MACHINE LEARNING MODELS

XGBOOST

- *XGBoost* is an open-source software library which provides the gradient boosting framework for C++, Java, Python, R, and Julia.
- It now has integrations with *scikit-learn* for Python users
- https://machinelearningmastery.com/develop-first-xgboost-model-python-scikit-learn/ (How to Develop Your First XGBoost Model in Python with scikit-learn)
- https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/ (A Gentle Introduction to the Gradient Boosting Algorithm for Machine Learning)
- https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/ (A Gentle Introduction to XGBoost for Applied Machine Learning)

LightGBM

- https://lightgbm.readthedocs.io/en/latest/
- LightGBM uses the leaf-wise (best first) tree growth algorithm, while many other popular tools use depth-wise tree growth. Compared with depth-wise growth, the leaf-wise algorithm can converge much faster. However, the leaf-wise growth may be overfitting if not used with the appropriate parameters.
- LightGBM is a great implementation that is similar to XGBoost but varies in a few specific ways, especially in how it creates the trees.
- It offers some different parameters but most of them are very similar to their XGBoost counterparts.
- If you use the same parameters, you almost always get a very close score. In most cases, the training will be 2-10 times faster thoug
- *Why don't more people use it then?* XGBoost has been around longer and is already installed on many machines. LightGBM is rather new and didn't have a Python wrapper at first. The current version is easier to install and use so no obstacles here. Many of the more advanced users on Kaggle and similar sites already use LightGBM and for each new competition, it gets more and more coverage. Still, the starter scripts are often based aroudn XGBoost as people just reuse their old code and adjust a few parameters. I'm sure this will increase once there are a few more tutorials and guides on how to use it (most of the non-ScikitLearn guides currently focus on XGBoost or neural networks).

- https://www.analyticsvidhya.com/blog/2017/06/which-algorithm-takes-the-crown-light-gbm-vs-xgboost/ -> Comparison between XGBoost and LightGBM

Multiple Layer Perception

- A **multilayer perceptron** (MLP) is a class of feedforward artificial neural network. An MLP consists of at least three layers of nodes. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable.
- https://machinelearningmastery.com/neural-networks-crash-course/
- https://www.springboard.com/blog/beginners-guide-neural-network-in-python-scikit-learn-0-18/
- Multi-layer Perceptron is sensitive to feature scaling.
- http://benalexkeen.com/feature-scaling-with-scikit-learn/ -> Feature Scaling with scikit-learn