COMPARISON OF ADVANCED ALGORITHMS

What are **Ensemble Methods**??

Ensemble learning is a machine learning paradigm where multiple models (often called "weak learners") are trained to solve the same problem and combined to get better results. The main hypothesis is that when weak models are correctly combined, we can obtain more accurate and/or robust models.

Two families of ensemble methods are usually distinguished:

- In averaging methods, the driving principle is to build several estimators independently and then to average their predictions. On average, the combined estimator is usually better than any of the single base estimator because its variance is reduced.

Examples: Bagging methods-Random Forest

- By contrast, in boosting methods, base estimators are built sequentially and one tries to reduce the bias of the combined estimator. The motivation is to combine several weak models to produce a powerful ensemble.

Examples: Boosting methods-XGBoost

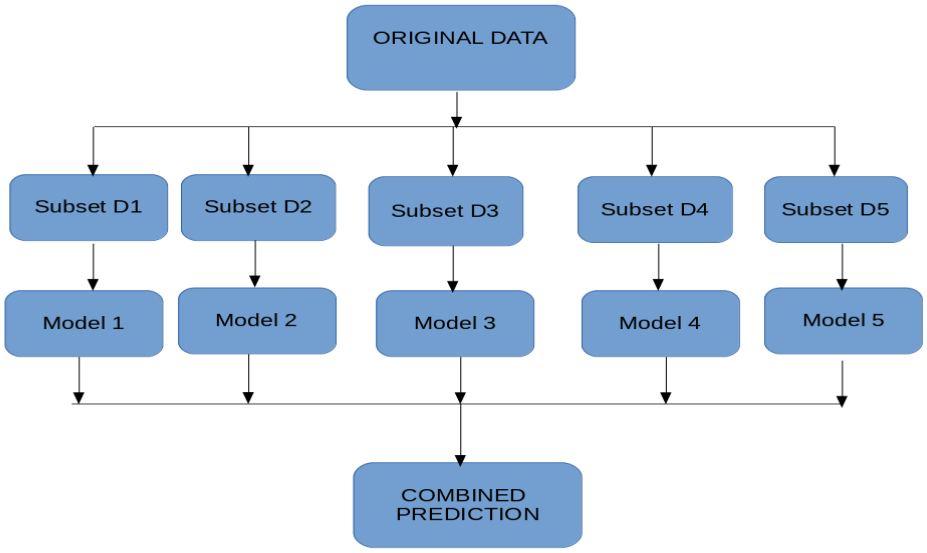
Here we are discussing following advanced algorithms:

1. Bagging
2. AdaBoost
3. GBM- Gradient Boosting Machines
4. Light BGM
5. XGBoost (Extreme Gradient)
6. CatBoost

1. Bagging:

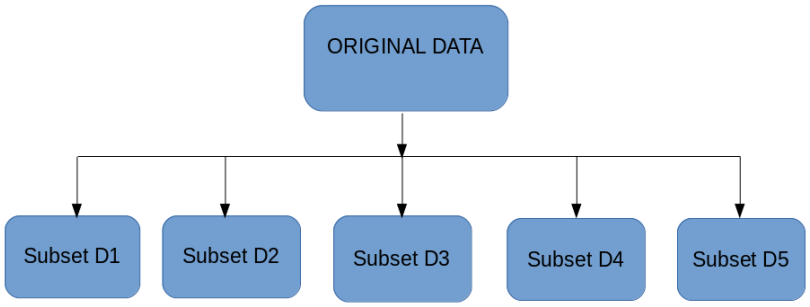
- **Bagging** is an ensemble technique mainly used to reduce the variance of our predictions by combining the result of multiple classifiers modelled on different sub-samples of the same data set.
- Bagging is also called as **Bootstrap Aggregation**, is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression.
- Bagging was proposed by **Leo Breiman** in 1994 to improve classification by combining classifications of randomly generated training sets.
- Bagging is effective because you are improving the accuracy of a single model by using multiple copies of it trained on different sets of data.
- Bagging is not recommended on models that have a high bias. In such cases, boosting (Adaboost) is used which goes a step ahead and eliminates the effect of a high bias present in the baseline model.

- Bagging is used to improve accuracy in **statistical classification** and **regression**.
- Bootstrapping is a sampling technique in which we create subsets of observations from the original dataset, **with replacement**. The size of the subsets is the same as the size of the original set.
- Bagging (or Bootstrap Aggregating) technique uses these subsets (bags) to get a fair idea of the distribution (complete set). The size of subsets created for



bagging may be less than the original set.

1. Multiple subsets are created from the original dataset, selecting observations with replacement.
2. A base model (weak model) is created on each of these subsets.
3. The models run in parallel and are independent of each other.
4. The final predictions are determined by combining the predictions from all the mod



- **Random Forest** basically a Bagging Technique. It splits the model into different decision tress having low bias and high variance

- **Random forest** improves on **bagging** because it decorrelates the trees with the introduction of splitting on a **random** subset of features. This means that at each split of the tree, the model considers only a small subset of features rather than all of the features of the model.

- **Advantages:**

  - ✓ Bagging offers the advantage of allowing many weak learners to combine efforts to outdo a single strong learner.
  - ✓ It also helps in the reduction of variance, hence eliminating the overfitting of models in the procedure.

- **Disadvantages:**

  - ✓ It introduces a loss of interpretability of a model. The resultant model can experience lots of bias when the proper procedure is ignored.
  - ✓ Despite bagging being highly accurate, it can be computationally expensive and this may discourage its use in certain instances.

- **Similarities Between Bagging and Boosting**

  - ✓ Both are ensemble methods to get N learners from 1 learner.
  - ✓ Both generate several training data sets by random sampling.
  - ✓ Both make the final decision by averaging the N learners (or taking the majority of them i.e Majority Voting).
  - ✓ Both are good at reducing variance and provide higher stability.

- **Research papers URL:**

1. https://www.sciencedirect.com/science/article/pii/S0022030212007783
2. https://www.sciencedirect.com/science/article/pii/S2405896319302009

➢ **Differences Between Bagging and Boosting**

S.NO	BAGGING	BOOSTING
1.	Simplest way of combining predictions that belong to the same type.	A way of combining predictions that belong to the different types.
2.	Aim to decrease variance, not bias.	Aim to decrease bias, not variance.
3.	Each model receives equal weight.	Models are weighted according to their performance.
4.	Each model is built independently.	New models are influenced by performance of previously built models.
5.	Different training data subsets are randomly drawn with replacement from the entire training dataset.	Every new subsets contains the elements that were misclassified by previous models.
6.	Bagging tries to solve over-fitting problem.	Boosting tries to reduce bias.
7.	If the classifier is unstable (high variance), then apply bagging.	If the classifier is stable and simple (high bias) the apply boosting.
8.	Random forest.	Gradient boosting.

Reference :
https://stats.stackexchange.com/questions/18891/bagging-boosting-and-stacking-in-machine-learning

2. AdaBoost:

- **AdaBoost**, short for Adaptive Boosting, is a machine learning meta-algorithm formulated by **Yoav Freund** and **Robert Schapire**, who won the 2003 Gödel Prize for their work.
- It can be used in conjunction with many other types of learning algorithms to improve performance.
- The output of the other learning algorithms ('weak learners') is combined into a weighted sum that represents the final output of the boosted classifier.
- AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified by previous classifiers.
- AdaBoost is sensitive to noisy data and outliers. In some problems, it can be less susceptible to the overfitting problem than other learning algorithms.
- The individual learners can be weak, but as long as the performance of each one is slightly better than random guessing, the final model can be proven to converge to a strong learner.
- To build a **AdaBoost classifier**, imagine that as a first base classifier we train a Decision Tree algorithm to make predictions on our training data. Now, following the methodology of AdaBoost, the weight of the misclassified

training instances is increased. The second classifier is trained and acknowledges the updated weights and it repeats the procedure over and over again.

- The common algorithms with AdaBoost used are **decision trees** with level one. A weak learner is a classifier or predictor which performs relatively poor in terms of accuracy.

- **Advantages:**

  - ✓ Very good use of weak classifiers for cascading.
  - ✓ Different classification algorithms can be used as weak classifiers.
  - ✓ AdaBoost has a high degree of precision.
  - ✓ Relative to the bagging algorithm and Random Forest Algorithm, AdaBoost fully considers the weight of each classifier.

- **Disadvantages:**

  - ✓ The number of AdaBoost iterations is also a poorly set number of weak classifiers, which can be determined using cross-validation.
  - ✓ Data imbalance leads to a decrease in classification accuracy.
  - ✓ Training is time consuming, and it is best to cut the point at each re-selection of the current classifier.

- **Research papers URL:**
  1. http://rob.schapire.net/papers/explaining-adaboost.pdf
  2. https://www.researchgate.net/publication/321583409_AdaBoost_typical_Algorithm_and_its_application_research

3. Gradient Boosting:

- **Gradient Boosting** works by sequentially adding the previous predictors under-fitted predictions to the ensemble, ensuring the errors made previously are corrected.
- Contrary to AdaBoost, which takes the instance weights at every interaction, this method tries to fit the new predictor to the residual errors made by the previous predictor.
- The idea of gradient boosting originated in the observation by **Leo Breiman** that boosting can be interpreted as an optimization algorithm on a suitable cost function. Explicit regression gradient boosting algorithms were subsequently developed by **Jerome H. Friedman**, simultaneously with the more general functional gradient boosting perspective of **Llew Mason**, **Jonathan Baxter**, **Peter Bartlett** and **Marcus Frean**.
- Gradient boosting is one of the most powerful techniques for building predictive models.

- Gradient Boosting is ML technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction model, typically **decision trees**.
- It is typically used with decision trees (especially CART trees) of a fixed size as base learners.
- Gradient boosting involves three elements:
  - ✓ A loss function to be optimized.
  - ✓ A weak learner to make predictions.
  - ✓ An additive model to add weak learners to minimize the loss function.
- It can be used in the field of learning to rank. The commercial web search engines *Yahoo* and *Yandex* use variants of gradient boosting in their machine-learned ranking engines.

- **Advantages:**

  - ✓ Often provides predictive accuracy that cannot be beat.
  - ✓ Lots of flexibility - can optimize on different loss functions and provides several hyperparameter tuning options that make the function fit very flexible.
  - ✓ No data pre-processing required - often works great with categorical and numerical values as is.
  - ✓ Handles missing data - imputation not required.

- **Disadvantages:**

  - ✓ GBMs will continue improving to minimise all errors. This can overemphasise outliers and cause overfitting. Must use cross-validation to neutralize.
  - ✓ Computationally expensive - GBMs often require many trees (>1000) which can be time and memory exhaustive.
  - ✓ The high flexibility results in many parameters that interact and influence heavily the behavior of the approach (number of iterations, tree depth, regularization parameters, etc.). This requires a large grid search during tuning.
  - ✓ Less interpretable although this is easily addressed with various tools (variable importance, partial dependence plots, LIME, etc.).

- **Research papers URL:**
  1. https://www.cis.upenn.edu/~mkearns/papers/boostnote.pdf
  2. https://www.researchgate.net/publication/259653472_Gradient_Boosting_Machines_A_Tutorial

- The major difference between AdaBoost and Gradient Boosting Algorithm is how the two algorithms identify the shortcomings of weak learners (eg. decision trees). While the AdaBoost model identifies the shortcomings by using high weight data points, gradient boosting performs the same by using gradients

in the loss function (*y=ax+b+e), e needs a special mention as it is the error term).*

- A great **application** of GBM is anomaly detection in Supervised learning settings where data is often highly unbalanced such as DNA sequences, credit card transactions or Cybersecurity Gradient boosting classifiers are specific types of algorithms that are used for classification task.

4. Light GBM:

- Light GBM is a fast, distributed, high-performance gradient boosting framework based on decision tree algorithm, used for ranking, classification and many other machine learning tasks.

- Light GBM grows tree vertically while other algorithm grows trees horizontally meaning that Light GBM grows tree leaf-wise while other algorithm grows level-wise. It will choose the leaf with max delta loss to grow. When growing the same leaf, Leaf-wise algorithm can reduce more loss than a level-wise algorithm.
- Light GBM is prefixed as 'Light' because of its **high speed** and can handle the **large size** of data and takes **lower memory** to run.
- Light GBM focuses on **accuracy** of results, it also supports **GPU learning.**
- Implementation of LGBM is easy, the only complicated thing is parameter tuning. It covers more than 100 parameters.
- LGBM is contributed by **Microsoft.**
- Important parameters of LGBM
    a) **Task**: default value = train ; options = train , prediction ; Specifies the task we wish to perform which is either train or prediction.
    b) **Application**: default=regression, type=enum, options= options :
        a. Regression: Perform regression task.
        b. Binary: Binary Classification.
        c. Multiclass: Multiclass classification.
        d. Landmark: Landmark applications.
    c) **data**: type=string; training data , LightGBM will train from this data.
    d) **num_iterations**: number of boosting iterations to be performed ; default=100; type=int
    e) **num_leaves**: number of of leaves in one tree ; default = 31 ; type =int
    f) **device**: default=cpu; options = gpu,cpu. Device on which we want to train our model. Choose GPU for faster training.
    g) **max_depth**: Specify the max depth to which tree will grow. This parameter is used to deal with overfitting
    h) **min_data_in_leaf**: Min number of data in one leaf.

i) **feature_fraction**: default=1, specifies the fraction of features to be taken for each iteration
j) **bagging_fraction**: default=1, specifies the fraction of data to be used for each iteration and is generally used to speed up the training and avoid overfitting.
k) **min_gain_to_split**: default=1, min gain to split.
l) **max_bin**: max number of bins to bucket the feature values.
m) **min_data_in_bin**: min no of data in one bin.
n) **num_threads**: default=OpenMP_default, type=int; Number of threads for LGBM.
o) **label**: type=string ; specify the label column
p) **categorical_feature**: type=string; specify the categorical features we want to use for training our model
q) **num_class**: default=1; type=int ; used only for multi-class classification.

- Light GBM is great using with decision trees.
- **Advantages:**

  - ✓ **Faster Training speed and higher efficiency:** Light GBM use histogram based algorithm i.e it buckets continuous feature values into discrete bins which fasten the training procedure.
  - ✓ **Lower Memory Usage**: Replaces continuous values to discrete bins which result in lower memory usage.
  - ✓ **Better accuracy than any other boosting algorithm**: It produces much more complex trees by following leaf wise split approach rather than a level-wise approach which is the main factor in achieving higher accuracy. However, it can sometimes lead to overfitting which can be avoided by setting the max_depth parameter.
  - ✓ **Compatibility with large datasets**: It is capable of performing equally good with large datasets with a significant reduction in training time as compared to XGBOOST.
  - ✓ **Parallel learning supported.**

- **Disadvantages:**

  - ✓ **Narrow user base**: but that is changing fast.
    This algorithm apart from being more accurate and time-saving than XGBOOST has been limited in usage due to less documentation available.

- **Reaserch paper URL:**

  1. https://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf
  2. https://www.researchgate.net/publication/332268924_Light_GBM_Machine_Learning_Algorithm_to_Online_Click_Fraud_Detection

5. XG Boost:

- **XGBoost** is a decision-tree based ensemble machine learning algorithm that uses a gradient boosting framework. In prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform all other algorithms or frameworks.
- XGBoost is an implementation of gradient boosted decision **trees** designed for speed and performance.
- XGBoost initially started as a research project by **Tianqi Chen** as part of the Distributed (Deep) Machine Learning Community (DMLC) group. Initially, it began as a terminal application which could be configured using a libsvm configuration file.
- **XGBoost** is a scalable and accurate implementation of gradient boosting machines and it has proven to push the limits of computing power for boosted trees algorithms as it was built and developed for the sole purpose of model performance and computational speed.
- **Xgboost** is **good** for tabular data with a small number of variables, whereas neural nets based deep learning is **good** for images or data with a large number of variables. Both methods are great in their own rights and are well respected.
- **XGBoost** is also known as the regularised version of **GBM**. This framework includes built-in L1 and L2 regularisation which means it can prevent a model from overfitting. Traditionally, **XGBoost** id slower **than** lightGBM but it achieves **faster** training via Histogram binning.
- **LightGBM** improves on **XGBoost**. The **LightGBM** paper uses **XGBoost** as a baseline and outperforms it in training speed and the dataset sizes it can handle. The accuracies are comparable. **LightGBM** in some cases reaches it's top accuracy in under a minute and while only reading a fraction of the whole dataset.

- The algorithm differentiates itself in the following ways:
  a) wide range of applications: Can be used to solve regression, classification, ranking, and user-defined prediction problems.
  b) Portability: Runs smoothly on Windows, Linux, and OS X.
  c) Languages: Supports all major programming languages including C++, Python, R, Java, Scala, and Julia.
  d) Cloud Integration: Supports AWS, Azure, and Yarn clusters and works well with Flink, Spark, and other ecosystems.

- **Advantages:**

  - ✓ **Regularization**: XGBoost has in-built L1 (Lasso Regression) and L2 (Ridge Regression) regularization which prevents the model from overfitting. That is why, XGBoost is also called regularized form of GBM (Gradient Boosting Machine).

- ✓ **Parallel Processing**: XGBoost utilizes the power of parallel processing and that is why it is much faster than GBM. It uses multiple CPU cores to execute the model.
- ✓ **Handling Missing Values**: XGBoost has an in-built capability to handle missing values. When XGBoost encounters a missing value at a node, it tries both the left and right hand split and learns the way leading to higher loss for each node. It then does the same when working on the testing data.
- ✓ **Cross Validation**: XGBoost allows user to run a cross-validation at each iteration of the boosting process and thus it is easy to get the exact optimum number of boosting iterations in a single run. This is unlike GBM where we have to run a grid-search and only a limited values can be tested.
- ✓ **Effective Tree Pruning**: A GBM would stop splitting a node when it encounters a negative loss in the split. Thus it is more of a greedy algorithm. XGBoost on the other hand make splits upto the max_depth specified and then start pruning the tree backwards and remove splits beyond which there is no positive gain.

- **Disadvantages:**
  - ✓ Only work with numeric features.
  - ✓ Leads to overfitting if hyperparameters are not tuned properly.

- **Research papers:**
  1. https://www.kdd.org/kdd2016/papers/files/rfp0697-chenAemb.pdf
  2. https://www.researchgate.net/publication/318132203_Experimenting_XGBoost_Algorithm_for_Prediction_and_Classification_of_Different_Datasets

6. CatBoost:

- CatBoost is an algorithm for **gradient boosting on decision trees**. It is developed by **Yandex** researchers and engineers, and is used for search, recommendation systems, personal assistant, self-driving cars, weather prediction and many other tasks at Yandex and in other companies, including CERN, Cloudflare, Careem taxi. It is in open-source and can be used by anyone.
- CatBoost is a high-performance open source library for gradient boosting on decision trees. It can easily integrate with deep learning frameworks like Google's TensorFlow and Apple's Core ML.
- CatBoost can work with diverse data types to help solve a wide range of problems that businesses face today. To top it up, it provides best-in-class accuracy.

- It is especially powerful in two ways:

- ✓ It yields state-of-the-art results without extensive data training typically required by other machine learning methods.
- ✓ Provides powerful out-of-the-box support for the more descriptive data formats that accompany many business problems.

- "CatBoost" name comes from two words "**Category**" and "**Boosting**".
- The library works well with multiple Categories of data, such as audio, text, image including historical data.

- **Advantages**:

  - ✓ **Performance:** CatBoost provides state of the art results and it is competitive with any leading machine learning algorithm on the performance front.
  - ✓ **Handling Categorical features automatically:** We can use CatBoost without any explicit pre-processing to convert categories into numbers. CatBoost converts categorical values into numbers using various statistics on combinations of categorical features and combinations of categorical and numerical features.
  - ✓ **Robust:** It reduces the need for extensive hyper-parameter tuning and lower the chances of overfitting also which leads to more generalized models. Although, CatBoost has multiple parameters to tune and it contains parameters like the number of trees, learning rate, regularization, tree depth, fold size, bagging temperature and others.
  - ✓ **Easy-to-use:** You can use CatBoost from the command line, using an user-friendly API for both Python and R.

- **Disadvantages**:
  - ✓ Unlike stochastic gradient and neural network models CatBoost requires all data to be readily available in memory for fast random sampling.
  - ✓ Memory usage is indeed.

- **Research papers URL**:
  1. https://papers.nips.cc/paper/7898-catboost-unbiased-boosting-with-categorical-features.pdf
  2. https://arxiv.org/pdf/1706.09516.pdf

Submitted By:
SHABIN K
mailbox.shabin@gmail.com