<https://towardsdatascience.com/how-to-compare-machine-learning-algorithms-ccc266c4777>

1. Logistic Regression : Logistic Regression is one of the oldest and most basic algorithms to solve a classification problem
2. Linear Discriminant Analysis
3. K-Nearest Neighbors: The k-nearest neighbors (KNN) algorithm is a supervised machine learning algorithm that can be used to solve both classification and regression problems. For KNN, it is known that it does not work so well with large datasets (high sample size) and in with many features (high dimensions) in particular.
4. Neural Network : Deep learning uses an artificial neural network that uses multiple layers to progressively extract higher level features from the training data. We are using a simple three-layer network without any optimisation, except the usage of a small validation datase
5. Naive Bayes : The Naive Bayes method is a supervised learning algorithm based on applying Bayes’ theorem with the “naive” assumption of conditional independence between every pair of features given the value of the class variable
6. Support Vector Machines: The Support Vector Machine is a simple algorithm for classification and regression tasks. It can provide high accuracy with less computation power very fast. Due to the large number of features, we are using the LinearSVC. It turned out that setting the regularisation parameter C=0.0001 improves the quality of the prediction and reduces overfitting.
7. Random Forest : The Random Forest Algorithm is another frequently used ensemble learning classifier which uses multiple decision trees. The Random Forest classifier is basically a modified bagging algorithm of a Decision Tree that selects the subsets differently. I found out that max\_depth=10 is a good value for this feature-rich dataset
8. Decision Tree : A Decision Tree is a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules (if-else) inferred from the data features.
9. Gradient Boosting : he idea behind boosting comes from the intuition that weak learners could be modified in order to become better. Gradient Boosting is an iterative functional gradient algorithm, i.e an algorithm which minimizes a loss function by iteratively choosing a function that points towards the negative gradient; a weak hypothesis.
10. AdaBoost : n general, we can’t use Boosting to improve a completely overfitted model with score = 1. To apply Boosting, we first have to tweak the decision tree classifier a bit. It took me some trial and error until I got the best parameters for the Decision Tree and the AdaBoost Classifier. I am sure you can further improve by playing around a bit more.

<https://learn.microsoft.com/en-us/azure/machine-learning/how-to-select-algorithms?view=azureml-api-1>

Certainly! Here's a comparison of some popular supervised learning algorithms:

* + Pros: Cons: Decision Trees:
  + Pros: Easy to understand and interpret, can handle both numerical and categorical data, can capture non-linear relationships.
  + Cons: Prone to overfitting, may not generalize well to unseen data.

1. Random Forest:
   * Pros: Robust performance, handles complex data, reduces overfitting through ensemble of decision trees.
   * Cons: Less interpretable compared to individual decision trees, computationally expensive for large datasets.
2. Support Vector Machines (SVM):
   * Pros: Effective for binary classification, can handle high-dimensional data, works well with small to medium-sized datasets.
   * Cons: Can be sensitive to the choice of kernel function, less suitable for large datasets due to computational complexity.
3. Naive Bayes:
   * Pros: Simple and computationally efficient, works well with text classification and high-dimensional data, handles missing values well.
   * Cons: Assumes independence between features, may not capture complex relationships.
4. Neural Networks:
   * Pros: Capable of learning complex patterns, suitable for a wide range of tasks, can handle large amounts of data.
   * Cons: Requires large amounts of data and computational resources, prone to overfitting without proper regularization, less interpretable.
5. High Bias, Low Variance:
6. Low Bias, High Variance:
   * Decision Trees: Decision trees can have low bias as they can capture complex relationships in the data. However, they are prone to overfitting and can have high variance, leading to poor generalization on unseen data.
7. Trade-off between Bias and Variance:
   * Random Forest: Random forests mitigate the overfitting problem of decision trees by combining multiple trees and reducing variance. They tend to have a balance between bias and variance, resulting in robust performance.
   * Support Vector Machines (SVM): SVMs aim to find an optimal hyperplane that maximizes the margin between classes. They strike a balance between bias and variance by using the kernel trick and regularization parameters to control the complexity of the model.
8. Bayesian Methods:
   * Naive Bayes: Naive Bayes assumes independence between features given the target variable and has low bias. While it may oversimplify the relationships, it typically has low variance and can handle high-dimensional data well.
9. Neural Networks:
   * Neural networks are highly flexible models that can learn complex patterns, making them capable of low bias. However, they are prone to overfitting and can have high variance, especially with large and complex networks. Regularization techniques, such as dropout or weight decay, can help control variance.

In general, models with high bias tend to underfit the data, while models with high variance tend to overfit. The goal is to strike a balance between bias and variance to achieve good generalization on unseen data. Understanding the bias-variance trade-off helps in selecting the appropriate algorithm and tuning its parameters to achieve the desired model performance.

|  | Pros | cons | bias/variance |
| --- | --- | --- | --- |
| Linear Regression | Simple, interpretable, computationally efficient, works well with linear relationships. | Assumes a linear relationship between features and target variable, may not handle complex data. | Linear Regression: Linear regression models have a high bias due to their assumption of a linear relationship between features and target variable. They may not capture complex patterns but tend to have low variance and can generalize well. |
|  |  |  |  |

Logistic regression

Training and inference are two key steps in the process of building and using a logistic regression model. Let's explore each step in detail:

Training:

1. Training a logistic regression model involves using labeled data to estimate the parameters of the model. The labeled data consists of input variables (features) and corresponding binary or categorical target variables (labels). The goal is to learn the relationship between the input variables and the probability of a certain outcome.

Here's an overview of the training process:

a. Data Preparation: The first step is to prepare the training data. This involves selecting relevant features, preprocessing the data (e.g., handling missing values, scaling, encoding categorical variables), and splitting the data into a training set and a validation set for model evaluation.

b. Model Initialization: Initialize the logistic regression model with random values for its parameters, including the intercept term and coefficients associated with each feature.

c. Objective Function: Logistic regression uses the maximum likelihood estimation (MLE) technique to find the optimal parameters. The objective function, also known as the log-loss or cross-entropy loss, measures the difference between the predicted probabilities and the actual labels. The goal is to minimize this loss function.

d. Optimization Algorithm: The model parameters are iteratively updated using an optimization algorithm, such as gradient descent or its variants (e.g., stochastic gradient descent). The algorithm adjusts the parameters in the direction that reduces the loss function.

e. Iterative Training: The optimization algorithm continues to update the parameters by repeatedly calculating the gradient of the loss function with respect to the parameters and adjusting them accordingly. This process continues until the algorithm converges or a stopping criterion is met (e.g., a maximum number of iterations or a threshold for the improvement in the loss function).

f. Model Evaluation: After training, the model's performance is evaluated using the validation set or other evaluation metrics, such as accuracy, precision, recall, or area under the receiver operating characteristic curve (AUC-ROC). If the performance is satisfactory, the model can be considered ready for inference.

Inference:

1. Inference refers to the process of using the trained logistic regression model to make predictions on new, unseen data. Here's an overview of the inference process:

a. Data Preprocessing: Similar to the training phase, the new data needs to be preprocessed to ensure compatibility with the model's input format. This includes feature selection, scaling, and encoding categorical variables, following the same preprocessing steps as during training.

b. Model Prediction: Once the data is preprocessed, the logistic regression model calculates the weighted sum of the features, adds the intercept, and applies the logistic function (sigmoid) to obtain the predicted probabilities. The predicted probabilities represent the likelihood of the binary outcome or the probabilities for each category in a multi-class setting.

c. Thresholding (Binary Classification): If the logistic regression model is used for binary classification, a threshold is applied to the predicted probabilities to convert them into binary predictions (e.g., 0 or 1). Commonly, a threshold of 0.5 is used, where predicted probabilities above 0.5 are classified as one category and those below as the other category.

d. Model Evaluation: The predictions made by the logistic regression model on the new data can be evaluated using appropriate metrics for the specific task, such as accuracy, precision, recall, or F1 score.

That's a high-level explanation of the training and inference process for a logistic regression model. Keep in mind that logistic regression assumes a linear relationship between the features and the log-odds of the outcome, and it's commonly used for binary classification problems, although it can be extended to handle multi-class classification tasks as well.

—--------------

The parameters of a logistic regression model include the following:

1. Intercept (Bias term): The intercept term, denoted as β₀, is a constant term added to the weighted sum of the features. It accounts for the baseline probability of the outcome when all the features are zero.
2. Coefficients (Weights): For each feature in the model, there is a corresponding coefficient, denoted as β₁, β₂, β₃, ..., βᵣ (where r is the number of features). These coefficients represent the impact or contribution of each feature on the log-odds of the outcome. Each coefficient indicates the change in the log-odds ratio associated with a one-unit change in the corresponding feature, assuming the other features are held constant.

Mathematically, the logistic regression model can be represented as:

logit(p) = β₀ + β₁x₁ + β₂x₂ + ... + βᵣxᵣ

where logit(p) represents the log-odds (logarithm of the odds) of the outcome variable, p is the probability of the outcome, and x₁, x₂, ..., xᵣ are the feature values.

The logistic regression model uses these parameters to estimate the log-odds of the outcome and then applies the logistic function (sigmoid) to convert the log-odds into probabilities.

During the training phase, the logistic regression model learns the optimal values for these parameters by minimizing the loss function (e.g., log-loss or cross-entropy loss) through an optimization algorithm (e.g., gradient descent). The optimized parameters reflect the relationship between the features and the probability of the outcome in the training data. These parameters are then used during the inference phase to make predictions on new, unseen data.

—--------

Maximum likelihood estimation (MLE) is a statistical method used to estimate the parameters of a probabilistic model by maximizing the likelihood function. It is a common approach for parameter estimation in various statistical models, including logistic regression.

Here's an explanation of maximum likelihood estimation:

Likelihood Function:

1. In MLE, we start by defining a likelihood function that represents the probability of observing the given data as a function of the model parameters. For example, in logistic regression, the likelihood function captures the probability of obtaining the observed labels (target variable) given the features and the model parameters.

Assumptions:

1. MLE assumes that the observations in the dataset are independent and identically distributed (i.i.d.). This means that each data point is assumed to be generated independently from the same underlying distribution.

Log-Likelihood Function:

1. To simplify calculations, the likelihood function is often transformed into its logarithm, resulting in the log-likelihood function. Taking the logarithm allows us to convert products (from the likelihood function) into sums, making optimization easier.

Maximizing the Log-Likelihood:

1. The goal of MLE is to find the values of the model parameters that maximize the log-likelihood function. In other words, we want to find the parameter values that make the observed data most likely to occur according to the model.

Optimization Algorithm:

1. To maximize the log-likelihood, optimization algorithms are employed. The most commonly used algorithm is gradient descent, or its variants, which iteratively adjusts the parameter values in the direction that increases the log-likelihood. The algorithm calculates the gradient of the log-likelihood function with respect to each parameter and updates the parameter values accordingly. This process continues until the algorithm converges or a stopping criterion is met.

Estimation of Parameters:

1. After the optimization process, the values of the parameters that maximize the log-likelihood function are obtained. These estimated parameter values represent the best-fit values given the observed data and the assumed model. In logistic regression, these parameters correspond to the intercept term and the coefficients associated with each feature.

Inference:

1. Once the model parameters are estimated using MLE, the model can be used for various purposes, such as making predictions on new data, estimating probabilities, or understanding the relationships between the features and the target variable.

In summary, maximum likelihood estimation is a method for estimating the parameters of a statistical model by finding the parameter values that maximize the likelihood (or log-likelihood) function. It is a powerful and widely used approach for parameter estimation in various models, including logistic regression.