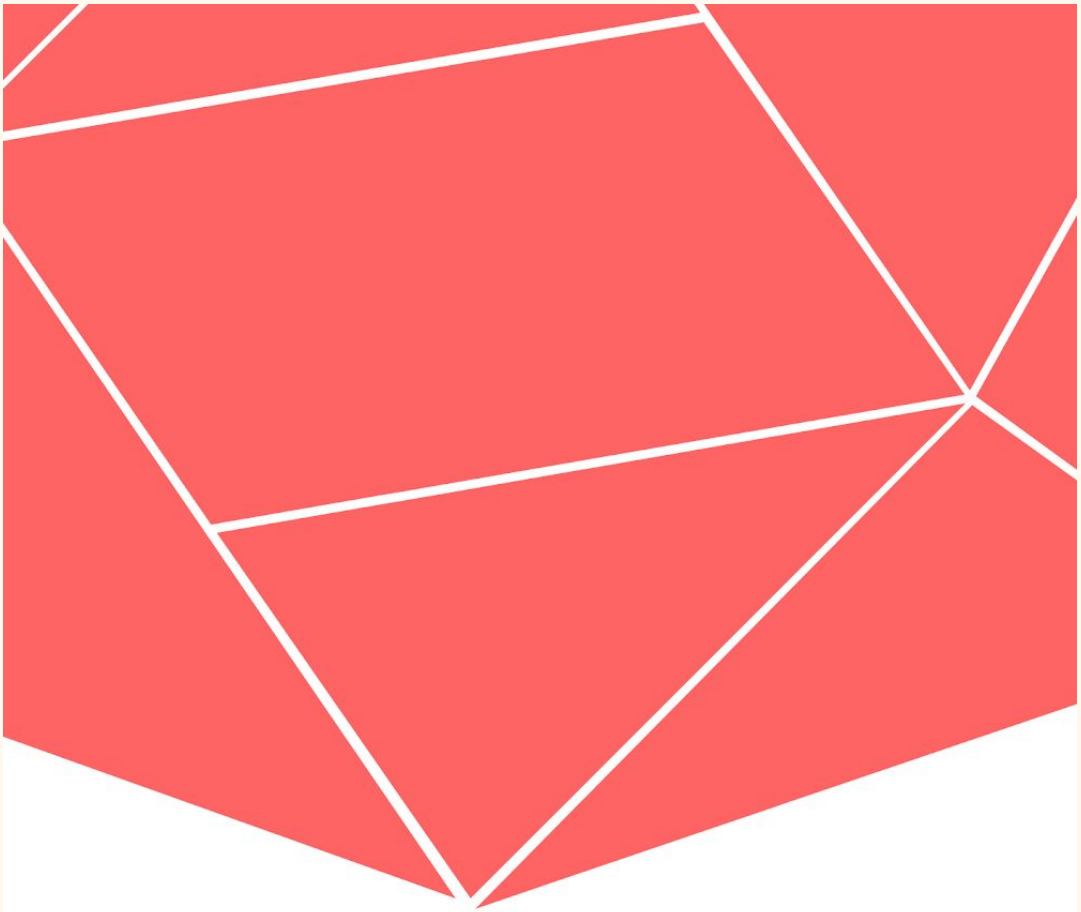


NeuralNet.tech

MACHINE LEARNING

THE MACHINE LEARNING E-BOOK

INDEX

What is Machine Learning?

Machine Learning is the most popular technique of predicting the future or classifying information to help people in making necessary decisions. Machine Learning algorithms are trained over instances or examples through which they learn from past experiences and also analyze the historical data. Therefore, as it trains over the examples, again and again, it is able to identify patterns in order to make predictions about the future.

Why Machine Learning?

The world today is evolving and so are the needs and requirements of people. Furthermore, we are witnessing a fourth industrial revolution of data. In order to derive meaningful insights from this data and learn from the way in which people and the system interface with the data, we need computational algorithms that can churn the data and provide us with results that would benefit us in various ways. Machine Learning has revolutionized industries like medicine, healthcare, manufacturing, banking, and several other industries. Therefore, Machine Learning has become an essential part of modern industry.

Types of Machine Learning

Machine Learning Algorithms can be classified into 3 types as follows:

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

Supervised learning

In Supervised Learning, the dataset on which we train our model is labeled. There is a clear and distinct mapping of input and output. Based on the example inputs, the model is able to get trained in the instances. An example of supervised learning is spam filtering. Based on the labeled data, the model is able to determine if the data is spam or ham. This is an easier form of training. Spam filtering is an example of this type of machine learning algorithm.

Unsupervised Learning

In Unsupervised Learning, there is no labeled data. The algorithm identifies the patterns within the dataset and learns them. The algorithm groups the data into various clusters based on their density. Using it, one can perform visualization on high dimensional data. One example of this type of Machine learning algorithm is the Principle Component Analysis. Furthermore, *K-Means Clustering* is another type of Unsupervised Learning where the data is clustered in groups of a similar order.

The learning process in Unsupervised Learning is solely on the basis of finding patterns in the data. After learning the patterns, the model then makes conclusions.

Reinforcement Learning

Reinforcement Learning is an emerging and most popular type of Machine Learning Algorithm. It is used in various autonomous systems like cars and industrial robotics. The aim of this algorithm is to reach a goal in a dynamic environment. It can reach this goal based on several rewards that are provided to it by the system.

It is most heavily used in programming robots to perform autonomous actions. It is also used in making intelligent self-driving cars. Let us consider the case of robotic navigation. Furthermore, the efficiency can be improved with further experimentation with the agent in its environment. This the main principle behind reinforcement learning. There are similar sequences of action in a reinforcement learning model.

Machine Learning Algorithms

Let us see some most common machine learning approaches:

1. Regression

Regression models are used extensively to predict values based on the variables that are dependent on several factors. The most

common example of regression is Linear Regression where there is a linear relationship or correlation between the predictor variable and the response variable. There are also other types of regression such as ARIMA regression that makes use of an auto-correlation regression model to forecast continuous values provided by the time-series data. They are used in forecasting the stock prices and other values that are based on time.

2. Decision Tree Learning

Decision Trees are a supervised type of machine learning algorithms. These trees are mainly used for predictive modeling. We create a decision tree that is able to take decisions based on user input. Decision Trees can be used for both regressions as well as classification. These trees are used to provide graphical outputs to the user based on several independent variables.

3. Support Vector Machines

Support Vector Machines or SVMs are machine learning algorithms that are used to classify data into two categories or classes. It is a type of supervised learning algorithms that makes use of several types of kernels to classify the data. Based on the prediction performed, it can categorize whether it falls into one class or any other class.

With the help of SVMs, one can perform both linear as well as non-linear classification. An SVM classifier divides the data into two classes using a hyperplane.

4. Association Rule Learning

Association Rule Mining is used for finding relationships between several variables that are present in the database. It is a type of *data mining technique* through which you can discover association between several items. It applies in sales industries mostly to predict if the customer will buy item Y if he has purchased the item X.

5. Artificial Neural Networks (ANN)

An Artificial Neural Network is an advanced form of machine learning technique. These neural networks are modeled after the human nervous system and are therefore called neural networks. There is a connection of several neurons which compute the information. These neurons capture the statistical structure and are therefore able to create a joint probability distribution over the input variables.

These neural networks are apt at finding patterns over large datasets. Neural Networks can perform classification as well as regression tasks with high accuracy. Furthermore, they eliminate the requirement for doing heavy statistical tasks in pre-processing as they are quite adequate in realizing patterns on their own.

6. Inductive Logic Programming

In this, logic programming forms the core part to produce a rule-like learning model. Inductive Logic Programming or ILP presents the input information, hypothesis as well as the background contextual knowledge in the form of several rules that have to be followed with

logic. It makes use of functional programs to carry out inductive programming to process hypotheses in part rules.

Training models are quite often used for developing this model which is then used to forge relationships between several variables.

7. Reinforcement Learning

The aim of *Reinforcement Learning* is to direct the agent towards maximizing rewards and reach its goal. This takes place in a dynamic environment where the agent has to chart its way to the goal through a series of trials and errors. Each time it takes a correct route, its profit is maximized and when it encounters a wrong approach, its profit is minimized. Reinforcement Learning is widely used in self-driving cars and autonomous robotics that require self-decision making capability.

Reinforcement Learnings are experimental in nature and through a series of trials are able to reach their goals with maximum accuracy (or rewards).

8. Clustering

In *clustering*, the observations are divided into groups or clusters. These clusters are formed based on similar data and have similar criteria. These criteria can be density or similar structure of the data. There are several clustering techniques that make use of different criteria to cluster the data. For instance, the distance between the data, the density of the data and graph connectivity are some of the criteria that define techniques for clustering in machine learning. Since

there is no labeled data or input-output mapping, this type of technique is an unsupervised machine learning procedure.

9. Similarity and Metric Learning

Similarity determination is one of the key functions of machine learning. In this form of learning, the ML model is provided a mix of similar as well as dissimilar data objects. The machine learning model learns to map similar objects together and learns a similarity function that allows it to group similar objects together in the future.

10. Bayesian Networks

A *Bayesian Network* is an acyclic directed graphical model. This model is also called DAG which represents the probability of several independent conditioned variables. One can illustrate the relationship between disease and symptoms. It can be used to compute the probabilities of various diseases. They can be used to find the diagnosis of several diseases through a calculated approach of listing probabilities of various factors that could have contributed towards it. More advanced forms of Bayesian Networks are Deep Bayesian Networks.

The basic principle behind the Bayesian Network is the Bayes theorem which is the most important part of the probability theory. With the help of Bayes Theorem, we determine the conditional probability of an event. This conditional probability is of a known

event. The conditional probability itself is the hypothesis. And, we calculate this probability based on the previous evidence.

$$P (A/B) = P (B/A)*P (A)/P (B)$$

Using a well-defined network of a connected graph, a user can make a DAG to model conditional dependencies

11. Representation Learning

In order to represent the data in a more structured format, we make use of representation learning. This formats the data efficiently so that the model can train better to provide accurate results. The representation of data is one of the key factors that can affect the performance of the machine learning method. This allows the algorithm to learn better from the data.

Using representation learning, algorithms are able to preserve the input data and essential information. Therefore, a model is able to capture most of the information during pre-processing.

Furthermore, the inputs present in pre-processing are able to gather data generating a defined distribution.

12. Sparse Dictionary Learning

In the method of Sparse Dictionary, a linear combination of basis functions as well as sparse coefficients are assumed. The elements of a sparse dictionary are called atoms. These atoms altogether compose a dictionary. It is an extension of representation learning. It

is used most widely in compressed sensing and signal recovery. In this method, we represent a datum as a linear combination of basis functions and then assume the coefficients to be sparse.

So, this was all in the latest Machine learning tutorial for beginners. Many of you might find the umbrella terms Machine learning, Deep learning, and AI confusing. So, here is some additional help; below is the difference between machine learning, deep learning, and AI in simple terms.

LIST OF ONLINE TUTORIALS

- http://www.datascienceassn.org/sites/default/files/Introduction%20to%20Machine%20Learning.pdf
- http://kioloa08.mlss.cc/files/smola.pdf
- http://ciml.info/dl/v0_8/ciml-v0_8-all.pdf
- https://www.tutorialspoint.com/machine_learning/machine_learning_tutorial.pdf
- https://www.researchgate.net/publication/328614973_Machine_Learning_For_Beginners
- https://ai.stanford.edu/~nilsson/MLBOOK.pdf
- https://www.ibm.com/downloads/cas/GB8ZMQZ3
- https://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning/understanding-machine-learning-theory-algorithms.pdf
- http://deeplearning.net/tutorial/deeplearning.pdf
- https://www.guru99.com/machine-learning-tutorial.html

Machine Learning Project Ideas

1. Machine Learning Gladiator

We're affectionately calling this "machine learning gladiator," but it's not new. This is one of the fastest ways to build *practical* intuition around machine learning.

The goal is to take **out-of-the-box models** and apply them to different datasets. This project is awesome for 3 main reasons:

First, you'll build intuition for model-to-problem fit. Which models are robust to missing data? Which models handle categorical features well? Yes, you can dig through textbooks to find the answers, but you'll learn better by seeing it in action.

Second, this project will teach you the invaluable skill of prototyping models quickly. In the real world, it's often difficult to know which model will perform best without simply trying them.

Finally, this exercise helps you master the **workflow** of model building. For example, you'll get to practice…

- Importing data
- Cleaning data
- Splitting it into train/test or cross-validation sets
- Pre-processing

- Transformations
- Feature engineering

Because you'll use out-of-the-box models, you'll have the chance to focus on honing these critical steps.

Check out the sklearn (Python) or caret (R) documentation pages for instructions. You should practice **regression**, **classification**, and **clustering** algorithms.

Tutorials

- Python: sklearn – Official tutorial for the sklearn package
- Predicting wine quality with Scikit-Learn – Step-by-step tutorial for training a machine learning model
- R: caret – Webinar given by the author of the caret package

Data Sources

- UCI Machine Learning Repository – 350+ searchable datasets spanning almost every subject matter. You'll definitely find datasets that interest you.
- Kaggle Datasets – 100+ datasets uploaded by the Kaggle community. There are some really fun datasets here, including PokemonGo spawn locations and Burritos in San Diego.
- data.gov – Open datasets released by the U.S. government. Great place to look if you're interested in social sciences.

2. Play Money Ball

In the book Moneyball, the Oakland A's revolutionized baseball through analytical player scouting. They built a competitive squad while spending only 1/3 of what large market teams like the Yankees were paying for salaries.

First, if you haven't read the book yet, you should check it out. It's one of our favorites!

Fortunately, the sports world has a ton of data to play with. Data for teams, games, scores, and players are all tracked and freely available online.

There are plenty of fun machine learning projects for beginners. For example, you could try…

- **Sports betting…** Predict box scores given the data available at the time right before each new game.
- **Talent scouting…** Use college statistics to predict which players would have the best professional careers.
- **General managing…** Create clusters of players based on their strengths in order to build a well-rounded team.

Sports is also an excellent domain for practicing **data visualization** and **exploratory analysis**. You can use these skills to help you decide which types of data to include in your analyses.

Data Sources

- Sports Statistics Database – Sports statistics and historical data covering many professional sports and several college ones. Clean interface makes it easier for web scraping.
- Sports Reference – Another database of sports statistics. More cluttered interface, but individual tables can be exported as CSV files.
- cricsheet.org – Ball-by-ball data for international and IPL cricket matches. CSV files for IPL and T20 internationals matches are available.

3. Predict Stock Prices

The stock market is like candy-land for any data scientists who are even remotely interested in finance.

First, you have many types of data that you can choose from. You can find prices, fundamentals, global macroeconomic indicators, volatility indices, etc… the list goes on and on.

Second, the data can be very granular. You can easily get time series data by day (or even minute) for each company, which allows you think creatively about trading strategies.

Finally, the financial markets generally have short feedback cycles. Therefore, you can quickly validate your predictions on new data.

Some examples of beginner-friendly machine learning projects you could try include…

- **Quantitative value investing…** Predict 6-month price movements based fundamental indicators from companies' quarterly reports.
- **Forecasting…** Build time series models, or even recurrent neural networks, on the delta between implied and actual volatility.
- **Statistical arbitrage…** Find similar stocks based on their price movements and other factors and look for periods when their prices diverge.

Obvious disclaimer: Building trading models to practice machine learning is simple. Making them profitable is extremely difficult. Nothing here is financial advice, and we do not recommend trading real money.

Tutorials

- Python: sklearn for Investing – YouTube video series on applying machine learning to investing.
- R: Quantitative Trading with R – Detailed class notes for quantitative finance with R.

Data Sources

- Quandl – Data market that provides free (and premium) financial and economic data. For example, you can bulk download end-of-day stock prices for over 3000 US companies or economic data from the Federal Reserve.

- Quantopian – Quantitative finance community that offers a free platform for developing trading algorithm. Includes datasets.
- US Fundamentals Archive – 5 years of fundamentals data for 5000+ U.S. companies.

4. Teach a Neural Network to Read Handwriting

Neural networks and deep learning are two success stories in modern artificial intelligence. They've led to major advances in image recognition, automatic text generation, and even in self-driving cars.

To get involved with this exciting field, you should start with a manageable dataset.

The **MNIST Handwritten Digit Classification Challenge** is the classic entry point. Image data is generally harder to work with than "flat" relational data. The MNIST data is beginner-friendly and is small enough to fit on one computer.

Handwriting recognition will challenge you, but it doesn't need high computational power.

To start, we recommend with the first chapter in the tutorial below. It will teach you how to build a neural network from scratch that solves the MNIST challenge with high accuracy.

Tutorial

- Neural Networks and Deep Learning (Online Book) – Chapter 1 walks through how to write a neural network from scratch in Python to classify digits from MNIST. The author also gives a very good explanation of the intuition behind neural networks.

Data Sources

- MNIST – MNIST is a modified subset of two datasets collected by the U.S. National Institute of Standards and Technology. It contains 70,000 labeled images of handwritten digits.

5. Investigate Enron

The Enron scandal and collapse was one of the largest corporate meltdowns in history.

In the year 2000, Enron was one of the largest energy companies in America. Then, after being outed for fraud, it spiraled downward into bankruptcy within a year.

Luckily for us, we have the Enron email database. It contains 500 thousand emails between 150 former Enron employees, mostly senior executives. It’s also the only large public database of real emails, which makes it more valuable.

In fact, data scientists have been using this dataset for education and research for years.

Examples of machine learning projects for beginners you could try include…

- **Anomaly detection…** Map the distribution of emails sent and received by hour and try to detect abnormal behavior leading up to the public scandal.
- **Social network analysis…** Build network graph models between employees to find key influencers.
- **Natural language processing…** Analyze the body messages in conjunction with email metadata to classify emails based on their purposes.

Data Sources

- Enron Email Dataset – This is the Enron email archive hosted by CMU.
- Description of Enron Data (PDF) – Exploratory analysis of Enron email data that could help you get your grounding.

6. Write ML Algorithms from Scratch

Writing machine learning algorithms from scratch is an excellent learning tool for two main reasons.

First, there's no better way to build true understanding of their mechanics. You'll be forced to think about every step, and this leads to true mastery.

Second, you'll learn how to translate mathematical instructions into working code. You'll need this skill when adapting algorithms from academic research.

To start, we recommend picking an algorithm that isn't too complex. There are dozens of subtle decisions you'll need to make for even the simplest algorithms.

After you're comfortable building simple algorithms, try extending them for more functionality. For example, try extending a vanilla **logistic regression** algorithm into a **lasso/ridge regression** by adding regularization parameters.

Finally, here's a tip every beginner should know: Don't be discouraged is your algorithm is not as fast or fancy as those in existing packages. Those packages are the fruits of years of development!

Tutorials

- Python: Logistic Regression from Scratch
- Python: k-Nearest Neighbors from Scratch
- R: Logistic Regression from Scratch

7. Mine Social Media Sentiment

Social media has almost become synonymous with "big data" due to the sheer amount of **user-generated content.**

Mining this rich data can prove unprecedented ways to keep a pulse on opinions, trends, and public sentiment. Facebook, Twitter, YouTube, WeChat, WhatsApp, Reddit… the list goes on and on.

Furthermore, every generation is spending even more time on social media than their predecessors. This means that social media data is will become even more relevant for marketing, branding, and business as a whole.

While there are many popular social media platforms out there, **Twitter is the classic entry point for practicing machine learning.**

With Twitter data, you get an interesting blend of data (tweet contents) and meta-data (location, hashtags, users, re-tweets, etc.) that open up nearly endless paths for analysis.

Tutorials

- Python: Mining Twitter Data – How to perform sentiment analysis on Twitter data
- R: Sentiment analysis with machine learning – Short and sweet sentiment analysis tutorial

Data Sources

- Twitter API – The twitter API is a classic source for streaming data. You can track tweets, hashtags, and more.
- StockTwits API – StockTwits is like a twitter for traders and investors. You can expand this dataset in many interesting ways

by joining it to time series datasets using the timestamp and ticker symbol.

8. Improve Health Care

Another industry that's undergoing rapid changes thanks to machine learning is global health and health care.

In most countries, becoming a doctor requires many years of education. It's a demanding field with long hours, high stakes, and an even higher barrier to entry.

As a result, there has recently been significant effort to alleviate doctors' workload and improve the overall efficiency of the health care system with the help of machine learning.

Uses cases include:

- **Preventative care…** Predicting disease outbreaks on both the individual and the community level.
- **Diagnostic care…** Automatically classifying image data, such as scans, x-rays, etc.
- **Insurance…** Adjusting insurance premiums based on publicly available risk factors.

As hospitals continue to modernize patient records and as we collect more granular health data, there will be an influx of low-hanging fruit opportunities for data scientists to make a difference.

Tutorials

- R: Building meaningful machine learning models for disease prediction
- Machine Learning in Health Care – Excellent presentation by Microsoft Research

Data Sources

- Large Health Data Sets – Collection of large health-related datasets
- data.gov/health – Datasets related to health and health care provided by the U.S. government.
- Health Nutrition and Population Statistics – Global health, nutrition, and population statistics provided by the World Bank.

Machine Learning Glossary

Definitions of common machine learning terms.

Accuracy

Percentage of correct predictions made by the model.

Algorithm

A method, function, or series of instructions used to generate a machine learning model. Examples include linear regression, decision trees, support vector machines, and neural networks.

Attribute

A quality describing an observation (e.g. color, size, weight). In Excel terms, these are column headers.

Bias metric

What is the average difference between your predictions and the correct value for that observation?

- **Low bias** could mean every prediction is correct. It could also mean half of your predictions are above their actual values and half are below, in equal proportion, resulting in low average difference.
- **High bias** (with low variance) suggests your model may be underfitting and you're using the wrong architecture for the job.

Bias term

Allow models to represent patterns that do not pass through the origin. For example, if all my features were 0, would my output also be zero? Is it possible there is some base value upon which my features have an effect? Bias terms typically accompany weights and are attached to neurons or filters.

Categorical Variables

Variables with a discrete set of possible values. Can be ordinal (order matters) or nominal (order doesn't matter).

Classification

Predicting a categorical output.

- **Binary classification** predicts one of two possible outcomes (e.g. is the email spam or not spam?)
- **Multi-class classification** predicts one of multiple possible outcomes (e.g. is this a photo of a cat, dog, horse or human?)

Classification Threshold

The lowest probability value at which we're comfortable asserting a positive classification. For example, if the predicted probability of being diabetic is > 50%, return True, otherwise return False.

Clustering

Unsupervised grouping of data into buckets.

Confusion Matrix

Table that describes the performance of a classification model by grouping predictions into 4 categories.

- **True Positives**: we *correctly* predicted they do have diabetes
- **True Negatives**: we *correctly* predicted they don't have diabetes
- **False Positives**: we *incorrectly* predicted they do have diabetes (Type I error)
- **False Negatives**: we *incorrectly* predicted they don't have diabetes (Type II error)

Continuous Variables

Variables with a range of possible values defined by a number scale (e.g. sales, lifespan).

Convergence

A state reached during the training of a model when the loss changes very little between each iteration.

Deduction

A top-down approach to answering questions or solving problems. A logic technique that starts with a theory and tests that theory with observations to derive a conclusion. E.g. We suspect X, but we need to test our hypothesis before coming to any conclusions.

Deep Learning

Deep Learning is derived from one machine learning algorithm called perceptron or multi layer perceptron that gain more and more attention nowadays because of its success in different fields like, computer vision to signal processing and medical diagnosis to self-driving cars. As all other AI algorithms deep learning is from decades, but now today we have more and more data and cheap computing power that make this algorithm really powerful to achieve state of the art accuracy. In modern world this algorithm knowns as artificial neural network. deep learning is much more than traditional artificial neural network. But it was highly influenced by machine learning's neural network and perceptron network.

Dimension

Dimention for machine learning and data scientist is differ from physics, here Dimention of data means how much feature you have in you data ocean(data-set). e.g in case of object detection application, flatten image size and color channel(e.g 28*28*3) is a feature of the input set. In case of house price prediction (maybe) house size is the data-set so we call it 1 dimentional data.

Epoch

An epoch describes the number of times the algorithm sees the entire data set.

Extrapolation

Making predictions outside the range of a dataset. E.g. My dog barks, so all dogs must bark. In machine learning we often run into trouble when we extrapolate outside the range of our training data.

False Positive Rate

Defined as

FPR=1−Specificity=

FalsePositives

alsePositives+TrueNegatives

FPR=1−Specificity=FalsePositivesFalsePositives+TrueNegatives

The False Positive Rate forms the x-axis of the ROC curve.

Feature

With respect to a dataset, a feature represents an attribute and value combination. Color is an attribute. “Color is blue” is a feature. In Excel terms, features are similar to cells. The term feature has other definitions in different contexts.

Feature Selection

Feature selection is the process of selecting relevant features from a data-set for creating a Machine Learning model.

Feature Vector

A list of features describing an observation with multiple attributes. In Excel we call this a row.

Gradient Accumulation

A mechanism to split the batch of samples—used for training a neural network—into several mini-batches of samples that will be run sequentially. This is used to enable using large batch sizes that require more GPU memory than available.

Hyperparameters

Hyperparameters are higher-level properties of a model such as how fast it can learn (learning rate) or complexity of a model. The depth of trees in a Decision Tree or number of hidden layers in a Neural Networks are examples of hyper parameters.

Induction

A bottoms-up approach to answering questions or solving problems. A logic technique that goes from observations to theory. E.g. We keep observing X, so we infer that Y must be True.

Instance

A data point, row, or sample in a dataset. Another term for observation.

Label

The “answer” portion of an observation in supervised learning. For example, in a dataset used to classify flowers into different species, the features might include the petal length and petal width, while the label would be the flower’s species.

Learning Rate

The size of the update steps to take during optimization loops like Gradient Descent. With a high learning rate we can cover more ground each step, but we risk overshooting the lowest point since the slope of the hill is constantly changing. With a very low learning rate, we can confidently move in the direction of the negative

gradient since we are recalculating it so frequently. A low learning rate is more precise, but calculating the gradient is time-consuming, so it will take us a very long time to get to the bottom.

Loss

Loss = true_value(from data-set)- predicted value(from ML-model) The lower the loss, the better a model (unless the model has over-fitted to the training data). The loss is calculated on training and validation and its interpretation is how well the model is doing for these two sets. Unlike accuracy, loss is not a percentage. It is a summation of the errors made for each example in training or validation sets.

Machine Learning

Mitchell (1997) provides a succinct definition: “A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.” In simple language machine learning is a field in which human made

algorithms have an ability learn by itself or predict future for unseen data.

Model

A data structure that stores a representation of a dataset (weights and biases). Models are created/learned when you train an algorithm on a dataset.

Neural Networks

Neural Networks are mathematical algorithms modeled after the brain's architecture, designed to recognize patterns and relationships in data.

Normalization

Restriction of the values of weights in regression to avoid overfitting and improving computation speed.

Noise

Any irrelevant information or randomness in a dataset which obscures the underlying pattern.

Null Accuracy

Baseline accuracy that can be achieved by always predicting the most frequent class (“B has the highest frequency, so lets guess B every time”).

Observation

A data point, row, or sample in a dataset. Another term for instance.

Outlier

An observation that deviates significantly from other observations in the dataset.

Overfitting

Overfitting occurs when your model learns the training data too well and incorporates details and noise specific to your dataset. You can tell a model is overfitting when it performs great on your

training/validation set, but poorly on your test set (or new real-world data).

Parameters

Parameters are properties of training data learned by training a machine learning model or classifier. They are adjusted using optimization algorithms and unique to each experiment.

Examples of parameters include:

- weights in an artificial neural network
- support vectors in a support vector machine
- coefficients in a linear or logistic regression

Precision

In the context of binary classification (Yes/No), precision measures the model's performance at classifying positive observations (i.e. "Yes"). In other words, when a positive value is predicted, how often is the prediction correct? We could game this metric by only

returning positive for the single observation we are most confident in.

P=

TruePositives

ruePositives+FalsePositives

P=TruePositivesTruePositives+FalsePositives

Recall

Also called sensitivity. In the context of binary classification (Yes/No), recall measures how “sensitive” the classifier is at detecting positive instances. In other words, for all the true observations in our sample, how many did we “catch.” We could game this metric by always classifying observations as positive.

R=

TruePositives

ruePositives+FalseNegatives

R=TruePositivesTruePositives+FalseNegatives

Recall vs Precision

Say we are analyzing Brain scans and trying to predict whether a person has a tumor (True) or not (False). We feed it into our model and our model starts guessing.

- **Precision** is the % of True guesses that were actually correct! If we guess 1 image is True out of 100 images and that image is actually True, then our precision is 100%! Our results aren't helpful however because we missed 10 brain tumors! We were super precise when we tried, but we didn't try hard enough.
- **Recall**, or Sensitivity, provides another lens which with to view how good our model is. Again let's say there are 100 images, 10 with brain tumors, and we correctly guessed 1

had a brain tumor. Precision is 100%, but recall is 10%. Perfect recall requires that we catch all 10 tumors!

Regression

Predicting a continuous output (e.g. price, sales).

Regularization

Regularization is a technique utilized to combat the overfitting problem. This is achieved by adding a complexity term to the loss function that gives a bigger loss for more complex models

Reinforcement Learning

Training a model to maximize a reward via iterative trial and error.

ROC (Receiver Operating Characteristic) Curve

A plot of the true positive rate against the false positive rate at all classification thresholds. This is used to evaluate the performance of a classification model at different classification thresholds. The

area under the ROC curve can be interpreted as the probability that the model correctly distinguishes between a randomly chosen positive observation (e.g. “spam”) and a randomly chosen negative observation (e.g. “not spam”).

Segmentation

Contribute a definition!

Specificity

In the context of binary classification (Yes/No), specificity measures the model’s performance at classifying negative observations (i.e. “No”). In other words, when the correct label is negative, how often is the prediction correct? We could game this metric if we predict everything as negative.

S=

TrueNegatives

rueNegatives+FalsePositives

S=TrueNegativesTrueNegatives+FalsePositives

Supervised Learning

Training a model using a labeled dataset.

Test Set

A set of observations used at the end of model training and validation to assess the predictive power of your model. How generalizable is your model to unseen data?

Training Set

A set of observations used to generate machine learning models.

Transfer Learning

A machine learning method where a model developed for a task is reused as the starting point for a model on a second task. In transfer learning, we take the pre-trained weights of an already

trained model (one that has been trained on millions of images belonging to 1000's of classes, on several high power GPU's for several days) and use these already learned features to predict new classes.

True Positive Rate

Another term for recall, i.e.

TPR=

TruePositives

ruePositives+FalseNegatives

TPR=TruePositivesTruePositives+FalseNegatives

The True Positive Rate forms the y-axis of the ROC curve.

Type 1 Error

False Positives. Consider a company optimizing hiring practices to reduce false positives in job offers. A type 1 error occurs when candidate seems good and they hire him, but he is actually bad.

Type 2 Error

False Negatives. The candidate was great but the company passed on him.

Underfitting

Underfitting occurs when your model over-generalizes and fails to incorporate relevant variations in your data that would give your model more predictive power. You can tell a model is underfitting when it performs poorly on both training and test sets.

Universal Approximation Theorem

A neural network with one hidden layer can approximate any continuous function but only for inputs in a specific range. If you train a network on inputs between -2 and 2, then it will work well for inputs in the same range, but you can’t expect it to generalize to

other inputs without retraining the model or adding more hidden neurons.

Unsupervised Learning

Training a model to find patterns in an unlabeled dataset (e.g. clustering).

Validation Set

A set of observations used during model training to provide feedback on how well the current parameters generalize beyond the training set. If training error decreases but validation error increases, your model is likely overfitting and you should pause training.

Variance

How tightly packed are your predictions for a particular observation relative to each other?

- **Low variance** suggests your model is internally consistent, with predictions varying little from each other after every iteration.
- **High variance** (with low bias) suggests your model may be overfitting and reading too deeply into the noise found in every training set.

References

http://robotics.stanford.edu/~ronnyk/glossary.html

https://developers.google.com/machine-learning/glossary

Machine Learning Research Papers

1. **Dropout: a simple way to prevent neural networks from overfitting**, by Hinton, G.E., Krizhevsky, A., Srivastava, N., Sutskever, I., & Salakhutdinov, R. (2014). Journal of Machine Learning Research, 15, 1929-1958. (cited 2084 times, HIC: 142 , CV: 536).

   Summary: The key idea is to randomly drop units (along with their connections) from the neural network during training. This prevents units from co-adapting too much. This significantly reduces overfitting and gives major improvements over other regularization methods

2. **Deep Residual Learning for Image Recognition**, by He, K., Ren, S., Sun, J., & Zhang, X. (2016). CoRR, abs/1512.03385. (cited 1436 times, HIC: 137 , CV: 582).
   Summary: We present a residual learning framework to ease the training of deep neural networks that are substantially deeper than those used previously. We explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. We provide comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth.

3. **Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift**, by Sergey Ioffe,

Christian Szegedy (2015) ICML. (cited 946 times, HIC: 56 , CV: 0).
Summary: Training Deep Neural Networks is complicated by the fact that the distribution of each layer's inputs changes during training, as the parameters of the previous layers change. We refer to this phenomenon as internal covariate shift, and address the problem by normalizing layer inputs. Applied to a state-of-the-art image classification model, Batch Normalization achieves the same accuracy with 14 times fewer training steps, and beats the original model by a significant margin.

4. **Large-Scale Video Classification with Convolutional Neural Networks** , by Fei-Fei, L., Karpathy, A., Leung, T., Shetty, S., Sukthankar, R., & Toderici, G. (2014). IEEE Conference on Computer Vision and Pattern Recognition (cited 865 times, HIC: 24 , CV: 239)
Summary: Convolutional Neural Networks (CNNs) have been established as a powerful class of models for image recognition problems. Encouraged by these results, we provide an extensive empirical evaluation of CNNs on large-scale video classification using a new dataset of 1 million YouTube videos belonging to 487 classes .
5. **Microsoft COCO: Common Objects in Context** , by Belongie, S.J., Dollár, P., Hays, J., Lin, T., Maire, M., Perona, P., Ramanan, D., & Zitnick, C.L. (2014). ECCV. (cited 830 times, HIC: 78 , CV: 279) Summary: We present a new dataset

with the goal of advancing the state-of-the-art in object recognition by placing the question of object recognition in the context of the broader question of scene understanding. Our dataset contains photos of 91 objects types that would be easily recognizable by a 4 year old. Finally, we provide baseline performance analysis for bounding box and segmentation detection results using a Deformable Parts Model.

6. **Learning deep features for scene recognition using places database** , by Lapedriza, À., Oliva, A., Torralba, A., Xiao, J., & Zhou, B. (2014). NIPS. (cited 644 times, HIC: 65 , CV: 0)
   Summary: We introduce a new scene-centric database called Places with over 7 million labeled pictures of scenes. We propose new methods to compare the density and diversity of image datasets and show that Places is as dense as other scene datasets and has more diversity.

7. **Generative adversarial nets**, by Bengio, Y., Courville, A.C., Goodfellow, I.J., Mirza, M., Ozair, S., Pouget-Abadie, J., Warde-Farley, D., & Xu, B. (2014) NIPS. (cited 463 times, HIC: 55 , CV: 0)
   Summary: We propose a new framework for estimating generative models via an adversarial process, in which we simultaneously train two models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G.

8. **High-Speed Tracking with Kernelized Correlation Filters**, by Batista, J., Caseiro, R., Henriques, J.F., & Martins, P. (2015). CoRR, abs/1404.7584. (cited 439 times, HIC: 43 , CV: 0)
   Summary: In most modern trackers, to cope with natural image changes, a classifier is typically trained with translated and scaled sample patches. We propose an analytic model for datasets of thousands of translated patches. By showing that the resulting data matrix is circulant, we can diagonalize it with the discrete Fourier transform, reducing both storage and computation by several orders of magnitude.

9. **A Review on Multi-Label Learning Algorithms**, by Zhang, M., & Zhou, Z. (2014). IEEE TKDE, (cited 436 times, HIC: 7 , CV: 91)
   Summary: This paper aims to provide a timely review on multi-label learning studies the problem where each example is represented by a single instance while associated with a set of labels simultaneously.

10. **How transferable are features in deep neural networks**, by Bengio, Y., Clune, J., Lipson, H., & Yosinski, J. (2014) CoRR, abs/1411.1792. (cited 402 times, HIC: 14 , CV: 0)
    Summary: We experimentally quantify the generality versus specificity of neurons in each layer of a deep convolutional neural network and report a few surprising results. Transferability is negatively affected by two distinct issues: (1) the specialization of higher layer neurons to their original task

at the expense of performance on the target task, which was expected, and (2) optimization difficulties related to splitting networks between co-adapted neurons, which was not expected.

11. **Do we need hundreds of classifiers to solve real world classification problems**, by Amorim, D.G., Barro, S., Cernadas, E., & Delgado, M.F. (2014). Journal of Machine Learning Research (cited 387 times, HIC: 3 , CV: 0) Summary: We evaluate 179 classifiers arising from 17 families (discriminant analysis, Bayesian, neural networks, support vector machines, decision trees, rule-based classifiers, boosting, bagging, stacking, random forests and other ensembles, generalized linear models, nearest-neighbors, partial least squares and principal component regression, logistic and multinomial regression, multiple adaptive regression splines and other methods). We use 121 data sets from UCI data base to study the classifier behavior, not dependent on the data set collection. The winners are the random forest (RF) versions implemented in R and accessed via caret) and the SVM with Gaussian kernel implemented in C using LibSVM.

12. **Knowledge vault: a web-scale approach to probabilistic knowledge fusion**, by Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., ... & Zhang, W. (2014, August). In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining ACM.

(cited 334 times, HIC: 7 , CV: 107).
Summary: We introduce Knowledge Vault, a Web-scale probabilistic knowledge base that combines extractions from Web content (obtained via analysis of text, tabular data, page structure, and human annotations) with prior knowledge derived from existing knowledge repositories for constructing knowledge bases. We employ supervised machine learning methods for fusing distinct information sources. The Knowledge Vault is substantially bigger than any previously published structured knowledge repository, and features a probabilistic inference system that computes calibrated probabilities of fact correctness.

13. **Scalable Nearest Neighbor Algorithms for High Dimensional Data**, by Lowe, D.G., & Muja, M. (2014). IEEE Trans. Pattern Anal. Mach. Intell., (cited 324 times, HIC: 11 , CV: 69).
Summary: We propose new algorithms for approximate nearest neighbor matching and evaluate and compare them with previous algorithms. In order to scale to very large data sets that would otherwise not fit in the memory of a single machine, we propose a distributed nearest neighbor matching framework that can be used with any of the algorithms described in the paper.

14. **Trends in extreme learning machines: a review**, by Huang, G., Huang, G., Song, S., & You, K. (2015). Neural Networks, (cited 323 times, HIC: 0 , CV: 0)

Summary: We aim to report the current state of the theoretical research and practical advances on Extreme learning machine (ELM). Apart from classification and regression, ELM has recently been extended for clustering, feature selection, representational learning and many other learning tasks. Due to its remarkable efficiency, simplicity, and impressive generalization performance, ELM have been applied in a variety of domains, such as biomedical engineering, computer vision, system identification, and control and robotics.

15. **A survey on concept drift adaptation**, by Bifet, A., Bouchachia, A., Gama, J., Pechenizkiy, M., & Zliobaite, I. ACM Comput. Surv., 2014 , (cited 314 times, HIC: 4 , CV: 23) Summary: This work aims at providing a comprehensive introduction to the concept drift adaptation that refers to an online supervised learning scenario when the relation between the input data and the target variable changes over time.

16. **Multi-scale Orderless Pooling of Deep Convolutional Activation Features**, by Gong, Y., Guo, R., Lazebnik, S., & Wang, L. (2014). ECCV(cited 293 times, HIC: 23 , CV: 95) Summary: To improve the invariance of CNN activations without degrading their discriminative power, this paper presents a simple but effective scheme called multi-scale orderless pooling (MOP-CNN).

17. **Simultaneous Detection and Segmentation**, by Arbeláez, P.A., Girshick, R.B., Hariharan, B., & Malik, J. (2014) ECCV , (cited 286 times, HIC: 23 , CV: 94)

Summary: We aim to detect all instances of a category in an image and, for each instance, mark the pixels that belong to it. We call this task Simultaneous Detection and Segmentation (SDS).

18. **A survey on feature selection methods**, by Chandrashekar, G., & Sahin, F. Int. J. on Computers & Electrical Engineering, (cited 279 times, HIC: 1 , CV: 58)
Summary: Plenty of feature selection methods are available in literature due to the availability of data with hundreds of variables leading to data with very high dimension.

19. **One Millisecond Face Alignment with an Ensemble of Regression Trees**, by Kazemi, Vahid, and Josephine Sullivan, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2014, (cited 277 times, HIC: 15 , CV: 0)
Summary: This paper addresses the problem of Face Alignment for a single image. We show how an ensemble of regression trees can be used to estimate the face's landmark positions directly from a sparse subset of pixel intensities, achieving super-realtime performance with high quality predictions.

20. **A survey of multiple classifier systems as hybrid systems** , by Corchado, E., Graña, M., & Wozniak, M. (2014). Information Fusion, 16, 3-17. (cited 269 times, HIC: 1 , CV: 22)
Summary: A current focus of intense research in pattern classification is the combination of several classifier systems,

which can be built following either the same or different models and/or datasets building.

Machine Learning To-Do

Step 0: Prerequisites

Machine learning can appear intimidating without a gentle introduction to its prerequisites. You don't need to be a professional mathematician or veteran programmer to learn machine learning, but you do need to have the core skills in those domains.

The good news is that once you fulfill the prerequisites, the rest will be fairly easy. In fact, almost all of ML is about applying concepts from statistics and computer science to data.

Task: Make sure you are caught up to speed for at least programming and statistics.

- 

  Python for Data Science

  You can’t use machine learning unless you know how to program. Luckily, we have a free guide: How to Learn Python for Data Science, The Self-Starter Way

- 

  Statistics for Data Science

  Understanding statistics, especially Bayesian probability, is essential for many machine learning algorithms. We have a free guide for you: How to Learn Statistics for Data Science, The Self-Starter Way

- 

  Math for Data Science

  Original algorithm research requires a foundation in linear algebra and multivariable calculus. We have a free guide: How to Learn Math for Data Science, The Self-Starter Way

Back to Table of Contents

Step 1: Sponge Mode

Sponge mode is all about soaking in as much theory and knowledge as possible to give yourself a strong foundation.

Pictured: Spongebob (NOT Sponge Mode)

Now, some people may be wondering: *"If I don't plan to perform original research, why would I need to learn the theory when I can just use existing ML packages?"*

This is a reasonable question!

However, **learning the fundamentals is important for anyone who plans to apply machine learning in their work.** Here are 5 super practical reasons for learning ML theory. They span the entire modeling process:

1. **Planning and data collection.** Data collection can be an expensive and time consuming process. *What types of*

data do I need to collect? How much data do I need (hint: it's different depending on the model)? Is this challenge feasible?

2. **Data assumptions and preprocessing.** Different algorithms have different assumptions about the input data. *How should I preprocess my data? Should I normalize it? Is my model robust to missing data? How about outliers?*
3. **Interpreting model results.** The notion that ML is a "black box" is simply false. Yes, not all results are directly interpretable, but you need to be able to diagnose your models to improve them. *How can I tell if my model is overfit or underfit? How do I explain these results to business stakeholders? How much room for improvement is left?*
4. **Improving and tuning your models.** You'll rarely reach the best model on your first try. You need to understand the nuances of different tuning parameters and regularization methods. *If my model is overfit, how can I remedy it? Should I spend more time on feature-engineering or on data collection? Can I ensemble my models?*

5. **Driving to business value.** ML is never done in a vacuum. If you don't truly understand the tools in your arsenal, you can't maximize their effectiveness. *Which outcome metrics are most important to optimize? Are there other algorithms that work better here? When is ML not the answer?*

Here's the great news... you *don't* need to have all the answers to these questions right from the start. In fact, the approach we recommend is to learn just enough theory to get started and not go astray. Then, you can build mastery over time by alternating between theory and practice.

1.1 Best Free Machine Learning Courses

These next two free courses are world-class (from Harvard and Stanford) resources for Sponge Mode.

Task: Complete at least one of the courses below.

- 

Harvard's Data Science Course

End-to-end data science course. While there's less emphasis on ML than in Andrew Ng's course, you'll get more practice with the entire data science workflow from data collection to analysis. (Course Homepage | Lecture Videos and Slides | Homework Assignments)

- 

Stanford's Machine Learning Course

This is the famous course taught by Andrew Ng, and it's the gold standard when it comes to learning machine learning theory. These videos really clear up the core concepts behind ML. *If you only have time for 1 course, we recommend this one.* (Course Videos)

1.2 Keys to Success

Here are a few keys to success for this step:

A.) Pay attention to the big picture and always ask "why."

Every time you're introduced to a new concept, ask "why." Why use a decision tree instead of regression in some cases? Why regularize parameters? Why split your dataset? When you understand *why* each tool is used, you'll become a true machine learning *practitioner*. For example, by the end of this step, you should know when to preprocess your data, when to use supervised vs. unsupervised algorithms, and methods for preventing model overfitting.

B.) Accept that you will not remember everything.

Don't stress about taking insane notes or reviewing everything 3 times. Accept that you'll need to cycle back and review concepts as you encounter them in the wild.

C.) Keep moving and don't be discouraged.

Try to avoid dwelling on any topic for too long. Some concepts can't be explained easily, even by the best professors. Your confusion will clear up once you start applying them in practice.

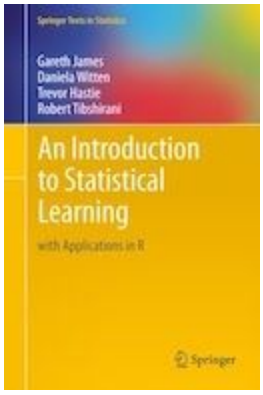
D.) Videos are more effective than textbooks.

From our experience, textbooks can be great reference tools, but they often omit the vital color commentary surrounding key concepts. We strongly recommend video lectures during Sponge Mode.

1.3 Free Reference Textbooks

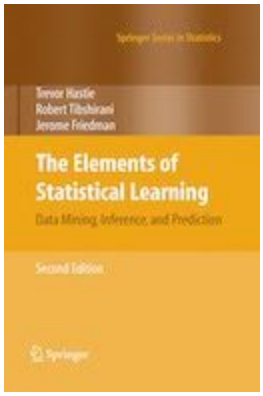
Next, we have free (legal) PDFs of 2 classic textbooks in the industry.

Task: Download the free PDFs for your future reference.

- 

  An Introduction to Statistical Learning

  Gentler introduction than Elements of Statistical Learning. Recommended for everyone. (PDF)

- 

  Elements of Statistical Learning

  Rigorous treatment of ML theory and mathematics. Recommended for ML researchers. (PDF)

Back to Table of Contents

Step 2: Targeted Practice

After Sponge Mode, you've probably already gotten a healthy dose of practice.

Now it's time to take that practice to the next level.

Step 2: Targeted Practice is all about using specific, deliberate exercises to hone your skills. The goal of this step is threefold:

1. **Practice the entire machine learning workflow:** Data collection, cleaning, and preprocessing. Model building, tuning, and evaluation.
2. **Practice on real datasets:** You'll start to build intuition around which types of models are appropriate for which types challenges.
3. **Deep dive on individual topics:** For example, in Step 1, you learned about clustering algorithms. In Step 2, you'll apply different types of clustering algorithms on datasets to see which perform the best.

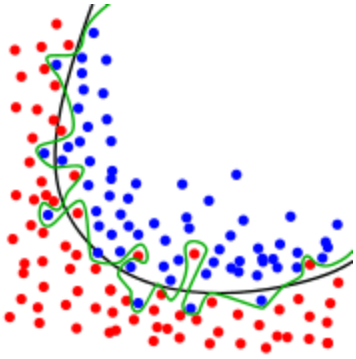
After this step, you'll be ready to tackle bigger projects without feeling overwhelmed.

2.1 - The 9 Essential Topics

Machine learning is a broad and rich field. There are applications for almost any industry. It's easy to get flustered by all there is to learn. Plus, it's also easy to get lost in the weeds of individual models and lose sight of the big picture.

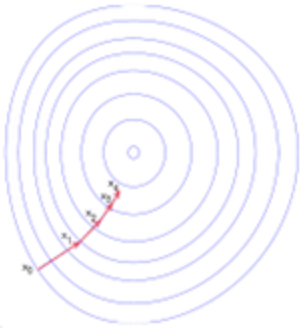
Therefore, we've broken the essentials into the following 9 topics.

These are building block topics that collectively represent the simple value proposition of machine learning: **taking data and transforming it into something useful.**

- 

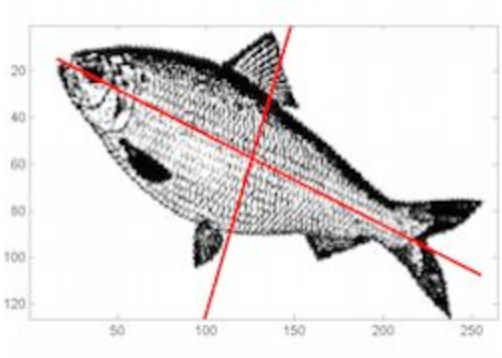
  The Big Picture

  Essential ML theory, such as the Bias-Variance tradeoff.



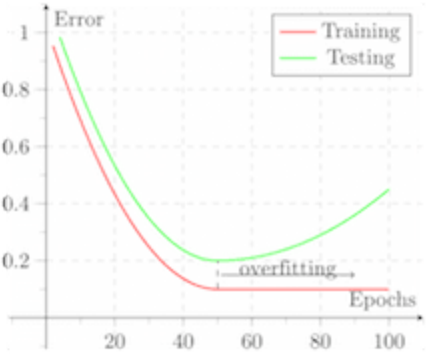
- **Optimization**

  Algorithms for finding the best parameters for a model.



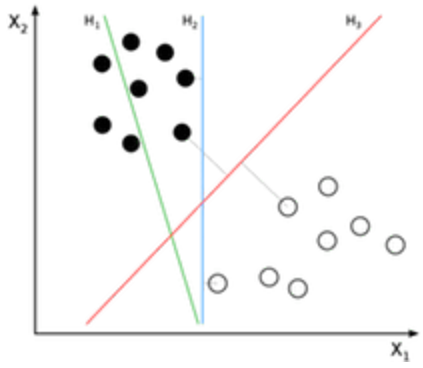
- **Data Preprocessing**

  Dealing with missing data, skewed distributions, outliers, etc.



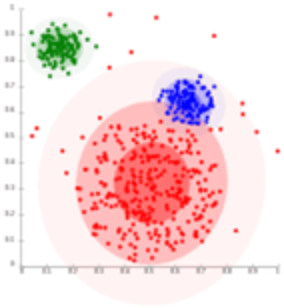
- **Sampling & Splitting**

How to split your datasets to tune parameters and avoid overfitting.

- 

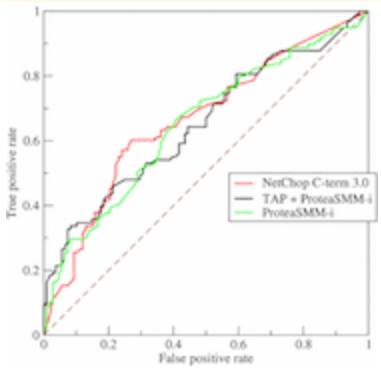
Supervised Learning

Learning from labeled data using classification and regression models.

- 

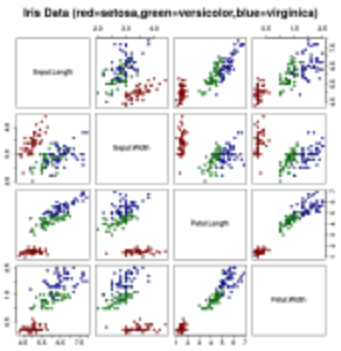
Unsupervised Learning

Learning from unlabeled data using factor and cluster analysis models.



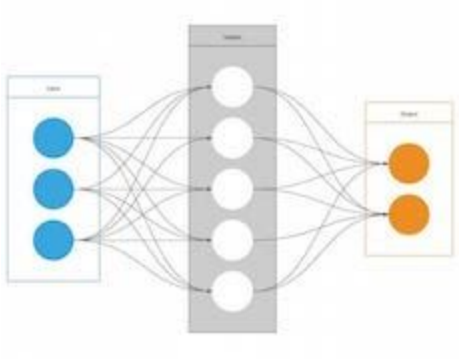
- **Model Evaluation**

  Making decisions based on various performance metrics.



- **Ensemble Learning**

  Combining multiple models for better performance.



- **Business Applications**

  How machine learning can help different types of businesses.

2.2 - Tools of the Trade

For this step, we strongly recommend that you start with out-of-the-box algorithm implementations for two reasons.

First, this is how most ML is performed in the industry. Sure, there will be times when you'll need to research original algorithms or develop them from scratch, but prototyping always starts with existing libraries.

Second, you'll get the chance to practice the entire ML workflow without spending too much time on any one portion of it. This will give you an invaluable "big picture intuition."

Depending on your programming language of choice, you have 2 excellent options.

Task: Complete the Quickstart guide for one of the libraries below.

Python: Scikit-Learn

Scikit-learn, or sklearn, is the gold standard Python library for general purpose machine learning. It does almost everything, and it has implementations of all the common algorithms.

Scikit-Learn Tutorial, Wine Snob Edition

R: Caret

Caret is love. Caret is life. Caret is a library that provides a unified interface for many different model packages in R. It also includes

functions for preprocessing, data splitting, and model evaluation, making it a complete end-to-end solution.

Quickstart Webinar

2.3 - Datasets for Practice

For this step, you'll need datasets to practice building and tuning models.

Again, the point of Step 2: Targeted Practice is to **take the theory that's floating around in your mind after Step 1: Sponge Mode and put it into code.**

Much of the art in data science and machine learning lies in dozens of micro-decisions you'll make to solve each problem. This is the perfect time to practice making those micro-decisions and evaluating the consequences of each.

Task: Pick 5-10 datasets from the options below. We recommend starting with the UCI Machine Learning Repository. For example,

you can pick 3 datasets each for regression, classification, and clustering.

Task: For each dataset, try at least 3 different modeling approaches using Scikit-Learn or Caret. Think about the following questions:

- What types of preprocessing do you need to perform for each dataset?
- Do you need to reduce dimensions or perform feature selection? If so, what methods can you use?
- How should you sample or split your dataset?
- How do you know if your model is overfit?
- What types of performance metrics should you use?
- How do different tuning parameters affect your model results?
- Can you ensemble to get better results?
- (*For clustering*) Do your clusters appear intuitive?

We also have a curated list of some of our favorite datasets for practice and projects.

- 

  UCI Machine Learning Repo

  This is an incredible collection of over 350 different datasets specifically curated for practicing machine learning. You can search by task (i.e. regression, classification, or clustering), industry, dataset size, and more. (Go to website)

- 

  Kaggle

  Kaggle.com is most famous for hosting data science competitions, but the site also houses over 180 community datasets for fun topics ranging from Pokemon data to European Soccer matches. (Go to website)

- 

  Data.gov

  If you're looking for social science or government-related

datasets, look no further than Data.gov, a collection of the U.S. government's open data. You can search over 190,000 datasets. (Go to website)

Back to Table of Contents

Step 3: Machine Learning Projects

Alright, now comes the *really* fun part! Up to now, we've covered prerequisites, essential theory, and targeted practice. We're now ready to dive into some bigger projects.

The goal of this step is to **practice integrating machine learning techniques into complete, end-to-end analyses.**

Task: Complete the projects below. The order is up to you, but we ordered them by difficulty (easiest first).

3.1 - Titanic Survivor Prediction

The Titanic Survivor Prediction challenge is an incredibly popular project for practicing machine learning. In fact, it's the most popular competition on Kaggle.com.

We love this project as a starting point because there's a wealth of great tutorials out there. You can take a peek into the minds of more experienced data scientists and see how they approach data exploration, feature engineering, and model tuning.

The Titanic is sinking!

Python Tutorials

- Four-Part Tutorial by Kaggle - Detailed tutorial that starts from cleaning and exploring the data. We really like this tutorial because it teaches you how to properly preprocess and wrangle your data properly before using sklearn.

- Tutorial and iPython Notebooks by Pycon UK - Great tutorial that's presented in iPython Notebook. It has excellent appendices on cross-validation and visualization.

R Tutorials

- Binary Outcome Modeling Tutorial - Walks through a couple different models in R using the caret package. This tutorial nicely summarizes the predictive modeling process from end-to-end.
- An "Irresponsibly" Fast Tutorial - Bare bones tutorial that completely skips the theory. Useful as another perspective (and it shows random forests in action).

3.2 - Algorithm from Scratch

There's nothing that pushes your understanding quite like writing an algorithm from scratch. They say the devil's in the details, and here's where that really rings true.

We recommend starting with something simple, like logistic regression, decision trees, or k-nearest neighbors.

This project will also give you invaluable practice in translating math into code. This skill will be very handy when you eventually need to use the latest research from academia in your work.

If you get stuck, here are some tips:

- Wikipedia is a great resource for this project because it has pseudo-code for many common algorithms.
- For inspiration, try looking at the source code from existing ML packages.
- Break your algorithm into pieces. Write separate functions for sampling, gradient descent, etc.
- Start simple. Implement a decision tree before trying to write a random forest.

She's only a few years away from learning machine learning...

3.3 - Pick a Fun Project or Interesting Domain

You wouldn't be a self-starter if you didn't have curiosity and ideas. By now, you're probably itching to get started (or have already started) on some grand idea that you've been mulling over.

This is honestly the best part about learning machine learning. It's such a powerful tool that once you start to understand, so many ideas will come to you.

The good news is that if you've been following along, then you're more than ready to jump in. Go forth, and reap the fruits of your labor!

We'll also keep a list of project ideas here for inspiration:

Project Ideas

- 8 Fun Machine Learning Projects for Beginners

Back to Table of Contents

Great Job! (So Far...)

Congratulations on reaching the end of the self-study guide!

Here's some great news: If you've followed along and completed all the tasks, you're better at applied machine learning than 90% of the people out there claiming to be data scientists. You have an awesome skillset that employers will drool over.

Now, here's some better news: There's still much to learn! For example, deep learning, computer vision, and natural language processing are a few of the fascinating, cutting-edge subfields that await you.

The key to becoming the best data scientist or machine learning engineer you can be is to **never stop learning.** Welcome to the start of your journey in this dynamic, exciting field!

So great job! *So far..*

Bonus Goodies:

Top 10 Tips for Beginners

If you've chosen to seriously study machine learning, then congratulations! You have a fun and rewarding journey ahead of you.

Here are 10 tips that every beginner should know:

1. Set concrete goals or deadlines.

Machine learning is a rich field that's expanding every year. It can be easy to go down rabbit holes. Set concrete goals for yourself and keep moving.

2. Walk before you run.

You might be tempted to jump into some of the newest, cutting edge sub-fields in machine learning such as deep learning or NLP. Try to stay focused on the core concepts at the start. These advanced topics will be much easier to understand once you've mastered the core skills.

3. Alternate between practice and theory.

Practice and theory go hand-in-hand. You won't be able to master theory without applying it, yet you won't know what to do without the theory.

4. Write a few algorithms from scratch.

Once you've had some practice applying algorithms from existing packages, you'll want to write a few from scratch. This will take your understanding to the next level and allow you to customize them in the future.

5. Seek different perspectives.

The way a statistician explains an algorithm will be different from the way a computer scientist explains it. Seek different explanations of the same topic.

6. Tie each algorithm to value.

For each tool or algorithm you learn, try to think of ways it could be applied in business or technology. This is essential for learning how to "think" like a data scientist.

7. Don't believe the hype.

Machine learning is *not* what the movies portray as artificial intelligence. It's a powerful tool, but you should approach problems with rationality and an open mind. ML should just be one tool in your arsenal!

8. Ignore the show-offs.

Sometimes you'll see people online debating with lots of math and jargon. If you don't understand it, don't be discouraged. What

matters is: Can you use ML to add value in some way? And the answer is yes, you absolutely can.

9. Think "inputs/outputs" and ask "why."

At times, you might find yourself lost in the weeds. When in doubt, take a step back and think about how data inputs and outputs piece together. Ask "why" at each part of the process.

10. Find fun projects that interest you!

Rome wasn't built in a day, and neither will your machine learning skills be. Pick topics that interest you, take your time, and have fun along the way.