can prove our **older conclusion** wrong.
* **Sampling noise:** Getting non-representative data due to **small dataset** size.
* **Sampling bias:** Getting non-representative data with **large dataset**, due to **flawed sampling method**.
* **Non-response bias:** Biased introduced due to **ruling out** data giving **different** insights.

Poor quality data:-

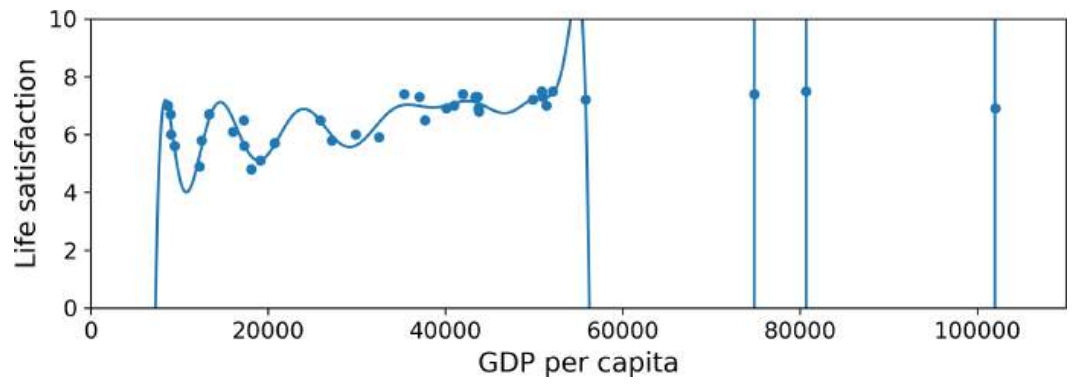
* This can **inhibit** the system from detecting the **actual pattern** among data.
* The trainer has to decide if they have to **remove** or **fill** any missing attribute values.
* In such case, the trainer might train **2 models**.
* One with the missing **attributes filled** & another with such **rows** **removed**.

Irrelevant features:-

* A good ML project includes ***feature engineering***, which is:
  + Feature selection
  + Feature extraction
  + Gaining new features from data

Overfitting the training data:-

* **Overfitting:** Bad generalization caused because the model is **too complex** to understand a **simple pattern** in dataset & presence of **outliers**.
* Like if a taxi driver in UK rips you off, then you might conclude that all taxi drivers in UK are thieves, which might **not** be true.



* If the training data is **noisy**, the model will try to find some **irrelevant pattern** in it.
* Like for the old **life-satisfaction model**, it may detect that countries with **w** in their **names** have life-satisfaction **greater than 7**.
* **Regularization:** Modifying a model to be simpler by **removing overfitting**.
* It helps build **better regression slopes** by removing the **outliers**.
* Amount of **regularization** is controlled by ***hyperparameters***.
* **Hyperparameter** is set to a learning algorithm, **not** to the model.
* Too high of a **hyperparameter** will result in a line with **no slope** (parallel to **x-axis**).

Underfitting the training data:-

* Opposite of **overfitting**, the model is **too simple** to understand a **complex data**.
* Methods to fix underfitting:
  + Selecting a more complex model
  + Feature engineering
  + Reducing constraints like hyperparameters

**Testing & Validation (In a Nutshell)**

* Hyperparameter tuning
* Data mismatch

**Testing & Validating (Brief)**

* Rather than waiting for your model users to report any error, try **splitting** the pre-existing data into **training** & **testing sets**.
* Then measure the **accuracy** of the predictions on **testing set**.
* **Generalization error:** Error rate on new cases.
* Also known as ***out-of-sample errors***.
* **Low training error** but **high generalization error** means **overfitting** training data.
* The **training** & **testing sets** must be split **carefully** into proper quantity.
* For example, though the common practice is to keep **80% training set** & **20% testing set**, for a dataset with **10 million** examples **1%** means **100,000 examples**.
* Which must be enough for **testing** & estimating **generalization error**.

Hyperparameter tuning:-

* When confused in choosing one among multiple models, one method is to **train all** & **compare** how well they **generalize**.
* Then you have to set **hyperparameter** to reduce effects of **overfittings**.
* But which **hyperparameter** value will be best for your model?
* One way to know that is by **training various copies** of the model on different **hyperparameter** values & select the best one.
* But that **doesn’t** guarantee **less generalization error** as the **training set** & **testing set** are part of same dataset.
* That means the testing set becomes **adaptable** as per that particular dataset only.
* This problem can be solved by ***holdout validation*** method.
* In **holdout validation**, a part of training set is reserved for **testing** called ***validation set***.
* After various models are trained on **training set** (excluding **validation set**), the best **hyperparameter** is chosen to be tested on **validation set**.
* After testing, the whole model is **re-trained** on the **training set** & **validation set**.
* Then finally it is tested on **testing set** for estimating **generalization error**.
* **Small validation** sets will result in **imprecise models**.
* **Large validation** sets will result in **small training set**.
* So rather, each model must be trained on different fragments of **validation sets** & then the **average** of its performances will determine its **overall performance**.
* But the problem is that the **training time** is **multiplied** by the number of **validation sets**.

Data mismatch:-

* Sometimes the **training data** may **not** represent the ***production data*** perfectly.
* **Production data:** New instances fed by **end users**.
* For example, if an application is created to recognize species of flower, then the photos might be taken from **different angles**.
* So, if **training data** **doesn’t** contain photos from **different angles**, then it might lead to ***data mismatch***.
* To solve this, one must ensure that the **training set** & **validation set** contain the type of data expected to be seen in **production** (preferably **shuffled** **without duplicates** in both the sets).
* Sometimes your model trained at **different places** might show **different performance**, like one trained on **custom pictures** & another on **web pictures**.
* This might be caused due to either **overfitting** or **data mismatch** between both **training sets**.
* To know which of the both is causing the problem, we **divide** the **training set** further into two parts. The second part is called ***train-dev set***.
* If the model performs well, then it is **not** **overfitting** but **data mismatch**.
* To address this **data mismatch**, we can **preprocess** the model trained on **web pictures** to look like the pictures in **mobile app**.
* But if it was to be an **overfitting**, then steps as discussed earlier must be taken to address it.

**No Free Lunch Theorem**

* Says that models work on **assumptions** on data.
* Means a model is just a **simplified version** of the whole **training data**.
* For example, a **linear model** assumes that the data is **linear** & the datapoints around the line is just **noise**.

