A Crash Course in Machine Learning

Ava Lee

January 17, 2021

Contents

Chapter 1

Introduction

What is Machine Learning (ML)? According to the firm Emerj, this is the best definition of ML [1]:

> Machine Learning is the science of getting computers to learn and act like humans do, and improve their learning over time in autonomous fashion, by feeding them data and information in the form of observations and real-world interactions.

While ML in the definition sounds like artificial intelligence (AI), it is important to note that ML is a subset of AI and is a technique used to realise AI.

The aim of the course is to give a short overview of ML algorithms and discuss the practical considerations of constructing a ML solution. The course structure is based loosely on the *Introduction to Machine Learning with Python* book [2] by Andreas Müller. Additional resources, including implementations in Python if applicable, are referenced. If two references are given, the second is the implementation in Python.

Nomenclature

Data is often considered as a table with each data point as a row, i.e. *sample*, and each property that describes the data point as a column i.e. *features*. Dimensions refer to the number of features in the data.

Notation

Input space is denoted by X and the output space by Y. For each sample i, $\boldsymbol{x}_i$ is a vector of features and y_i is its actual class or output value (if supervised learning). The predicted output space is $\hat{Y}$ with predicted outputs as $\hat{y}$.

1.1 Types of Learning

ML techniques are broadly separated into 3 categories:

- *Supervised learning:* Training data fed to the algorithm includes both the inputs and the desired outputs, where the supervision stems from.
- *Unsupervised learning:* Draws inferences from its inputs since only the input data is known.
- *Reinforcement learning:* Algorithm learns by itself the best strategy to perform certain goals through rewards and punishments. (Not covered in this course.)

1.2 Optimisation

Learning is often formulated as an optimisation problem, in which the typical objective is to minimise the **loss function [3, 4]**, which measures the performance of the fit of the algorithm to the training data. It can be used during training, such as in the case of decision trees, or after training to evaluate the performance of the algorithm, which is discussed further in Section 3.3.

1.2.1 Examples

Gradient descent [5] is the most popular optimisation algorithm for finding the minimum of a function by taking iterative steps in the opposite direction of the gradient, i.e. direction of steepest descent. (Conversely, gradient ascent is used to

find the maximum of a function, where the steps are taken in the direction of the gradient.)

Variants of gradient descent include

- **Stochastic gradient descent (SGD) [6, 7]:** Uses random data-points at each iteration compared to gradient descent, which uses all data at each iteration.
- **Adaptive gradient estimation (AdaGrad) [8, 9]:** Builds on SGD and uses an adaptive learning rate that is set according to the cumulative sum of the squared gradients.
- **Root mean square prop (RMSprop) [8, 9]:** Similar to AdaGrad but uses the exponential moving average (MA) of the squared gradients.
- **Adaptive momentum estimation (Adam) [8, 9]:** Incorporates momentum, which uses the MA of the gradient instead of the gradient itself, into RMSprop.

1.3 Supervised learning

The problems in supervised learning are categorised depending on the output:

- *Classification*: Output is discrete/categorical. Classification can be binary or multiclass, i.e. predicting one class out of 2 classes or more than 2 classes, respectively.
- *Regression:* Output is continuous.

Supervised learning is generally well-understood and its performance is easy to measure. Hence, if your application can be formulated as a supervised learning problem, it is likely that ML can solve your problem!

1.3.1 Examples

- ***k*-nearest neighbours (*k*NN) [10]:** Calculate distance between a sample x to all training samples. Find k nearest neighbours to x. Classify x by majority

vote of the labels of its k nearest neighbours.

  - Good for small and low dimensional datasets with continuous values.
  - kNN is a type of instance-based learning where the training does not explicitly occur as the algorithm only memorises the training data.

- **Linear regression [11]:** Minimises mean squared error between y and $\hat{y}$.
  - Good for large and high dimensional datasets.
- **Logistic regression [12]:** Uses a logistic function to calculate the probability that x_i belongs to class y in Y.
  - Only for classification (despite the name). Good for large and high dimensional datasets.
- **Naives' Bayes [13]:** Developed from Bayes' Theorem. The conditional probability of each class y in Y given x_i is calculated. Class with the highest probability is the predicted class of x_i.
  - Only for classification. Good for very large datasets and high dimensional data. Fast, but often less accurate.
- **Decision trees [14, 15]**: At each node, a feature in X is compared to its corresponding threshold value to recursively split the data into two until a stopping criterion is fulfilled. The terminal node, i.e. leaf, that x_i arrives at determines $\hat{y}_i$.
  - Scaling of data not required. Fast, but not robust.
- **Random forests [16]**: A collection of decision trees, where each tree is grown using a random subset of the data. Majority vote on the predicted outputs from each tree to obtain the final predicted classes or values.
  - Not very good for very high dimensional sparse data. Very robust and powerful. Scaling of data not required.
- **Supported vector machine (SVM) [17]:** Support vectors are samples that

are close to the hyperplane to maximise the distance between samples of different classes.

  - Good for medium-sized datasets of features with similar meaning.

- **Artificial neural networks i.e. Neural networks (NNs) [18]:** Inspired by biological neural networks. There are many different types of NNs. A simple model is **multilayer perceptrons (MLPs) [18, 19]**, also known as feed-forward NNs, or just NNs. MLPs are formed of one or more layers of neurons that compute the output signal from the weighted input signals using an activation function.

  - Good for building very complex models, particularly for large datasets. Large models need a long time to train.

1.4 Unsupervised learning

In contrast, unsupervised learning is harder to understand and evaluate since the data does not contain any labelled information. These learning algorithms are often used in an exploratory setting to understand the data better. Unsupervised algorithms can also be used as a preprocessing step for supervised algorithms by learning a new representation of the data.

1.4.1 Examples

(NNs can also be an unsupervised algorithm if the data does not contain outputs.)

Clustering

- ***k*-means clustering [20]:** Divides the data into k clusters by assigning each sample to the cluster with the nearest mean i.e. cluster centres.

  - Good for very large and low dimensional datasets.

- **Agglomerative clustering [21]:** Starts with each point as its own cluster. Merges two most similar clusters until a stopping criterion is reached.
  - Cannot make predictions for new samples. No prior knowledge to the number of clusters required.

Dimensionality reduction

The following are also feature extraction techniques (Section 2.3).

- **Principal component analysis (PCA) [22]:** Uses principal components (PCs), which capture the maximal variance with the data, to perform an orthogonal transformation on the data from correlate features to uncorrelated PCs.
  - For visualisation purposes and removing correlated features.
- **Non-negative matrix factorisation (NMF) [23, 24]:** Factorises an input matrix V of shape $m \times n$ into two matrices W and H, such that the dimension of W is $m \times p$ and that of H is $p \times n$, where p can be significantly less than both m and n. Assumes that all entries of W and H are positive given that all entries of V are positive.
  - For features with weak predictability.
- **t-Distributed Stochastic Neighbour Embedding (t-SNE) [25]:** Calculates a similarity measure between two samples in high dimensional space and in low dimensional space. Minimises the two similarity measures.
  - Good for exploratory data analysis, but can only transform the data they were trained for, hence for visualisation purposes.

Chapter 2

Data engineering

For all ML algorithms, it is important that your input data is represented in a way that the computer can understand. The techniques used to build such a representation are discussed in this chapter.

Prior to applying the techniques to the data in this chapter, the data needs to be cleaned. The standard data cleaning procedure [26] involves

- Removing duplicate data
- Removing irrelevant data
- Handling missing data
- Checking outliers for mistakes
- Standardisation and normalisation (not needed in some models) [27]
  - Standardisation: Transform data to have a mean of 0 and a standard deviation of 1.
  - Normalisation: Scale data to the range [0, 1].

2.1 Feature transformation

Features are split into categorical/discrete and continuous features, which is the input analogy to classification and regression.

(The transformations in this section are usually monotonic, unlike the some techniques in feature selection and extraction, which can also be considered as transformations.)

2.1.1 Continuous features

Transformations of continuous features [28] are crucial as many algorithms are strongly influenced by the scale and distribution of each feature, therefore transformation techniques, such as standardisation and normalisation, are commonly applied. (NOTE: Feature transformations are irrelevant for tree-based models, since these models split data by comparing features to threshold values.)

2.1.1.1 Example transformation techniques

(In addition to standardisation and normalisation.)

- Applying mathematical functions e.g. *log, exp, sin, cos*
- Scaling data to range [-1, 1]
- Removing median from data and scaling by the InterQuartile Range
- Mapping values to a normal distribution
- Applying generalised power functions

Furthermore, you can use your own functions to transform data.

2.1.2 Categorical features

Most ML algorithms cannot train with categorical features [29], although tree-based models can depending on the implementation.

The most common technique for transforming categorical features into continuous features is **one-hot encoding**, which introduces a binary feature/variable for each possible value of the categorical feature.

If the categorical features have an ordered relationship in the values, then **integer encoding** can be applied, in which each value in the categorical feature is assigned an integer value to represent their order in the feature.

2.2 Feature selection

Feature selection [30] reduces the number of features by filtering out irrelevant or unnecessary features. It is very similar to feature extraction except that feature selection retains a subset of the original features while feature extraction creates new features.

2.2.1 Examples

Supervised

- **Genetic algorithms (GAs) [30, 31]:** Inspired by natural selection. With "genes" representing individual features and "organism" representing a candidate set of features. Each organism is graded on a fitness score. Fittest organisms survive and reproduce until the algorithm converges to a solution.
  - Can efficiently select features from very high dimensional datasets. But computationally expensive, and hence slow.
- **Model-based selection [2]:** Uses a supervised model to judge the importance of each feature. Model does not need to be the same model used in training.
  - Usually good performance, but depends on the model used. Can be computationally expensive.
- **Stability selection [32]:** Built on model-based selection. Feature selection algorithm is repeatedly applied on different subsets of data, then selection results are aggregated.

  - Can efficiently select features from very high dimensional datasets. But very computationally expensive.

- **Recursive feature elimination (RFE) [32]:** Also built on model-based selection. Iteratively remove the worst performing feature after applying feature selection algorithm on data. Features ranked according to when they were eliminated.
  - Can efficiently select features from very high dimensional datasets. But very computationally expensive.

Unsupervised

- **Variance thresholds [30, 33]:** Remove features whose variance falls below a threshold, as features that do not vary much do not add much information.
  - Easy and fast. But variance threshold needs to be tuned and often not sufficient for reducing dimensions in data.
- **Correlation thresholds [30, 33]:** Remove features that are highly correlated to others according to a threshold.
  - For training algorithms that are not robust to correlated features. Easy and fast. But correlation threshold needs to be tuned.
- **Univariate statistics [34]:** Compute statistical relationship between each feature and the output. Features that are highly correlated to the output are selected.
  - For supervised learning algorithms in training (the procedure itself is unsupervised). Fast.

2.3 Feature extraction

Feature extraction is used to extract informative features, which is similar to feature selection, but it creates new features.

2.3.1 Examples

Supervised

- **MLPs:** (Section 1.3.1.)
- **Linear discriminant analysis (LDA) [30, 35]:** Similar to PCA, but maximises the separability between classes.
  - Can also be used as a classifier, commonly in topic modelling [36].
  - Can improve the predictive performance of extracted features. But new features may not be easily interpretable and need to tune the number of components to keep.

Unsupervised

- ***k*-means clustering:** (Section 1.4.1.)
- **PCA:** (Section 1.4.1.)
- **NMF:** (Section 1.4.1.)
- **Independent component analysis (ICA) [37]:** Decompose a multivariate signal into independent sub-components
  - Primarily used for signal processing.
- **Autoencoders [38]:** Consists of a pair of deep learning networks, encoder and decoder. Encoder compresses the input representation and the decoder attempts to recreate the original input by converting the encoder output. For feature extraction, the encoder output is saved.
  - (Feature extraction implementations for classification [39] and regression [40].)
  - Can efficiently select features from very high dimensional datasets. Computationally expensive.

Chapter 3

Model evaluation

ML models are evaluated to determine their performance and generalisation.

To evaluate the data and to avoid overfitting of the model, the data is split into three datasets [41]:

- *Training dataset:* Data samples used to fit the model.
- *Validation dataset:* Data samples to evaluate the fit while tuning the free parameters, i.e. *hyperparameters*, of the model. The evaluation is biased as the learning of the validation dataset is included into the final model configuration.
- *Test dataset:* Data samples to provide an unbiased evaluation of the fit of the final model on the training dataset.

3.1 Cross-validation

As the splitting of the data reduces the size of the data available in training, this may decrease the accuracy of the model. A better way of splitting the data is to use cross-validation, where the data is repeatedly split into a training dataset and a test dataset, and the validation set is no longer needed [42].

3.1.1 Examples

***k*-fold cross validation** [43] shuffles the data and splits the dataset into k number of folds, where one fold is used as a testing dataset in each iteration of k iterations. Variations [42] of k-fold cross validation include

- **Repeated *k*-fold:** Repeat k-fold n times.
- **Leave one out:** $k = n$, which is the number of samples in the data.
- **Shuffle split:** In each iteration, the data is shuffled again, then the data is split into k folds to redefine the training and test datasets.
- **Stratified *k*-fold:** Folds contain approximately the same percentage of each y in Y.
- **Stratified shuffle split:** Similar to shuffle split, but with the folds containing the same percentage of each y in Y.
- **Group *k*-fold:** For data with groups of dependent samples. Ensures same group is not represented in both training and test datasets.

3.2 Hyperparameter optimisation

Cross validation is incorporated into the tuning of hyperparameters by evaluating the performance of different hyperparameter combinations to select the final model.

Different optimisation techniques can be used, but the two most common and simplest methods [44, 45] are

- **Grid search:** Each combination from a grid of hyperparameter values is used to train and score the model. Every combination of hyperparameter values is evaluated, but can be very inefficient.
- **Random search:** Selects random combinations from a grid of hyperparameter values to train and score the model. A lot faster, but the result may not be as accurate as that of Grid Search.

3.3 Performance metrics

Accuracy is a metric used to evaluate the performance of a classification model, but it may not be sufficient to judge the performance especially in cases where the data is unbalanced. Other metrics [46] are listed in this section (for both classification and regression).

(NOTE: The metrics below are for supervised learning since evaluating unsupervised models is often a qualitative process. However, if ground truth references are available, the accuracy of unsupervised models could be evaluated.)

Classification

- **Confusion matrix:** Represents model predictions compared to actual outputs.
- **Precision:** Calculates the proportion of positive identifications that was correct.
- **Recall:** Calculates the proportion of actual positives that was identified correctly.
- **Precision-recall curve:** Shows trade-off between precision and recall.
- **F1 score:** Combines precision and recall.
- **Receiving operating characteristic (ROC) curve:** Shows true positive rate (i.e. recall) against false positive rate (i.e. proportion of actual positives that was missed by classifier). **Area under the curve (AUC)** is used as a measure of performance; bigger area indicates better performance.
- **Log loss [47]:** Represents the uncertainty in the prediction based on its variation from the actual output.

Regression

- **Mean absolute error (MAE):** Average absolute distance between predicted and actual values.

- **Mean squared error (MSE):** Average squared distance between predicted and actual values.
- **Root mean squared error (RMSE):** Average deviation in predictions from actual values. Assumes error follows a normal distribution.
- **Root mean squared logarithmic error (RMSLE):** Robust to outliers and penalises the underestimations more than overestimations.
- **Root mean squared logarithmic error (RMSLE):** Robust to outliers and penalises the underestimations more than overestimations.
- **R-squared:** $R^2 = 1 - MSE_{model}/MSE_{base}$. MSE_{model} is the MSE of predictions against real values. MSE_{base} is the MSE of mean predictions against real values.

Chapter 4

Summary

The key steps in creating a ML solution can be summarised in a workflow:

1. Data collection
2. Data engineering
   (a) Data cleaning
   (b) Feature transformation
   (c) Feature extraction and selection
3. Model evaluation
   (a) Cross-validation
   (b) Hyperparameter optimisation
   (c) Performance evaluation

The workflow is essentially the outline of this course. In `scikit-learn`, the `Pipeline` [48] class can be used to implement a workflow.

When building a ML model, the following questions from the coursebook may be helpful to consider:

- What question(s) am I trying to answer? Do I think the data collected can answer the question?
- What is the best way to phrase my question(s) as a ML problem?

- Have I collected enough data to represent the problem I want to solve?
- What features of the data did I extract, and will these enable the right predictions?
- How will I measure success in my application?
- How will the machine learning solution interact with other parts of my research or business product?

Hopefully, you have gained an understanding of the components in building a ML solution to your problem! Now you can apply what you have learned during this course in a short ML project on housing prices in California [49, 50].

Additional resources

- *The Elements of Statistical Learning* [51] for the mathematical theory behind the ML algorithms.
- *Data Science from Scratch* [52] to implement algorithms from sratch in Python.
- `https://www.kaggle.com/` has many datasets, notebooks, and courses available for free.
- *Machine Learning Crash Course* from Google [53] for lecture series on different ML concepts and includes practical applications in Python.

References

1. Faggella, D. *What is Machine Learning? - An Informed Definition* 2020. https://emerj.com/ai-glossary-terms/what-is-machine-learning/ (2021).
2. Müller, A. C. & Guido, S. *Introduction to Machine Learning with Python: A Guide for Data Scientists* ("O'Reilly Media, Inc.", 2016).
3. Brownlee, J. *Loss and Loss Functions for Training Deep Learning Neural Networks* 2019. https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/ (2021).
4. Mahendru, K. *A Detailed Guide to 7 Loss Functions for Machine Learning Algorithms with Python Code* 2019. https://www.analyticsvidhya.com/blog/2019/08/detailed-guide-7-loss-functions-machine-learning-python-code/ (2021).
5. Silva, T. *Machine Learning 101: An Intuitive Introduction to Gradient Descent* 2019. https://towardsdatascience.com/machine-learning-101-an-intuitive-introduction-to-gradient-descent-366b77b52645 (2021).
6. Srinivasan, A. V. *Stochastic Gradient Descent — Clearly Explained!!* 2019. https://towardsdatascience.com/stochastic-gradient-descent-clearly-explained-53d239905d31 (2021).
7. Mainkar, S. *Gradient Descent in Python* 2018. https://towardsdatascience.com/gradient-descent-in-python-a0d07285742f (2021).

8. Raimi Bin Karim. *10 Gradient Descent Optimisation Algorithms + Cheat Sheet* https://www.kdnuggets.com/10-gradient-descent-optimisation-algorithms-cheat-sheet.html/ (2021).

9. Chandra, A. L. *Learning Parameters Part 5: AdaGrad, RMSProp, and Adam* 2019. https://towardsdatascience.com/learning-parameters-part-5-65a2f3583f7d (2021).

10. Robinson, S. *K-Nearest Neighbors Algorithm in Python and Scikit-Learn* https://stackabuse.com/k-nearest-neighbors-algorithm-in-python-and-scikit-learn/ (2021).

11. Machine Learning Glossary. *Linear Regression* https://ml-cheatsheet.readthedocs.io/en/latest/linear_regression.html#multivariable-regression (2021).

12. Machine Learning Glossary. *Logistic Regression* https://ml-cheatsheet.readthedocs.io/en/latest/logistic_regression.html (2021).

13. Kumar, N. *Naive Bayes Classifiers* 2017. https://www.geeksforgeeks.org/naive-bayes-classifiers/ (2021).

14. Gupta, P. *Decision Trees in Machine Learning* 2017. https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052 (2021).

15. Sharma, A. *Decision tree implementation using Python* 2017. https://www.geeksforgeeks.org/decision-tree-implementation-python/ (2021).

16. Navlani, A. *Random Forests Classifiers in Python* 2018. https://www.datacamp.com/community/tutorials/random-forests-classifier-python (2021).

17. Gandhi, R. *Support Vector Machine — Introduction to Machine Learning Algorithms* 2018. https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47 (2021).

18. S, S. *Neural Networks: All YOU Need to Know* 2020. https://towardsdatascience.com/nns-aynk-c34efe37f15a (2021).

19. Brownlee, J. *Crash Course On Multi-Layer Perceptron Neural Networks* 2016. https://machinelearningmastery.com/neural-networks-crash-course/ (2021).

20. VanderPlas, J. *In Depth: k-Means Clustering | Python Data Science Handbook* https://jakevdp.github.io/PythonDataScienceHandbook/05.11-k-means.html (2021).

21. Nagesh Singh Chauhan. *What is Hierarchical Clustering?* https://www.kdnuggets.com/what-is-hierarchical-clustering.html/ (2021).

22. Sharma, A. *Principal Component Analysis (PCA) in Python* 2020. https://www.datacamp.com/community/tutorials/principal-component-analysis-in-python (2021).

23. Oracle. *Non-Negative Matrix Factorization* https://docs.oracle.com/cd/B28359_01/datamine.111/b28129/algo_nmf.htm#CHDJCEDJ (2021).

24. Manthiramoorthi, M. *Topic Modeling using Non Negative Matrix Factorization (NMF)* 2020. https://iq.opengenus.org/topic-modeling-nmf/ (2021).

25. Violante, A. *An Introduction to t-SNE with Python Example* 2018. https://towardsdatascience.com/an-introduction-to-t-sne-with-python-example-5a3a293108d1 (2021).

26. Kowalewski, M. *Data Cleaning In 5 Easy Steps* 2020. https://www.iteratorshq.com/blog/data-cleaning-in-5-easy-steps/ (2021).

27. Lakshmanan, S. *How, When and Why Should You Normalize/Standardize/Rescale Your Data?* 2020. https://medium.com/towards-artificial-intelligence/how-when-and-why-should-you-normalize-standardize-rescale-your-data-3f083def38ff (2021).

28. Huilgol, P. *9 Feature Transformation & Scaling Techniques| Boost Model Performance* 2020. https://www.analyticsvidhya.com/blog/2020/07/types-of-feature-transformation-and-scaling/ (2021).

29. Brownlee, J. *Why One-Hot Encode Data in Machine Learning?* 2017. https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/ (2021).

30. EliteDataScience. *Dimensionality Reduction Algorithms: Strengths and Weaknesses* 2017. https://elitedatascience.com/dimensionality-reduction-algorithms (2021).

31. GeeksforGeeks. *Genetic Algorithms* 2017. https://www.geeksforgeeks.org/genetic-algorithms/ (2021).

32. Saabas, A. *Selecting good features – Part IV: stability selection, RFE and everything side by side* https://blog.datadive.net/selecting-good-features-part-iv-stability-selection-rfe-and-everything-side-by-side/ (2021).

33. Malik, U. *Applying Filter Methods in Python for Feature Selection* https://stackabuse.com/applying-filter-methods-in-python-for-feature-selection/ (2021).

34. Brownlee, J. *How to Choose a Feature Selection Method For Machine Learning* 2019. https://machinelearningmastery.com/feature-selection-with-real-and-categorical-data/ (2021).

35. Brownlee, J. *Linear Discriminant Analysis for Dimensionality Reduction in Python* 2020. https://machinelearningmastery.com/linear-discriminant-analysis-for-dimensionality-reduction-in-python/ (2021).

36. Kapadia, S. *Topic Modeling in Python: Latent Dirichlet Allocation (LDA)* 2020. https://towardsdatascience.com/end-to-end-topic-modeling-in-python-latent-dirichlet-allocation-lda-35ce4ed6b3e0 (2021).

37. Sarkar, S. *Independent Component Analysis for Signal decomposition* 2020. https://medium.com/analytics-vidhya/independent-component-analysis-for-signal-decomposition-3db954ffe8aa (2021).

38. Lawton, G. *Autoencoders' example uses augment data for machine learning* https://searchenterpriseai.techtarget.com/feature/Autoencoders-example-uses-augment-data-for-machine-learning (2021).

39. Brownlee, J. *Autoencoder Feature Extraction for Classification* 2020. https://machinelearningmastery.com/autoencoder-for-classification/ (2021).

40. Brownlee, J. *Autoencoder Feature Extraction for Regression* 2020. https://machinelearningmastery.com/autoencoder-for-regression/ (2021).

41. Brownlee, J. *What is the Difference Between Test and Validation Datasets?* 2017. https://machinelearningmastery.com/difference-test-validation-datasets/ (2021).

42. scikit-learn. *Cross-validation: evaluating estimator performance* https://scikit-learn.org/stable/modules/cross_validation.html (2021).

43. Krishni. *K-Fold Cross Validation* 2018. https://medium.com/datadriveninvestor/k-fold-cross-validation-6b8518070833 (2021).

44. Worcester, P. *A Comparison of Grid Search and Randomized Search Using Scikit Learn* 2019. https://blog.usejournal.com/a-comparison-of-grid-search-and-randomized-search-using-scikit-learn-29823179bc85 (2021).

45. Brownlee, J. *Hyperparameter Optimization With Random Search and Grid Search* 2020. https://machinelearningmastery.com/hyperparameter-optimization-with-random-search-and-grid-search/ (2021).

46. Zivkovic, N. *14 Popular Machine Learning Evaluation Metrics* 2020. https://rubikscode.net/2020/10/19/14-popular-machine-learning-evaluation-metrics/ (2021).

47. Dembla, G. *Intuition behind Log-loss Score* 2020. https://towardsdatascience.com/intuition-behind-log-loss-score-4e0c9979680a (2021).

48. M, S. *What is a Pipeline in Machine Learning? How to create one?* 2019. https://medium.com/analytics-vidhya/what-is-a-pipeline-in-machine-learning-how-to-create-one-bda91d0ceaca (2021).

49. Lee, A. *ava-lee/MLIntroduction* 2021. https://github.com/ava-lee/MLIntroduction (2021).

50. Géron, A. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems* ("O'Reilly Media, Inc.", 2019).

51. Hastie, T., Tibshirani, R. & Friedman, J. *The Elements of Statistical Learning* https://web.stanford.edu/~hastie/Papers/ESLII.pdf (Springer, 2009).

52. Grus, J. *Data Science from Scratch: First Principles with Python* https://learning.oreilly.com/library/view/data-science-from/9781492041122/ ("O'Reilly Media, Inc.", 2015).

53. Google. *Machine Learning Crash Course* https://developers.google.com/machine-learning/crash-course (2021).