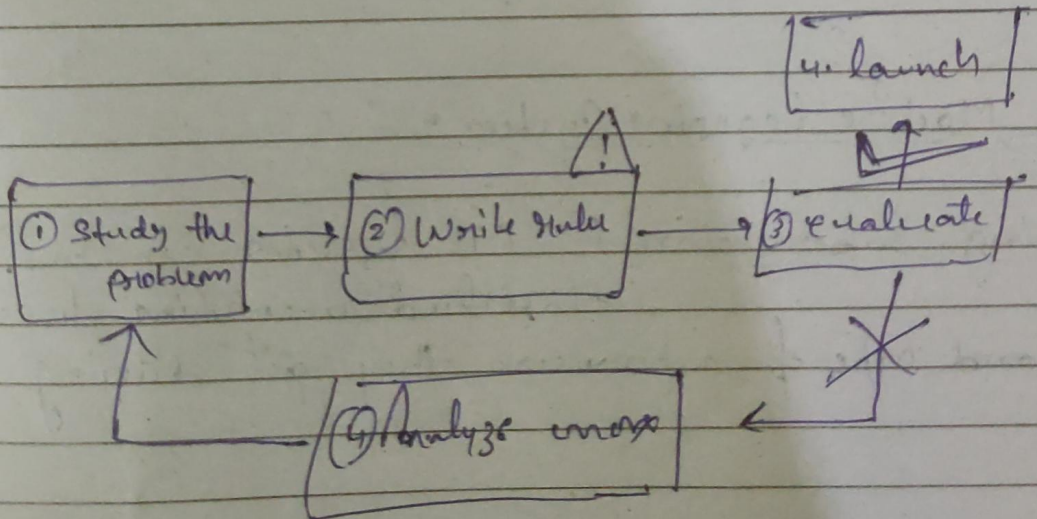
Chapter - 1

The machine learning landscape

Machine learning :- Machine learning is the science (and art) of programming computers so they can learn from data.

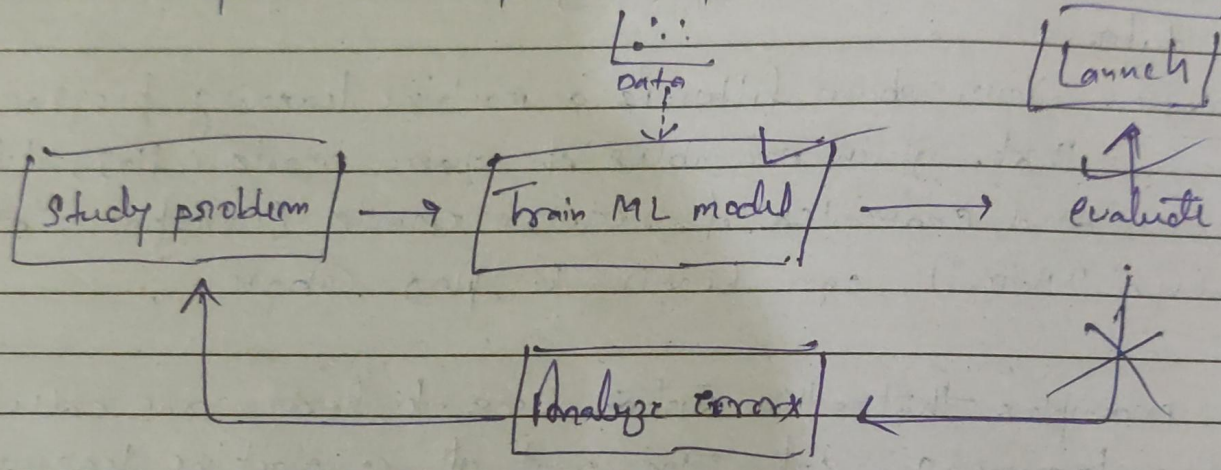
Example :- Your spam filter is a machine learning program that, given examples of spam emails (flagged by users) and examples of regular emails (nonspam, also called "ham"), can learn to flag spam.

The examples that the system uses to learn are called the training set. 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.



(1) Traditional Approach

A spam filter based on machine learning technique 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.



Digging into large amounts of data to discover hidden patterns is called data mining.

→ Types of Machine Learning Systems :-

① Training Supervision :- ML systems can be classified according to the amount and type of supervision they get during training.

(i) Supervised Learning :- In supervised learning, the training set you feed to the algorithm includes the desired solutions, called labels.

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.

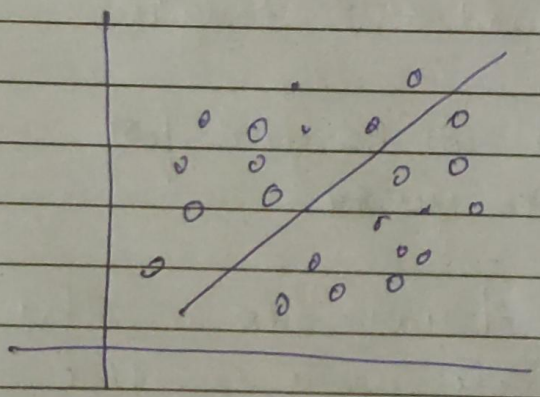
Another typical task is to predict a target numeric value, such as the price of a car, given a set of features (milage, age, brand, etc). This sort of task is called regression. To train the system, you need to give it many examples of cars, including both their features and their targets.

Note that some regression models can be used for classification as well, and vice versa.

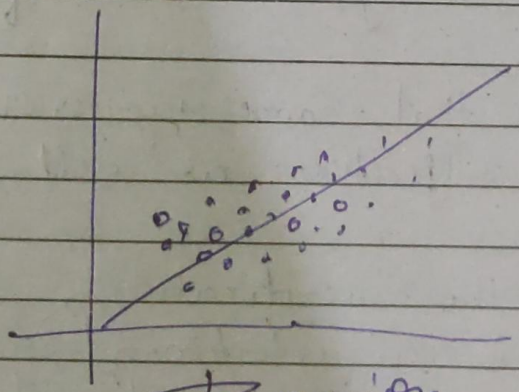
In the classification, the target variables in the problem statement are discrete: while in the regression, the target variables are continuous. Example, Spam classification, Disease prediction, etc. like problems are solved used classification algorithms. problems like House Price Prediction, Rainfall prediction like problems are solved using regression Algorithms.

* Regression Vs. Classification :- Regression and classification algorithms are supervised learning algorithms. Both the algorithms are used for prediction in machine learning and work with the labeled datasets. But the difference between both is how they are used for different machine learning problems.

The main difference between Regression and classification algorithms that regression algorithms are used to predict the continuous value such as price, salary, age, etc; and classification algorithms are used to predict/classify the discrete value. such as Male or Female, True or False, etc.



classification



Regression

(ii) Unsupervised learning :- In unsupervised learning, ~~as~~ ~~you might guess~~, the training data is unlabeled.

For example :- Say you have a lot of data about your blog's visitors. You may want to run a clustering algorithm to try to detect groups of similar visitors. At no point do you tell the algorithm which group a visitor belongs to: it finds those connections without your help.

Visualization algorithms are also good examples of unsupervised learning, you feed them a lot of complex and unlabeled ~~data~~ data, and they output a 2D or 3D representation of your data that can easily be plotted.

Yet another important unsupervised task is anomaly detection - for example, detecting unusual credit card transactions to prevent fraud, catching manufacturing defects or automatically removing outliers from a dataset before feeding it to another learning algorithm.

(iii) Semi-supervised learning :- Since labeling data is usually time-consuming and costly, you will often have plenty of unlabeled instances, and few labeled instances. Some algorithms can deal with data that's partially labeled. This is called semi-supervised learning.

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, 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.

(iv) Self-supervised learning :- Another approach to machine learning involves actually generating a fully labeled dataset from a fully unlabeled one. Again, once the whole dataset is labeled, any supervised learning algorithm can be used. This approach is

called self-supervised learning. 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 regards it's closer to supervised learning.

* (4) Reinforcement learning :→ Reinforcement learning is a very different beast. The learning systems, called an agent in this context, can observe the environment, select and perform actions, and get rewards in return (or penalties in the form of negative rewards). It must then learn by itself what is the best strategy, called a policy, to get the most reward over time. A policy defines what action the agent should choose when it is in a given situation.

* (2) Batch Versus Online Learning :—

* Main Challenges of Machine Learning :-

(i) Insufficient Quantity of Training Data :-

→ Data Preprocessing :- There are the major steps involved in data preprocessing:-

(i) Data cleaning (ii) ~~Data Integration~~ (iii) Data reduction
(iv) ~~Data transformation.~~

(i) Data cleaning :- Real-world data tend to be incomplete, noisy, and inconsistent. Data cleaning routines attempt to fill in missing values, smooth out noise while identifying outliers, and correct inconsistencies in the data.

(a) Missing Values :- Imagine that you need to analyze the company's sales and customer data. You note that many tuples have no recorded value for several attributes such as customer income, etc. There are following methods to handle the problem :-

(i) Ignore the tuple.
(ii) Fill in the missing value manually.

(iii) Use a global constant to fill in the missing value.
(iv) Use a measure of central tendency.
(v) Use the most probable value to fill in the missing value.

(b) Noisy Data :→ Noise is a random error or variance in a measured variable. There are many statistical description technique (e.g. boxplots and scatter plots) and methods of data visualization can be used to identify outliers, which may represent noise.

(i) Binning :→ Binning methods smooth a sorted data value by consulting its "neighborhood", that is, the values around it. The sorted values are distributed into a number of "buckets", or bins. Because binning methods consult neighborhood of values, they perform local smoothing.

(ii) Data smoothing can also be done by regression, a technique that confirms data values to a function.

(iii) Outlier analysis :- Outliers may be detected by clustering, for example, where similar values are organized into

groups, or "clusters". Intuitively, values that fall outside of the set of clusters may be considered outliers.

(ii) Data Reduction :→ Data reduction techniques can be applied to obtain a reduced representation of the data set that is much smaller in volume, yet closely maintains the integrity of the original data.

Data reduction strategies include dimensionality reduction, numerosity reduction, and data compression.

(a) Dimensionality reduction is the process of reducing the number of random variables or attributes under consideration. It include wavelet transforms and principal component analysis.

(b) Numerosity reduction techniques replace the original data volume by alternative, smaller forms of data representation.

(c) In data compression, transformations are applied so as to obtain a reduced or "compressed" representation of the

original data. If the original data can be reconstructed from the compressed data without any information loss, the data reduction is called lossless.

Unit :- Basic Python

* Python is a type-inferred language, so you don't have to explicitly define the variable type. It automatically knows that "abc" is a string and declares the str variable as a string.

```
str = "abc"
print(str)
```

* Python literals or constant :-
  - (i) Numeric literals
  - (ii) String literals
  - (iii) Boolean literals
  - (iv) Character literals
  - (v) Special literals
    eg:- value = None
  - (vi) Collection literals
    - list
    - tuple
    - dictionary
    - set

* Python Data type :-

(i) Numeric Data type :- int, float, complex.

signed integers

floating decimal (15 decimal places)

* In python we can easily represent number in binary, hexadecimal and octal number systems by placing a prefix before that.

Number System	
Binary	0b or 0B
Octal	0o or 0O
hexadecimal	0x or 0x

↓ zero.

```
print(0b1101011)  #prints 107
```

* Operations like addition, subtraction convert integers to float implicitly (automatically), if one of the operands is float.

* We can also use built-in func^{n} like int(), float() and complex() to convert between types explicitly.

* Python list :- Python lists are just like dynamically sized arrays. Lists need not be homogeneous always which makes it the most powerful tool in python. A single list may contain DataTypes like integers, String, as well as list, tuple, etc. Lists are mutable, and hence, they can be altered even after their creation.

Exp:- List = [1, 2, 3]

→ Basic methods for list :-

(i) len(list) = return the size of the list
(ii) .append() = Only one element at a time can be added to the list by using the append() method. at the end of list.
(iii) insert() = For the addition of elements at the desired position, insert() method is used.
(iv) extend() = This method is used to add multiple elements at the same time at the end of the list.
(v) remove() = Only remove the first occurrence of the searched element.

* Python tuples :→ Python tuple is a collection of objects separated by commas. The main difference between tuple and list is Python tuple is immutable, unlike the python list which is mutable.

→ Creating Python tuples :-
(1) using round brackets
(2) with one item
(3) Tuple Constructor.

→ Tuples are immutable and ordered and allow duplicate values. Some characteristics of Tuples in python :-

- We can find items in a tuples.
- One can't add items to a tuple once it is created
- Tuples can't be appended or extended.
- We can't remove items from a tuple once it is created.

→ Tuples in Python provide two ways by which we can access the elements of a tuple :-

① Positive Index — a[1]

② Negative Index — a[-1]

→ Remove Individual tuple elements is not possible, but we can delete the whole Tuple using Del keyword.

* Python String :- It is an immutable data type, meaning that once you have created a string, you can't change it. We can String can be created using single quotes, double quotes or even triple quotes.

Individual characters of a string can be accessed by using the method of indexing.

Python string can also allow the slicing for accessing a range of characters in the string.

→ As python strings are immutable in nature, we can't update the existing string. We can only assign a completely new value to the variable with the same name.

→ Python Sets :→ Python sets is an unordered collection of data types that is iterable, mutable, and has no duplicate elements.

The major advantage of using a set, as opposed to a list, is that it has a highly optimized method for checking whether a specific element is contained in the set.

~~A python set can't have mutable~~ A python set contains only unique elements but at the time of set creation, multiple duplicate values can also be passed. Set items can't be accessed by referring to an index, since sets are unordered the items has no index.

Date
Page No.

* Frozen Sets :→ Frozen sets in Python are immutable objects that only support methods and operators that produce a result without affecting the frozen set. For example:-

```
String = ('H', 'e', 'l', 'l', 'o')
Fset1 = frozenset(String)
print(Fset1).
```

* Dictionaries in Python :→ A python dictionary is a data structure that stores the values in Key: value pairs.
Python dictionaries are essential for efficient data mapping and manipulation in programming.
Values in a dictionary can be of any data type and can be duplicated, whereas keys can't be repeated and must be immutable.
A dictionary can also be created by the built-in function dict().

* Python Multiset :→ Multiset package is similar to the python set but it allows elements to occur multiple times. Implementation can be based on dictionary elements to their multiplicity in the multisets.

→ Features of Python Multi-set :-

1. An a unordered collection of element
2. Hashable just like in a set.
3. It supports the same methods as set.
4. It supports set operation
5. Multisets are mutable so we can change the element using update() methods.

* Python PIP :→ pip is the package manager for python packages. We can use pip to install packages that do not come with python.

Python pip comes pre-installed on 3.4 or older versions of python.

* Python Numpy :→ Numpy is the fundamental package for scientific computing in python. It is a python library that provides a multi-dimensional array object, various derived objects.

* Python Pandas :→ Pandas is a powerful and open source python library. used for data manipulation and analysis. Pandas is well-suited for working with tabular data, such as spreadsheets or SQL tables.

Pandas generally provide two data structures for manipulating data. They are :-

(i) Series

(ii) Dataframe

(i) Pandas Series :- It is a one-dimensional labeled array capable of holding data of any type.

(ii) Pandas DataFrame :- Pandas Dataframe is a two-dimensional data structure with labeled axes (rows and columns). It is created by loading the datasets from existing storage (which can be a SQL database, a csv file, or an Excel file).

→ Python Shallow Copy and Deep Copy and Copy using "=" operator :→

"="

In python, we use operator to create a copy of an object, but it only creates a new variable that shares the reference of the original object.

For example :-

```
old = [1, 2, 3, 4]
new = old
print(old)
```

```
print(new)
print( id(old))
print( id(new))
```

Output:-

```
1, 2, 3, 4
1, 2, 3, 4
140673330446
140673330446
```

In python, there are two ways to create copies :

1) Shallow Copy
2) Deep Copy

We use the copy module of python for shallow and deep copy operations.

1) Shallow Copy :→ A shallow copy creates a new object which stores the reference of the original elements.
So, a shallow copy doesn't create a copy of nested objects, instead it just copies the reference of nested objects.

For example :- ~~import copy~~

```
import copy
old = [1, 2, 3, 4]
new = copy.copy(old)
print(old)
print(new)
print("adding new element into old")
old.append(5)
print(old)
print(new)
print("updating an element in old")
old[1] = 9
print(old)
print(new)
```

Output:-

```
[1, 2, 3, 4]
[1, 2, 3, 4]
adding new element into old
[1, 2, 3, 4, 5]
[1, 2, 3, 4]
updating an element in old
[1, 9, 3, 4, 5]
[1, 9, 3, 4]
```

Both of the lists changes because, both shares the reference of same nested objects.

Deep Copy :→ A deep copy creates a new object and recursively adds the copies of nested objects present in the original elements.

For example :-

```
import copy
old = [1, 2, 3, 4]
new = copy.deepcopy(old)
print(old)
print(new)
old[1] = 9
print(old)
print(new)
```

Output :- [1, 2, 3, 4]
[1, 2, 3, 4]
[1, 9, 3, 4]
[1, 2, 3, 4]

Unit - 2

* Machine learning is best suited for :—

① Problems for which existing solutions require a lot of fine-tuning or long lists of rules.

② Complex problems for which using a traditional approach yields no good solution.

③ Fluctuating environment

④ Getting insights about complex problems and large amounts of data.

↱ Batch Versus Online learning :↰

(i) Batch learning :↰ In batch learning, the system is incapable of learning incrementally: it must be trained using all the available data. This will generally take a lot of time and computing resources, so it is typically done offline. First the system is trained, and then it is launched into production and runs without learning anymore; it just applies what it has learned. This is also called offline learning.

(→ Incremental learning)

(i) Online learning :- In online learning, you train the system incrementally by feeding it data instances sequentially, either individually or in small groups called mini-batches.

Online learning is useful for systems that need to adapt to change extremely rapidly. It is also a good option if you have limited computing resources;

Online learning algorithms can be used to train models on huge datasets that can't fit in one machine's main memory (this is called out-of-core learning). The algorithm loads part of the data, runs a training step on the data, and repeats the process until it has run on all of the data.

→ Instance-Based v/s Model-Based learning :-

(i) Instance-Based learning :- In 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

(ii) Model-based learning :- Another way to generalize from a set of examples is to build a model of these examples and then use that model to make predictions. This is called model-based learning.

→ Main challenges of Machine learning :-

(i) Insufficient Quantity of Training Data :-

Machine learning takes a lot of data for most machine learning algorithms to work properly. Even for very simple problems you typically needs thousands of examples.

(ii) Non-representative Training Data :- In order to generalize well, it is crucial that your training data be representative of the new cases you want to generalize to.

By using a non-representative training set, you trained a model that is unlikely to make accurate predictions,

It is crucial to use a training set that is representative of the cases you want to generalize to.

This is often harder than it sounds: if the sample is too small, you will have sampling noise (i.e., non-representative data as a result of chance), but even very large samples can be non-representative if the sampling method is flawed. This is called sampling bias.

→ Examples of sampling bias: Suppose you want to build a system to recognize funk music videos. One way to build your training set is to search for "funk music" on youtube and use the resulting videos. But, in reality, the search results are likely to be biased toward popular artists and if you live in Brazil you will get a lot of "funk carioca" videos.

(iii) Poor-Quality Data: Obviously. If your training data is full of errors, outliers, and noise, it will make it harder for the system to detect the underlying patterns, so your system is less likely to perform well.

(iv) Irrelevant Features :→ Your system will only be capable of learning if the training data contains enough relevant features and not too many irrelevant ones.

A critical part of the success of a machine learning project is coming up with a good set of features to train on. This process is called feature engineering, involves the following steps :-

(i) Feature Selection

(ii) Feature extraction

(v) Overfitting the Training Data :→ Overgeneralizing is something that we humans do all too often, and unfortunately machines can fall into the same trap if we are not careful. In machine learning this is called overfitting: it means that the model performs well on the training data, but it does not generalize well. An overfit model can give inaccurate predictions and cannot perform well for all types of new data. Overfitting happens due to several reasons, such as :

- The training data size is too small and does not contain enough data samples to accurately represent all possible input data values.
- The training data contains large amounts of irrelevant information
- The model trains for too long on a single sample set of data.

Constraining a model to make it simpler and reduce the risk of overfitting is called regularization. The amount of regularization to apply during learning can be controlled by a hyperparameter. A hyperparameter is a parameter of a learning algorithm. As such, it is not affected by the learning algorithm itself; it must be set prior to training and remains constant during training.

(vi) Underfitting the Training Data :- Underfitting is the opposite of overfitting: it occurs when your model is too simple to learn the underlying structure of the data. You get underfit models if they have not trained for the appropriate length of time on a large number of data points. Underfit models experience high bias —

they give inaccurate results for both the training data and test set.

→ Data Preprocessing :→ Incomplete, inaccurate and inconsistent data are commonplace properties of large real-world database and data warehouses. Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model.
There are the major steps involved in data preprocessing :—

(i) Data cleaning (ii) Data integration (iii) Data reduction
(iv) Data transformation.

(i) Data cleaning = ~~Raw final Ex~~

(ii) Data integration :→ Careful integration can help reduce and avoid redundancies and inconsistencies in the resulting data set. This can help improve the accuracy and speed of the subsequent data mining ~~and ex~~ process.

(c) Redundancy and Correlation Analysis :→ Redundancy is another important issue in data integration. An attribute may be redundant if it can be "derived" from another attribute

or set of attributes. Some redundancies can be detected by correlation analysis. Given two attributes such analysis can measure how strongly one attribute implies the other, based on the available data. For nominal data, we use the χ^2 test and for numeric attributes, we can use the correlation co-efficient and covariance.