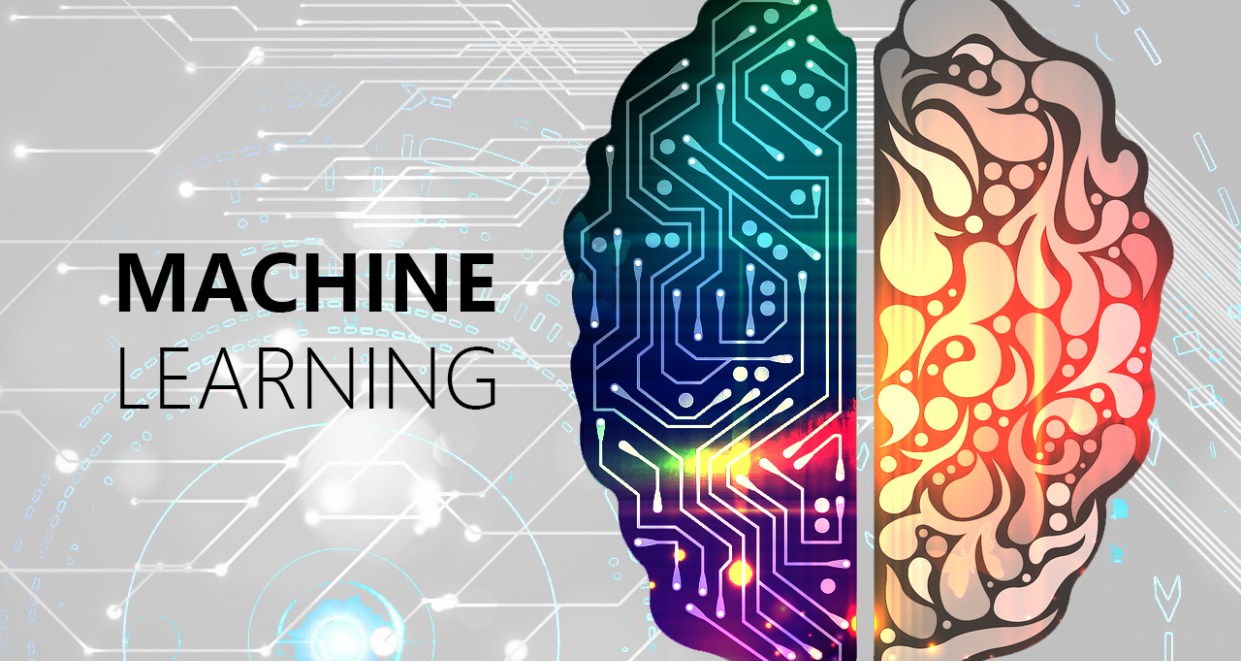
**Machine Learning – With Python**

”The field of machine learning is concerned with the question of how to construct computer programs that automatically improve with experience” – Page xv, Machine Learning, 1997

**Table of contents:**

* [Notes](#_Notes:)
* [Scatterplot Diagram](#_To_create_a)
* [Histogram Diagram](#_To_create_a_1)
* [Useful Tips](#_Useful_Tips:)
* [Common Challenges in ML](#_Common_Challenges_in)
* [Applications of ML](#_Applications_of_Machine)
* [Strengths and Weaknesses of Using Python](#_Strengths_and_Weaknesses)
* [Methods of Machine Learning](#_Methods_of_Machine)
* [Recommended ML Algorithms to Learn](#_Recommended_ML_Methods)
* [Skills needed for Machine Learning](#_Skills_needed_for)
* [Skills needed for Deep Learning](#_Skills_needed_for_1)
* [Learning Tensorflow](#_Learning_TensorFlow:)
* [Learning Keras](#_Learning_Keras:)
* [Learning Pandas](#_Learning_Pandas:)
* [Learning Numpy](#_Learning_Numpy:)
* [Learning Scikit-Learn](#_Learning_Scikit-Learn:)
* [Learning Opencv](#_Learning_OpenCV:)
* [Learning Matplotlib](#_Learning_MatplotLib:_)
* [Learning Pytorch](#_Learning_PyTorch:_)
* [Learning Pyqt5 GUI](#_Learning_PyQT5_GUI:)
* [Core Learning Algorithms](#_Core_Learning_Algorithms:)
* [Areas to learn and what they are](#_Area’s_to_learn)
* [Resources](#_Resources)



# Notes: [To Top](#_top)

|  |  |  |  |
| --- | --- | --- | --- |
| Symbol | **Category** | WordBOOK | Explanation |
|  | **Calculus** | Mean | Average value in the dataset(1) |
|  | **Calculus** | Median | The middlemost value in the dataset(1) |
|  | **Calculus** | Mode | The value that is most common(1) |
| σ | **Calculus** | Standard Deviation | Is a number that describes how spread out the numbers in the dataset are. (1) |
| σ 2 | **Calculus** | Variance | A number that indicates how spread out values are, it can also be used to see how far away each value is from the mean value. (1) |
|  | **Calculus** | Histogram | A visual representation of the occurrences of all of the values in a dataset(1) |
|  | **Calculus** | Scatterplot | This is a diagram where each value is represented by a dot in a coordinate grid. If we had a cars speed and age and put both in arrays, it would tell us which car were fastest and how old that car was. Useful for detecting patterns in a dataset(1) |
|  | **Calculus** | Regression | It is used to try and find the relationship between variables, that relationship is then used to predict the outcome of future events(1) |
|  | **Calculus** | Linear Regression | It uses the relationship between the data-points to draw a straight line through it all and this line can be used to predict future values. (In machine learning, predicting the future values is really important) (1) |
|  | **Calculus** | Polynomial Regression | It uses the relationship between the data-points to draw the best possible line through the data-points. A good alternative if you can’t use Linear regression(1) |
|  | **Python** | Numpy | A mathematical and statistics library. .mean, .median, .random.uniform and so (1) |
|  | **Python** | Scipy | A scientific and mathematical library. .mode, (1) |
| R | **Calculus** | R | A number value that defines the relationship between datapoints, no relationship would be a 0, while 1 & -1 indicate a perfect / 100% relationship. Everything in between can be calculated to a percentage. The closer to 1 or -1 it is, the better it is for linear regression. If it is closer to 0, then chances are that it isn’t suited for Linear regression(1) |
| R2 | **Calculus** | R2 | R-squared, just like the R, is a relationship number, however in this case, it can only go to max 1, and the lowest is 0. This number doesn’t go into negative(1) |
|  | **Calculus** | Scaling: Standardization | **z = (x - u) / s**. Z = new value, X = original value, U = mean value, S = standard deviation. This is the formula used when you have to scale values because 2 sets of values are in different measurements and you want to compare them. (1) |
|  | **Machine Learning** | Train/Test Machine learning model evaluation | It is called the Train/Test method because you split the dataset into 2 sets, a training set and a testing set. The percentage is up to you. You *train* the model using the training set, and you *test* the model using the testing set. Training the model means creating the model and getting it ready for testing. Test the model means test the accuracy of the model(1) |
|  | **Machine Learning** | Decision Tree | A decision tree is a flow chart and it can help you make decisions based on previous experience. (1) |
|  | **Machine Learning** | Decision Tree: Gini | The Gini method uses this formula: **Gini = 1 – (x/n)2 – (y/n)2** .Where x is the number of positive answers, in the example dataset called(‘shows.csv’) this would be: “GO”, n is the number of samples, y is the number of negative answers. In the example this would be: “NO” which gives us this calculation: 1 – (7 / 13)2 – (6 / 13)2 = 0.497(1) |
|  | **Machine Learning** | Task (T) | Problem-wise this defines Task as a real-world problem to be solved. It can be anything from finding the best house prices in a specific location with other perimeters and such to calculating distances to a location. If we are talking machine learning tho, the definition of a task is different because it is difficult to solve ML based tasks by conventional programming approaches. A task T in ML can be: Classification, Regression, Structured Annotation, Clustering, Transcription and the likes. A task T in ML describes a task that is based on the process(2) |
|  | **Machine Learning** | Performance (P) | A ML algorithm is supposed to perform a task and gain experience with the passage of time. The measure which tells whether ML algorithm is performing as per expectation or not is performance (P). P is a quantitative metric that tells how a model is performing the task, T, using its experience, E. Metrics that pertain to the Performance category could be: Accuracy score, F1 score, Confusion matric, precision, recall, sensitivity and so on. (2) |
|  | **Machine Learning** | Experience (E) | Experience is the knowledge gained from the data points provided to the algorithm or model. Once provided with the dataset, the model will run iteratively and will learn some inherent pattern. The learning thus acquired is called experience(E). A fitting analogy in terms of human learning, we can think of this situation as in which a human being is learning or gaining some experience from various attributes like situation, relationships etc. Supervised, unsupervised and reinforcement learning are some ways to learn or gain experience. The experience gained by out ML model or algorithm will be used to solve the task T. (2) |
|  | **Machine Learning** | Venn Diagram | Machine Learning Model(2) |
|  | **Machine Learning** | Categorial Output | Unordered and discrete values |
|  | **Machine Learning** | Weight in Neural networks | This weight number is representative of the connection strength between the activation layer of neurons and the hidden layer of neurons. It also represents the amount of influence the given connection has on the result. The higher the number, the bigger the influence |
| *σ* | **Machine Learning** | Sigmoid Function | a(1) = *σ*(Wa (0) +b) | Is a representation of how positive the weighted sum is.  W = Weight.  a = Activation, this number is a number between 0.0-1.0 this represents how much the given neuron has been activated. The closer to 1, the more activated it has become, the closer to 0 the less activated it has become.  Bias = for inactivity, this is a number that is used to correct for how often the neural network gets this slightly wrong and therefore this number should be adjusted to fit for better accuracy.  This function is a squished down and shortened version of a function that would normally be 784 weights, 784 activation neuron numbers + bias. Therefore it could be shortened down to this short function(3) |
| ReLU | **Machine Learning** | Rectified Linear Unit, or ReLU for short | ReLU is a more speed efficient way of squishing down numbers in the hidden layer to a number between 1 and 0. This is done due to the output needing to be a number between 1 and 0 |
| C(w) | **Machine Learning** | Cost function explanation | Giving the computer a penalty with a number. This number is calculated based on HOW wrong it is, therefore. The higher the number, the more wrong it’s final guess was. Therefore the computer would meaningfully search for the lowest cost. The cost function output is the average cost of all the training data. The lower the number, the more accurate it actually is. (3) |
| C(w) | **Machine Learning** | Cost function algorithm | ∇C, then small step in -∇C direction, repeat. Or Adjusted Weights and biases. Essentially this reduces the cost of the output data. This number actually represents the relative importance of each weight and bias. Which change gives the most change? is the answer to that question (3) |
| ∇C | **Calculus** | Gradient | A number which is positive if it’s a positive direction, and negative if it’s a negative direction. The higher the positive number, the steeper the descent is, the higher the negative number the steeper the ascent is(3) |
|  | **Calculus** | Gradient Descent | A name for the action of continuously nudging the weight and bias numbers in an effort to get the lowest cost number. In a sense, this is like pushing a ball down a hill as effectively as possible, meaning with the most progress each time. (3) |
|  | **Calculus** | Stochastic Gradient Descent | Is a less elegant way of doing a Gradient Descent, instead of a ball smoothly rolling down a hill, this is a drunk man stumbling down the hill, they essentially arrive at the same place. The speed is just drastically different and the steps in between are a lot less accurate. But in this case, it’s the end result that we wanted(3) |
|  | **Machine Learning** | Mini-batches | Randomly dividing the training data into smaller batches, so as to increase computational speed, while sacrificing a little bit of accuracy. Computing gradient descent steps based on each of these mini batches, and then averaging these numbers(3) |
|  | **Machine Learning** | Backpropagation | Backpropagation is the algorithm for determining how a single training example would like to nudge the weights and biases. This goes without saying, but it will determine the local fastest way for it to get the most changes. Most bang for your buck you could say(3) |
|  | **Data Science** | Dataset: Standardization | Refers to preprocessing text. Like removing punctuation and html elements and generally getting it to just text(4) |
|  | **Data Science** | Dataset: Tokenization | Refers to splitting the text, typically done by splitting the text into individual words and such by splitting on whitespaces. These splits are called tokens(4) |
|  | **Data Science** | Dataset: Vectorization | Refers to converting tokens into numbers so they can be fed into a neural network(4) |
|  | **Data Science** | Train/test skew | When a dataset is skewed towards one side. Meaning the values lean toward one side of the dataset. It’s really visible if you graph the datapoints |
|  | **Machine Learning with Python** | TensorFlow | TensorFlow is Google’s own neural network algorithm |
|  | **TensorFlow** | .cache() | Keeps data in memory after it’s loaded off disk. This will ensure the dataset does not become a bottleneck while training your model. It also helps in cases where your dataset is too large to fit into your memory, you can use this method to create a performant on-disk cache |
|  | **TensorFlow** | .prefetch() | Overlaps data preprocessing and the model execution while training |
|  | **TensorFlow Word Embeddings** | Embedding Vector | A second approach you might try is to encode each word using a unique number. Continuing the example above, you could assign 1 to "cat", 2 to "mat", and so on. You could then encode the sentence "The cat sat on the mat" as a dense vector like [5, 1, 4, 3, 5, 2]. This approach is efficient. Instead of a sparse vector, you now have a dense one (where all elements are full).  There are two downsides to this approach, however:   * The integer-encoding is arbitrary (it does not capture any relationship between words). * An integer-encoding can be challenging for a model to interpret. A linear classifier, for example, learns a single weight for each feature. Because there is no relationship between the similarity of any two words and the similarity of their encodings, this feature-weight combination is not meaningful. |
|  | **TensorFlow** | Embedding Layer | This layer takes the integer-encoded data (from Vectorization) and looks up an embedding vector for each word-index. These vectors are learned as the model trains. The vectors add a dimension to the output array. The resulting dimensions are: (batch, sequence, embedding). |
|  | **TensorFlow** | GlobalAveragePooling1D Layer | Returns a fixed-length output vector for each example by averaging over the sequence dimension. This allows the model to handle input of variable length, in the simplest way possible. This is further explained in the Tensorflow.DeepComputerVision Pooling operations |
|  | **TensorFlow** | Dense Layer | Means that every neuron is connected to every neuron in the previous layer. Thus, it is densely connected |
|  | **TensorFlow** | Loss Function | It is essentially a Cost function, just from a different perspective |
|  | **Machine Learning** | Optimizer | Optimizers are algorithms or methods used to change the attributes of the neural network such as weights and learning rate to reduce the losses. Essentially optimizers are the specific algorithm used for the back propagation in the case of Loss functions, it’s used to most accurately and effectively get closer to the weights and biases needed for optimal results.(5) |
|  | **TensorFlow & Data Processing** | Normalization / Standardization | It is when you scale values down to between 0-1 to ensure more stable training. This is commonly done through a preprocessing layer. This is useful when you have a dataset that doesn't fit a normal distribution (Gaussian Distribution), after normalization the data distribution will normally look like a bell curve. The other side of the same coin is: Standardization AKA scaling, is a form of processing where you are changing the range of your data. Yes, that is supposed to be really vague, normally standardization will change the range to -1:1. Which means the lowest possible value is -1 and the highest possible value is 1. This would mean that the mean average is 0 and the standard deviation is 1 |
|  | **Machine Learning** | Epoch | An Epoch is a measure of time that is based on 1 run of the training set. When the training set has been run through once, it is considered 1 epoch. |
|  | **Data Loading** | CSV Data | A data format that is commonly used for machine learning. The format is usually: First row for all the column names, all of which are separated by a comma(,). Second row and after are used for data, each data corresponding for a column is separated by a comma(,). For example:  **Age,Experience,Rank,Nationality, Go**  **36,10,9,UK,NO.**  That is the format for data, but what if you want to write a comment? It is the same as in python, simply use a #. What about quotes? There you will use “ ”. |
|  | **Data Processing** | Data preprocessing | Preprocessing of data is about making sure the data is in the right format and the right scale so that the machine as well as the supervising personnel can make sense of the given data. If we can’t make sense of the data, chances are neither can the machine. The most important part of data preprocessing is the conversion of the given data to the format and scale needed. Often this is done by normalization. The needed output will be between 0 – 1 so it makes sense for the needed input to be so as well. |
|  | **Data Processing** | Binarization | This is a data preprocessing technique that makes the data into binary. Meaning 0 and 1. It does this by being given a threshold where it should decide. A threshold of 0.5 means that everything over 0.5 is given the value of 1, and everything under is given the value of 0 |
|  | **Data Processing** | L1 Normalizer | Sometimes defined as the normalization technique that modifies the dataset values in a way that in each row the sum of the absolute values will always be up to 1. AKA Least Absolute Deviations |
|  | **Data Processing** | L2 Normalizer | It is sometimes defined as the normalization technique that modifies the dataset values in a way that in each row the sum of the squares will always be up to 1 |
|  | **Data Processing** | Label Encoding | A lot of python data handling library functions (such as Sklearn’s functions) deal with data in the form of numbers. But a lot of datasets use strings as the labels, therefore we need to convert the strings into numbers. |
|  | **Data Processing** | Data Feature Selection |  |
|  | **TensorFlow.DeepComputerVision** | Filters | A filter is a pattern of pixels that the neural network has learned or is learning to recognize. Examples when working with images of animals could be: Cat ears and cat eyes. |
|  | **TensorFlow.DeepComputerVision** | Pooling operations | Examples would be: Max pooling, Min pooling and Average pooling. This operation is called this because you try to size down your filters by pooling the minimum values or max values into a smaller pixel grid. Like going from 3x3 to 2x2 |
|  | **TensorFlow.DeepComputerVision** | CNN Architecture (Convolutional Neural Network) | A common architecture for a CNN is a stack of Conv2D and MaxPooling2D layers followed by a few densely connected layers. The idea is that the stack of convolutional and MaxPooling layers extract the features from the image. Then these features are flattened and fed to densely connected layers that determine the class of an image based on the presence of features. |
|  | **TensorFlow.DeepComputerVision** | Convolutional Layer | This is the layer where a Filter is applied over the whole image bit by bit. This can also be applied to already outputted feature maps as well |
|  | **Machine Learning** | Dimensionality | Is the phenomenon where the idea of: Higher dimensions with low dimensional data presents a scarcity of data points. An example would be: You have 3 data points. One point at 0 meters, one at 1 meter and 1 at 2 meters. These data points are sufficient as long as you look at the meter scale (dimension), but if you zoom in (higher dimension) and look at the centimeter scale, then the 3 data points aren’t sufficient. As you would have 99 missing points of data suddenly. This means that there is an exponential increase in the volume of data points needed per dimensional zoom (moving to a higher dimensional data type) Another way to understand this is that, input features are referred to as dimensionality. Meaning the more columns (feature columns) you have, the higher the dimensionality, which poses the problem of: How relevant is the information to the process of predicting the outcome. If it isn’t useful, you would have to delete or at least not use that feature column. |
|  | **Data Science** | Data Augmentation | To avoid overfitting and also create a larger dataset from a smaller one, we can use a technique called Data Augmentation. This is simply performing random transformations on our images so that our model can generalize better. These transformations can be things like compressions, rotations, stretches and even color changes |
|  | **TensorFlow.PretrainedModels** | Freezing the base | The term freezing refers to disabling the training property of a layer. It simply means we won’t make any changes to the weights of any layer that are frozen during training. This is important as we don’t want to change the convolutional base that is already trained correctly. |
|  | **Deep learning Neural Networks** | Recurrent Neural Networks | Instead of being given all of the input at once, it is given one input at once and processes that and then spits out an output based on that. It then keeps track of the output and uses that the next time it is given input, and this loops around continuously adding more and more experience. That means if you give it a sentence after having trained it, it should begin to understand the context and grammar of that sentence based on the experience it has gained before |
|  | **NeuralNetworks.RNN** | LSTM (Long Short-Term Memory) | Same as a Simple Recurrent Layer (it’s just the formula for RNN in a layer) with something added on top. That something is a kind of library function that stores all the previous outputs and makes them accessible at any time in the algorithm. It essentially allows the algorithm to have a long list of outputs available to pull or read, but only when it is needed. This defeats the normal flaw of short term memory that exists with the Simple Recurrent Layer |
|  | **Neural Networks** | Checkpoints | Checkpoints are made because deep learning models can take hours, days or even weeks to train completely, and therefore it is a good practice to make checkpoints automatically so that if the last hour or so of training was flawed you can always start from the checkpoint before that and continue training. What’s worse than a training being invalid after having waited for it to finish for hours? Nothing. |
|  | **Reinforcement Learning** | Environment | In Reinforcement learning tasks we have a notion of environment. This is what our agent will explore. An example of an environment in the case of training an AI to play say a game of Mario would be the level we are training the agent on |
|  | **Reinforcement Learning** | Agent | An agent is an entity that is exploring the environment. Our agent will interact and take different actions within the environment. In our Mario example the Mario character within the game would be our agent |
|  | **Reinforcement Learning** | State | At all times our agent will be in what we call a state. The state simply tells us about the status of the agent. The most common example of a state is the location of the agent within the environment. Moving locations would change the agents state |
|  | **Reinforcement Learning** | Action | Any interaction between the agent and environment would be considered an action. For example, moving to the left of jumping would be an action. An action may or may not change the current state of the agent. In fact, the act of doing nothing is actually an action as well! The action of say, not pressing a key if we are using our Mario example |
|  | **Reinforcement Learning** | Reward | Every action that our agent takes will result in a reward of some magnitude (whether positive or negative). The goal of our agent will be to maximize it’s reward in an environment. Sometimes the ward will be clear, for example if an agent performs an action which increases their score in the environment we could way they’ve received a positive reward. If the agent were to perform an action which results in them losing score or possibly dying in the environment then they would receive a negative reward.  The most important part of reinforcement learning is determining how to reward the agent. After all the goal of the agent is to maximize its rewards. This means we should reward the agent appropriately such that it reaches the desired goal |
|  | **Reinforcement Learning** | Q-Learning |  |

# Useful Tips / Information Nuggets: [To Top](#_top)

|  |  |  |
| --- | --- | --- |
| 1 | Python uses snakecasing | For example: variable\_name |
| 2 | It is common to name the list of independent values with an upper-case X, and the list of dependent values with a lower-case y | This tip applies for using the Pandas library |
| 3 | ML model | It’s the same as an algorithm, just about perspective |
| 4 | Cost function | Local minimum = Doable  Global minimum = Crazy hard |
| 5 | Normalization Good practice | It is good practice to normalize features that use different scales and ranges. One reason this is important is because the features are multiplied by the model weights. So, the scale of the outputs and the scale of the gradients are affected by the scale of the inputs. Although a model might converge without feature normalization, normalization makes training much more stable |
| 6 | TensorFlow Model Tip #1 | Training a model with tf.keras typically starts by defining the model architecture. Sometimes you might want to use a sequential model or other times another model. |
|  |  |  |
|  |  |  |

# Common Challenges in Machine Learning: [To Top](#_top)

* **Quality of data:** Good quality data is one of the biggest challenges. Low quality data leads to problems related to data preprocessing and feature extraction. Another problem is that it can often be harder to get big and good quality data sets. Depending on the data you are handling of course
* **Time-Consuming task:**  Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval. In other words: The longer a task takes, the less effective and time efficient the ML model actually is. This can be good or bad, let’s say that you are trying to sort out data in a huge data set, time consuming tasks would be tantamount to failure. While if you we’re trying to translate large pieces of a book correctly, a time-consuming task wouldn’t be a problem since it would probably mean higher quality translations
* **Lack of specialist persons:** As Machine learning Technology is still really new, availability of expert resources is a tough job. (2)
* **Issue of overfitting and underfitting:** If the ML model is overfitting (meaning that its output is more than what was needed), or underfitting (meaning that its output is less than what was needed) it doesn’t represent the problem well
* **Curse of dimensionality:** Is the phenomenon where the idea of: Higher dimensions with low dimensional data presents a scarcity of data points. An example would be: You have 3 data points. One point at 0 meters, one at 1 meter and 1 at 2 meters. These data points are sufficient as long as you look at the meter scale (dimension), but if you zoom in (higher dimension) and look at the centimeter scale, then the 3 data points aren’t sufficient. As you would have 99 missing points of data suddenly. This means that there is an exponential increase in the volume of data points needed per dimensional zoom (moving to a higher dimensional data type) Another way to understand this is that, input features are referred to as dimensionality. Meaning the more columns (feature columns) you have, the higher the dimensionality, which poses the problem of: How relevant is the information to the process of predicting the outcome. If it isn’t useful, you would have to delete or at least not use that feature column.
* **Difficulty in deployment:** Sometimes the complexity and specificity of a ML model can make it quite difficult to deploy to real world problems in reality. While it in theory would be really good, it might not be as good when you try it in the real world.

# Applications of Machine learning: [To Top](#_top)

* **Emotion Analysis**(2)
* **Sentiment Analysis**(2)
* **Error detection and prevention**(2)
* **Weather forecasting and prediction**(2)
* **Stock market analysis and forecasting**(2)
* **Speech synthesis**(2)
* **Speech recognition**(2)
* **Customer segmentation**(2)
* **Object recognition**(2)
* **Fraud detection**(2)
* **Fraud prevention**(2)
* **Recommendation of products to customer in online shopping**(2)

# Strengths and Weaknesses of Python: [To Top](#_top)

***Strengths:***

* **Easy to learn and understand:** The syntax of python is simpler; therefore, it is relatively easy. The most important thing is to understand coding concepts and elements
* **Multi-purpose language:** It’s multi-purpose because it supports structured programming, object-oriented programming as well as functional programming
* **Huge number of modules:** It has a huge number of modules for covering every aspect of programming. If the module doesn’t exist, chances are you aren’t specific enough
* **Support of open source community:** Due to it being open source, the language has a huge amount of support from other developers who constantly fixes bugs and you can therefore infer that the language is quite robust and adaptive
* **Scalability:** It is a scalable programming language because it provides an improved structure for supporting large programs than shell-scripts

***Weaknesses:***

* **Slow:** Python is really powerful and can do so many things. But the one thing that python suffers on because of that is execution speed. Python is by far not the fastest executing programming language

# Methods of Machine Learning: [To Top](#_top)

*Supervised Learning:*

Using an algorithm to learn the mapping of the input to output. The main objective would be to approximate the mapping function so well that we can predict using only Input what the output variable would be for that new input data. This is done using labeled Datasets to train it to classify data or predict outcomes. Common types of supervised learning algorithms are: **Decision tree, Random forest, KNN, Logistic regression and so on.** Therefore, we can divide them into these 2 broad divisions: (2)

* **Classification:** Its key objective is to predict categorial output labels or responses for the given input data. This means each output response will belong to a specific class or category. For more detailed Explanations, refer to the chapter about Classification(2)
* **Regression:** Its key objective is to predict output labels or responses which are continuous numeric values, for the given input data. This means that is uses the relationship between inputs and corresponding continuous numeric output values to predict the future values. (2)

*Unsupervised Learning:*

As the name suggests, unsupervised learning is the opposite of supervised learning. Common examples are algorithms where it only has the input data, and has to figure out the pattern to the input data. Thus, creating output data. Examples of unsupervised learning algorithms includes: **K-means clustering, K-nearest neighbors and so on.** They can be divided into following broad divisions: (2)

* **Clustering:** One of the most useful unsupervised ML methods. They are used to find similarity as well as relationship patterns among data samples and then cluster those samples into groups having similarity based on features. The real-world example of clustering would be: grouping customers by their purchasing behavior(2)
* **Association:** Another useful unsupervised ML method. It is used to analyze large datasets to find patterns which further represent the interesting relationships between various items. It is also termed as: **Market Basked Analysis**(2)
* **Dimensionality Reduction:** It is used to reduce the number of feature variables for each data sample by selecting a set of principal or representative features. A question arises here is that why we need to reduce the dimensionality? The answer is: Curse of dimensionality(2)
* **Anomaly Detection:** It is used to find rare occurrences in data. It literally finds anomalies in datasets(2)

*Deep Learning:*

Deep Learning is put beside Unsupervised and Supervised Learning because it can be both of them or even some of the other categories. Which uses an algorithm with a web of interconnected entities known as nodes wherein each node is responsible for a simple computation. In this way, a neural network functions similarly to the neurons in the human brain. It is essentially a collection of algorithms. (3)

* + **Feed-*forward* Neural Network:**
  + **Recurrent Neural Network:**
  + **Convolutional Neural Network:**
  + **Modular Neural Networks:**
  + **Supervised Neural Network:** Using an algorithm to learn the mapping of the input to output. The main objective would be to approximate the mapping function so well that we can predict using only Input what the output variable would be for that new input data. This is done using labeled Datasets to train it to classify data or predict outcomes
  + **Unsupervised Neural Network:** As the name suggests, unsupervised learning is the opposite of supervised learning. Common examples are algorithms where it only has the input data(unlabeled data points), and has to figure out the pattern to the input data. Thus, creating output data.
  + **Reinforcement Neural Network:** It usually has restrictions. Sort of a guide towards the result. Let’s say you want it to learn to drive. Then you would make walls making it impossible for it to drive out of the map, or making it get game over for driving out of bounds. You reward it for going towards the goal, and reprimand it for going away from the goal
  + **Semi-supervised Neural Network** A method that falls between supervised and unsupervised. Its build by following one of 2 approaches:
    - The first and simple approach is to build the supervised model based on small amount of labeled and annotated data and then build the unsupervised model by applying the same to the large amounts of unlabeled data to get more labeled samples. Now, train the model on them and repeat the process. (2)
    - The second approach needs some extra efforts. In this approach, we can first use the unsupervised methods to cluster similar data samples, annotate these groups and then use a combination of this information to train the model. (2)

*Semi-supervised Learning:*

A method that falls between supervised and unsupervised. Its build by following one of 2 approaches:

* The first and simple approach is to build the supervised model based on small amount of labeled and annotated data and then build the unsupervised model by applying the same to the large amounts of unlabeled data to get more labeled samples. Now, train the model on them and repeat the process. (2)
* The second approach needs some extra efforts. In this approach, we can first use the unsupervised methods to cluster similar data samples, annotate these groups and then use a combination of this information to train the model. (2)

*Reinforcement Learning:*

These methods are different from previously studied methods and very rarely used also. In this kind of learning algorithms, there would be an agent that we want to train over a period of time so that it can interact with a specific environment. The agent will follow a set of strategies for interacting with the environment and then after observing the environment it will take actions regards the current state of the environment. A good example of a well-known Reinforcement learning algorithm is: Q-Learning. The following are the main steps of reinforcement learning methods −

* **Step 1** − First, we need to prepare an agent with some initial set of strategies.
* **Step 2** − Then observe the environment and its current state.
* **Step 3** − Next, select the optimal policy regards the current state of the environment and perform important action.
* **Step 4** − Now, the agent can get corresponding reward or penalty as per accordance with the action taken by it in previous step.
* **Step 5** − Now, we can update the strategies if it is required so.
* **Step 6** − At last, repeat steps 2-5 until the agent got to learn and adopt the optimal policies.

(2)

***(Based on learning ability)***

*Batch Learning:*

In many cases, we have end-to-end Machine Learning systems in which we need to train the model in one go by using whole available training data. Such kind of learning method or algorithm is called **Batch or Offline learning**. It’s called Batch or Offline learning because it is a one-time procedure and the model will be trained with data in one single batch. The following are the main steps of Batch learning methods −

* **Step 1** − First, we need to collect all the training data for start training the model.
* **Step 2** − Now, start the training of model by providing whole training data in one go.
* **Step 3** − Next, stop learning/training process once you got satisfactory results/performance.
* **Step 4** − Finally, deploy this trained model into production. Here, it will predict the output for new data sample.

(2)

***(Based on learning ability)***

*Online Learning:*

It is completely opposite to the batch or offline learning methods. In these learning methods, the training data is supplied in multiple incremental batches, called mini-batches, to the algorithm. Followings are the main steps of Online learning methods −

* **Step 1** − First, we need to collect all the training data for starting training of the model.
* **Step 2** − Now, start the training of model by providing a mini-batch of training data to the algorithm.
* **Step 3** − Next, we need to provide the mini-batches of training data in multiple increments to the algorithm.
* **Step 4** − As it will not stop like batch learning hence after providing whole training data in mini-batches, provide new data samples also to it.
* **Step 5** − Finally, it will keep learning over a period of time based on the new data samples.

(2)

***(Based on Generalization Approach)***

*Instance Based Learning:*

Instance based learning method is one of the useful methods that build the ML models by doing generalization based on the input data. It is opposite to the previously studied learning methods in the way that this kind of learning involves ML systems as well as methods that uses the raw data points themselves to draw the outcomes for newer data samples without building an explicit model on training data.

In simple words, instance-based learning basically starts working by looking at the input data points and then using a similarity metric, it will generalize and predict the new data points. (2)

***(Based on Generalization Approach)***

*Model Based Learning*

In Model based learning methods, an iterative process takes place on the ML models that are built based on various model parameters, called hyperparameters and in which input data is used to extract the features. In this learning, hyperparameters are optimized based on various model validation techniques. That is why we can say that Model based learning methods uses more traditional ML approach towards generalization. (2)

# Recommended ML Methods to study: [To Top](#_top)

Deep learning,

Reinforcement learning,

K-nearest Neighbors (KNN),

K-Means clustering,

Decision Tree,

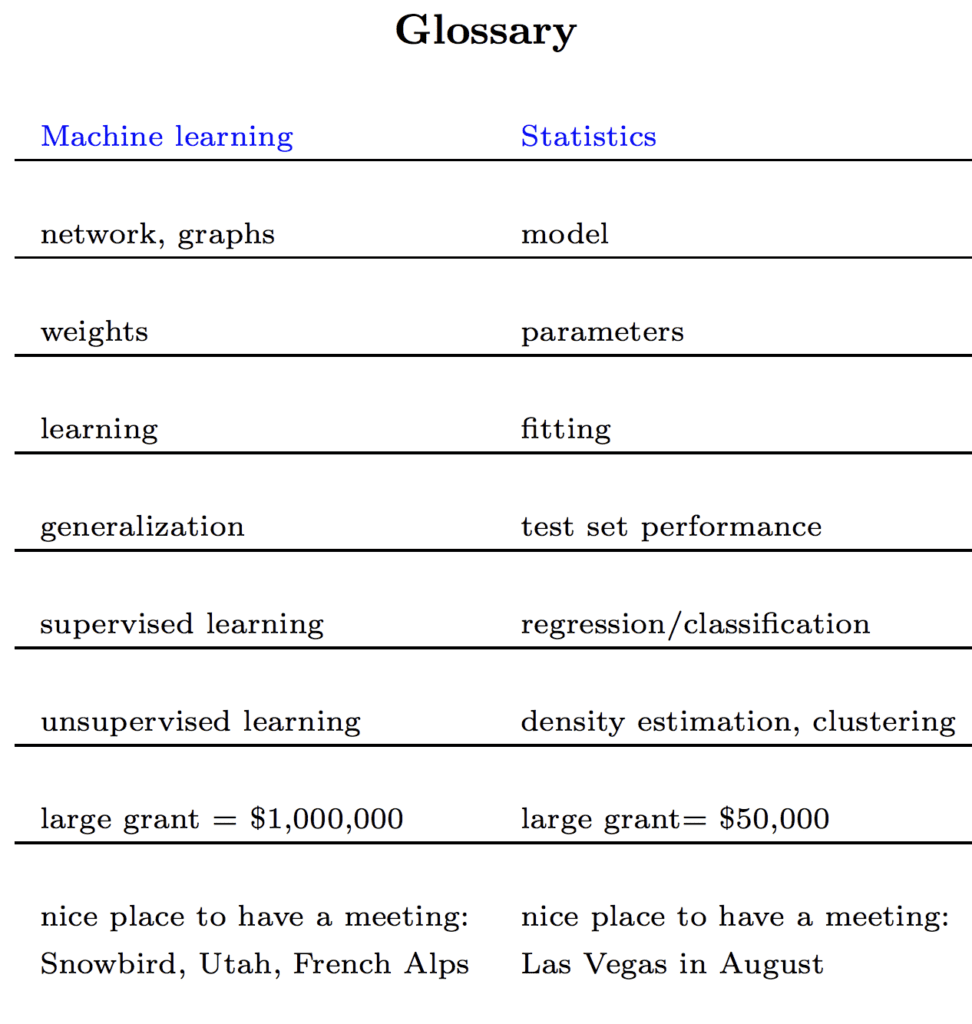
Random Forest,

Logistic Regression

# Skills needed for Machine learning: [To Top](#_top)

* **Programming**
* **Big Data**
  + What is Big Data: Big data is a term for data that is so large, fast or complex that traditional methods aren’t enough to handle it. The term refers to a field of data science that treats ways to analyze, systematically extract information from or otherwise deal with datasets that are too large for normal algorithms to handle
* **Knowledge of ML frameworks**
  + What are machine learning frameworks: Machine learning frameworks are basically what
* **Data structures**
  + What are Data structures: Data structures are your standard arrays, linked list, heap, stack, balanced tree, tree, binary tree and so on.
* **Algorithms**
  + What are Algorithms: Algorithms are well defined finite computer instructions. Typically used to solve a given problem
  + Algorithms to know: TensorFlow, Keras, Pandas, Numpy, Scikit-learn, OpenCV

# Skills needed for Deep learning: [To Top](#_top)

* **Data modelling**
  + What is data modelling: Data modelling is about making a visual representation of the neural network. You can also do this with machine learning algorithms. Data modelling is a great way to get algorithm or neural network ideas through to customers or clients in a company. It makes it easier to talk about and get the client to understand what you intend to make
* **Mathematics**
  + **Linear Algebra**
    - What is needed for use in Machine learning and Neural Networks: Understanding of Multivariable functions
  + **Multilinear Algebra**
    - What is needed for use in Machine Learning and Neural Networks: Understanding of Multilinear Tensors
  + **Statistics**
    - What do I need to know about statistics to use it for machine learning: Statistics is the practice of analyzing, collecting and handling large amounts of numerical data?
    - Predictive modelling: Is when you focus extensively only on making machine learning models that are good at completing one singular objective: Statistic Predictions
    - Statistical learning: AKA Applied statistics
  + 
* **Graph Theory**
  + What do I need to know to understand graphs and machine learnings relationship?
* **Programming**

# Learning TensorFlow: [To Top](#_top)

|  |  |  |  |
| --- | --- | --- | --- |
|  | **TensorFlow** | .cache() | Keeps data in memory after it’s loaded off disk. This will ensure the dataset does not become a bottleneck while training your model. It also helps in cases where your dataset is too large to fit into your memory, you can use this method to create a performant on-disk cache |
|  | **TensorFlow** | .prefetch() | Overlaps data preprocessing and the model execution while training |
|  | **TensorFlow Word Embeddings** | Embedding Vector | A second approach you might try is to encode each word using a unique number. Continuing the example above, you could assign 1 to "cat", 2 to "mat", and so on. You could then encode the sentence "The cat sat on the mat" as a dense vector like [5, 1, 4, 3, 5, 2]. This approach is efficient. Instead of a sparse vector, you now have a dense one (where all elements are full).  There are two downsides to this approach, however:   * The integer-encoding is arbitrary (it does not capture any relationship between words). * An integer-encoding can be challenging for a model to interpret. A linear classifier, for example, learns a single weight for each feature. Because there is no relationship between the similarity of any two words and the similarity of their encodings, this feature-weight combination is not meaningful. |
|  | **TensorFlow.DeepComputerVision** | Filters | A filter is a pattern of pixels that the neural network has learned or is learning to recognize. Examples when working with images of animals could be: Cat ears and cat eyes. |
|  | **TensorFlow.DeepComputerVision** | Pooling operations | Examples would be: Max pooling, Min pooling and Average pooling. This operation is called this because you try to size down your filters by pooling the minimum values or max values into a smaller pixel grid. Like going from 3x3 to 2x2 |
|  | **TensorFlow.DeepComputerVision** | CNN Architecture (Convolutional Neural Network) | A common architecture for a CNN is a stack of Conv2D and MaxPooling2D layers followed by a few densely connected layers. The idea is that the stack of convolutional and MaxPooling layers extract the features from the image. Then these features are flattened and fed to densely connected layers that determine the class of an image based on the presence of features. |
|  | **TensorFlow.DeepComputerVision** | Convolutional Layer | This is the layer where a Filter is applied over the whole image bit by bit. This can also be applied to already outputted feature maps as well |
|  | **TensorFlow** | Loss Function | It is essentially a Cost function, just from a different perspective |
|  | **TensorFlow & Data Processing** | Normalization / Standardization | It is when you scale values down to between 0-1 to ensure more stable training. This is commonly done through a preprocessing layer. This is useful when you have a dataset that doesn't fit a normal distribution (Gaussian Distribution), after normalization the data distribution will normally look like a bell curve. The other side of the same coin is: Standardization AKA scaling, is a form of processing where you are changing the range of your data. Yes, that is supposed to be really vague, normally standardization will change the range to -1:1. Which means the lowest possible value is -1 and the highest possible value is 1. This would mean that the mean average is 0 and the standard deviation is 1 |
|  | **TensorFlow.PretrainedModels** | Freezing the base | The term freezing refers to disabling the training property of a layer. It simply means we won’t make any changes to the weights of any layer that are frozen during training. This is important as we don’t want to change the convolutional base that is already trained correctly. |

# Learning Keras: [To Top](#_top)

Keras used to be a high-level API that sat on top of one of the three lower level neural network APIs and acted as a wrapper to these lower lever libraries. These libraries were referred to as Keras backend engines.

You could choose: TensorFlow, Theano or CNTK as the backend engine you’d like to work with that Keras would sit on top of. Now it’s integrated into Tensorflow

# Learning Pandas: [To Top](#_top)

# Learning Numpy: [To Top](#_top)

To create a random number “dataset”/array:

# Uniform: (There would be a near equal amount of every number, on a chart this would make a nearly straight line)

*Remember to import the correct library that needs to be used:*

**import NumPy**

**numpy.random.uniform(mean, deviation, amount of values)**

**Example would be:**

**x = numpy.random.uniform(0.0,5.0,250)**

# Normal: (This is also known as a bell curve, because the number that comes the most is the number in the middle of the lowest and highest numbers)

*Remember to import the correct library that needs to be used:*

**import NumPy**

**numpy.random.normal(mean, deviation, amount of values)**

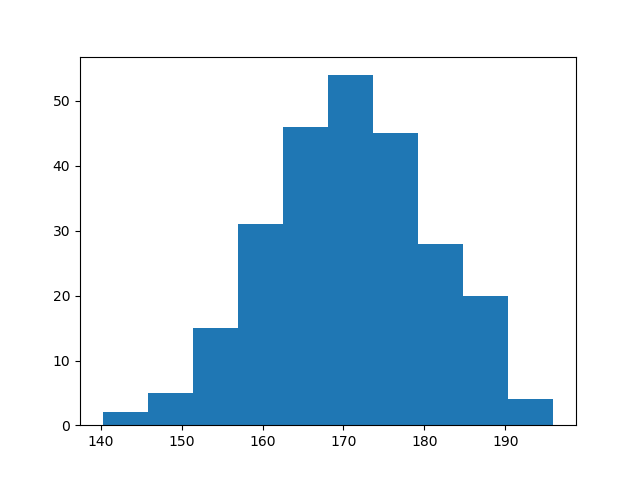
**Example would be:**

**x = numpy.random.normal(0.0,5.0,250)**

# Learning Scikit-Learn: [To Top](#_top)

# Learning OpenCV: [To Top](#_top)

# Learning Matplotlib: [To Top](#_top)

* What is a Scatterplot: a scatterplot is when you have data points created from 2 data structures (be it arrays or list and so on) on a diagram. These points are then scattered around according to the x and y axis. Meaning that the 2 data structures act like this for each point: x = data structure nr. 1, y = data structure nr. 2. By it doing this it scatters the points over a coordinate like system and thus we have a scatterplot
* What is a Histogram: A Histogram is a diagram/graph showing frequency distributions, like how often does this certain product get bought at doing the day. Showing that on a graph would be a histogram since it would detail the frequency that product got bought.
* Y and X Limit: by using ylim() or xlim() you can limit the range that the axis can go to. The format for the range input is: [StartRange, EndRange]
* Legend(): Shows the labels of each Axis, in case of a Pie chart it will show all of the labels.
* Y and X Labels: Allows you to label the X and Y axis. This is done by: ylabel() or xlabel() . The format for the label name is a string “ ”
* Grid: Adds grid lines to the diagram. Doesn’t work with Pie Charts of course. The format for the grid function is: grid()
* To create a Scatterplot diagram:

*Remember to import the correct library that needs to be used:*

**import matplotlib.pyplot as plt**

*Now you need 2 arrays*

**plt.scatter(array1,array2)**

**plt.show()**

* To create a Histogram diagram:

*Remember to import the correct library that needs to be used:*

**import NumPy**

**import matplotlib.pyplot as plt**

*Now you need 1 random dataset / array*

**plt.hist(array,number of histogrambars)**

**plt.show()**

# Learning PyTorch: [To Top](#_top)

# Learning PyQT5 GUI: [To Top](#_top)

# Core Learning Algorithms:

1. Linear Regression
   * What Linear Regression is: It draws a line through data-points. The line is drawn using the relationship between the data-points
   * What it’s used for: Is a method used mostly for predicting future values.
   * What to consider when choosing it: A linear regression might not fit, sometimes it would be better to look at a polynomial regression instead.
2. Classification
   * What it is: It’s the process of categorizing the given set of data into classes. This can be performed on both structured or unstructured data. It does this by predicting the class of given data-points. E.g. If you need to classify spam mail and non-spam mail.
   * What it’s used for: Classifying data into 2 or more classes/labels
   * What to consider when choosing it: Its function is to inherently classify data-points into classes/labels. If that’s not explicitly what you need, then you need to reconsider.
3. K-Means Clustering
   * What it is: It’s a method of clustering/grouping data under labels by assigning data-points to K-clusters which are labeled. The points are assigned based on which k-cluster they are closest to. The process then repeats when you want to add more data. This is smart when you want to group a massive amount of sales data that needs to be analyzed for business improvements.
   * What it’s used for: E.g. Sales data: You don’t need to look so tentatively to the biggest sales, but maybe more tentatively on the sales that are small and how to improve them. This is when it is smart to use clustering as it would group the data and you can then look more closely at it.
   * What to consider when choosing it: The algorithm doesn’t have any idea at all what anything is. It doesn’t care, it only groups data by values.
4. Hidden Markov Models
   * What it is: To understand this rather complicated algorithm we need to understand 3 terms: State, Observation, Transition. The Hidden Markov Model (HMM) is a relatively simple way to model sequential data. A hidden Markov model implies that the Markov Model underlying the data is hidden or unknown to you. More specifically, you only know observational data and not information about the states.
   * What it’s used for: Calculating the probability of a sequence of events happening, or calculating the probability of something transitioning
   * What to consider when choosing it:

# Area’s to learn and what they are [To Top](#_top)

|  |  |  |  |
| --- | --- | --- | --- |
| **The areas to learn** | **What the areas entail** | **What are the areas about** | **Reference Links** |
| **Framing** | Key Machine learning terminology | This area is about how you look at and understand the Subject matter terms | https://developers.google.com/machine-learning/crash-course/framing/ml-terminology |
| **Reducing Loss** | Gradient Descent, Learning Rate, Loss Functions and more | This area is about how you go about getting a more effective model. Meaning it’s more likely to be correct in predictions and such. | https://developers.google.com/machine-learning/crash-course/reducing-loss/an-iterative-approach |
| **Generalization** | Overfitting | This area is about how overfitting of a model is easily the best way to ruin any efficiency in the model | https://developers.google.com/machine-learning/crash-course/generalization/peril-of-overfitting |
| **Training and Test Sets** | Splitting Data | This area is about how you can split single datasets into testing and training sets. This is done if you don’t have separate training and testing sets | https://developers.google.com/machine-learning/crash-course/training-and-test-sets/splitting-data |
| **Feature Engineering** | Qualities of Good Features, Cleaning Data, Understanding why Feature engineering is done. | This area is about how to convert data to a feature vector you can use in tandem with model weights | https://developers.google.com/machine-learning/crash-course/representation/feature-engineering |
|  |  |  |  |
|  |  |  |  |

# Resources

(1) [Python Machine Learning – W3 School](https://www.w3schools.com/python/python_ml_getting_started.asp)

(2)[Machine Learning with Python – Tutorialspoint.com](https://www.tutorialspoint.com/machine_learning_with_python/index.htm)

(3)[Neural Networks Youtube Playlist – 3Blue1Brown](https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi)

(4)[Basic Text Classification - Tensorflow](https://www.tensorflow.org/tutorials/keras/text_classification#download_and_explore_the_imdb_dataset)

(5)[Overview of Various Optimizers – TowardsDataScience.com](https://towardsdatascience.com/overview-of-various-optimizers-in-neural-networks-17c1be2df6d5)