1. Contents
2. Acknowledgement
3. Abstract
4. List of Tables
5. List of Figures
6. Introduction
   1. Background
   2. Challenges
   3. Objective
   4. Motivation
7. Converting Categorical features to numeric: Encoding Techniques
   1. Related Work
   2. Classifying methods of encoding
8. GBDT
   1. GBDT Overview
   2. State-of-the-art in GBDT
   3. Comparison of XGBoost, LightGBM and CatBoost
   4. Related Work
9. Methodology
   1. Datasets
   2. Experimental Design
   3. Evaluation Metrics
10. Results
    1. One-Hot Encoding
    2. Mean Target Encoding
    3. Mixed Encoding
    4. Native methods for categorical features
11. Security and Privacy Considerations
12. Conclusion
13. Future Work

Bibliography

1. Contents
2. Acknowledgement
3. Abstract
4. List of Tables
5. List of Figures
6. Introduction
   1. Background

With rising connectivity of the world, there is an unprecedented and exponential increase in the amount of data being collected every minute. The internet population grew 9% during 2018-2019. Apart from internet web searches, this rise in data available can be attributed to social media websites, communication channels such as emails, whatsapp and skype, digital photographs, services such as uber, spotify, Wikipedia, and internet of things. The International Data Corporation predicts that by 2025, total digital data created would be a staggering 163 zettabytes, all attributing to growing number of devices and internet [1]. In fact, data has been aptly called the new oil of 21st century.

With such amounts of data, analytics, data mining and knowledge discovery applications have risen tremendously. Spam detecting systems protect our emails by learning from spam data and user responses; customer churn prediction helps businesses to analyse trends in customer behaviour and leverage the patterns for growth; cancer cells identification aids in early detection of life-threatening conditions; sentiment analysis identifies the propensity of a population; fraud detection systems prevent malicious activities at banks; anomaly detection systems can help particle physicists in uncovering events leading to advancements in human knowledge. All of these, and even more applications, depend on ability of system to capture the complex interdependencies and scaling capacity.

While the data being fed to these classification systems is highly complex with a mixture of numerical, categorical and time-series features, there are missing values and non-linear interactions among the features. The problem has been termed as the curse of dimensionality [2]. Analysing such high-dimensional data is a statistical challenge. Considering that the categorical features need to be converted to numeric for statistical analyses, it becomes imperative to be able to identify best practices to encode as much information contained in categorical data as possible, into numerical data. As such, machine learning classification problems become significantly harder.

For the purpose of classification problems, tree-based methods have become a default standard in recent times. This is due to their superior performance over the classical classification methods such as logistic regression and support vector machines. Ensemble of decision trees, Random Forest, have helped in limiting the problem of overfitting. Though Random Forest is still a popular algorithm, gradient tree boosting or more popularly, gradient boosting decision trees (GBDT) have outperformed all other algorithms in a variety of challenging tasks. These systems have been able to capture the complex nature of data and have proved to be scalable as well in a multitude of applications.

Ogutu et. al. [3] compared algorithms for predicting genomic breeding values and discovered that stochastic gradient boosting with decision trees outperformed random forest and support vector machines. Georganos et. al. [4] proved that GBDT outperformed random forest and SVM in land use-land cover application in larger sample sizes. Golden et. al. [5] compared GBDT with RF to predict presence of foodborne pathogens in soil, and found that GBDT outperformed RF significantly. Rao et. al. [6] used Ant bee colony method coupled with GBDT for feature selection in six high dimensional datasets, and found that the classification accuracy was significantly improved. Pan et. al. [7] compared a modified version of GBDT with RF and SVM to predict association of Single amino acid variations with diseases, and noted that the performance of their variation of GBDT was improved over other algorithms.

These results indicate that GBDTs, in fact, have higher better performance as compared to other algorithms. In time, however, there have been improvements and implementations of the GBDT algorithm. The most notable and popular ones currently are eXtreme Gradient Boosting (XGBoost) [], LightGBM [] and CatBoost []. Since these algorithms are fairly recent, with CatBoost introduced in 2018, there has not been an extensive comparison among these implementations of GBDTs. On top of this, there are multiple methods of encoding the categorical variables to numerical. It becomes ambiguous for a machine learning practitioner to select a combination of encoding technique and classification algorithm for their prediction task.

* 1. Challenges

While there is no dearth of data in present times, finding several suitable open-access datasets is still a task. In order to make the comparison of algorithms fair, it is imperative that the datasets be diverse in number of observations, number of features, number of label classes, have both numeric and categorical features and the categorical features have high cardinality. For this thesis, the datasets were carefully selected from reliable sources such as UCI Machine Learning Repository, Kaggle datasets and data.world, and also from government websites publishing open-access data for research.

All the GBDT algorithms considered in the thesis have a tens of hyperparameters that can be adjusted for best fit of the model to the dataset. However, searching for these is computationally expensive and time-taking task. GridSearchCV is a module provided by Scikit-learn [8] which iterates over a list of hyperparameters to find the optimum ones. This was used in the thesis.

Finally, since the thesis does a comparison of performance of GBDT algorithms with different encoding techniques on several datasets, the results are three-dimensional. In order to present the findings in an informative yet intuitive way, a creative visualization is needed. The thesis does so in the last part of results section.

[1] <https://www.forbes.com/sites/nicolemartin1/2019/08/07/how-much-data-is-collected-every-minute-of-the-day/#29902a923d66>

[2] Donoho DL et al (2000) High-dimensional data analysis: the curses and blessings of dimensionality. AMS Math Challenges Lecture, pp 1–32

[3] Ogutu, J.O., Piepho, H. & Schulz-Streeck, T. A comparison of random forests, boosting and support vector machines for genomic selection. BMC Proc 5, S11 (2011). <https://doi.org/10.1186/1753-6561-5-S3-S11>

[4] S. Georganos, T. Grippa, S. Vanhuysse, M. Lennert, M. Shimoni and E. Wolff, "Very High Resolution Object-Based Land Use–Land Cover Urban Classification Using Extreme Gradient Boosting," in IEEE Geoscience and Remote Sensing Letters, vol. 15, no. 4, pp. 607-611, April 2018, doi: 10.1109/LGRS.2018.2803259.

[5] Golden, C. E., Rothrock, M. J., & Mishra, A. (2019). Comparison between random forest and gradient boosting machine methods for predicting Listeria spp. prevalence in the environment of pastured

[6] Rao, H., Shi, X., Rodrigue, A. K., Feng, J., Xia, Y., Elhoseny, M., … Gu, L. (2018). Feature selection based on artificial bee colony and gradient boosting decision

[7] Pan, Y., Liu, D., & Deng, L. (2017). Accurate prediction of functional effects for variants by combining gradient tree boosting with optimal neighborhood properties. PLOS ONE, 12(6), e0179314. doi:10.1371/journal.pone.0179314

[8] <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html>

* 1. Objective

The main objective of this thesis is to identify if there is one reliable combination of categorical encoding and GBDT algorithm that can be used with high confidence in classification tasks when the data has high cardinality features. Specifically, we aim to understand existing methods of categorical feature encoding as well as GBDT algorithms. We dive deep into the implementations of state-of-art GBDTs and try to identify why some algorithms give better performance on certain datasets, while not on others.

* 1. Motivation

It is a widely accepted fact that in designing a machine learning solution, data pre-processing step consumes over 80% of the time required. Also, the cleaned data required is expected to be completely numeric. This requires that the categorical features in data be converted or encoded to numeric features. However, this causes problems when one or more categorical features have a high number of categories within them. The number of categories present in a feature is called as cardinality. This problem, also known as high-cardinality, is Achilles’ heel of tree-based methods of classification. The splitting of an internal node over categorical variables is a difﬁcult combinatorial problem. For instance, features such as *zip\_code* and *user\_id* need to be treated as categorical but can have thousands of unique values within a dataset.

Most state-of-the-art algorithms are all gradient boosting decision trees. Though the underlying method of building trees is same in all, the method of split and feature selection is varied. In such condition, there is not a clear distinction in the performance of these algorithms. While one algorithm may perform better with data having more categorical features, other may perform better with data having more numerical features.

In such situation, the need for an extensive comparison of combination of encoding techniques with classification algorithms cannot be overemphasized.

1. Converting Categorical features to numeric: Encoding Techniques

# Handling Categorical Data

Encoding of categorical features or variables is required to convert them to numerical form. This is needed because most modelling algorithms require features to be all numeric. This section discusses some of the attempts made by researchers in encoding.

A transformation scheme was proposed by Barreca et. al. [1] which was based on the observed value of the target attribute. The encoding, popularly known as Target Encoding, estimates the Empirical Bayesian Probability of the target given the category level with prior being observed from the training data.

While some machine learning practitioners tend to avoid using high-cardinality features in their model just for the sake of simpler models, Moeyersoms et.al. [2] aimed at investigating if including high cardinality features actually improve the model performance. Specifically, for the case of churn prediction in an energy sector company, the authors identify three high-cardinality features- family name, bank account number and zip codes. They propose couple of techniques based on transformation of identified features to a target statistic. The first method is Weight of Evidence, which replaces churners with log of ratio of churners to non-churners, and similarly for non-churners. The second method of transformation is the Supervised ratio. This is a ratio inspired from social network perspective, given by

(1)

where and are again the number of churners and the number of non-churners for the ith value of attribute X respectively. An ‘unseen’ value for X receives an SR-score equal to the average churn rate (TC/(TC + TN)). A third encoding technique based on Perlich’s work was also used, which uses cosine distance between case vectors. On performing prediction on multiple combinations of these methods of encoding, the WOE gives the best results in terms of AUC and lift (0.1%) whereas SR performs best in terms of TPR (1%), precision (1%) and lift (1%). The PR has the highest TPR (5%) and precision (5%).

Guo et. al. [3] employ entity embeddings coupled with neural networks to map discrete variables to a multi-dimensional space where similar categories are placed closer. A neural network is used wherein an entity embedding layer learns about the intrinsic property of each category by building on top of a one-hot encoding layer. Each category of a high cardinality predictor can be mapped to a vector. The vector is similar to one hot encoded vector, but shorter in length. It’s given by:

(2)

where is a vector of length equal to number of categories in the variable xi, in which the element is only non-zero when complexity parameter α= xi. is the weight connecting the one-hot encoding layer to the embedding layer and β is the index of the embedding layer. Thus, mapped embeddings are just the weights of embedding layer and can be learned in the same way as the parameters of other neural network layers. Finally, this vector and input of continuous variables is concatenated, and merged layer is treated as any normal Neural net layer. Through experimentation, the authors show that this method can improve performance of machine learning algorithms such as kNN, random forest gradient boosting trees and neural networks as well.

In their article focused on reducing the AutoML framework to optimizing gradient boosting model, Janek et. al. [4] perform categorical feature transformation as well. They identify three methods to encode categorical variables, encoding features into integers, one-hot encoding and impact encoding (similar to target encoding). Authors evaluate different combinations of encodings, mainly based on a threshold, e.g., features with less than k levels are dummy encoded while integer or impact encoding is done for the remaining categorical features. In their framework, it is also possible to tune this threshold k together with the GBT hyperparameters. The authors show that

While designing an alternative method of building trees, a different approach is taken by Nguyen et. al. [5] with respect to categorical features. The authors propose a new feature sampling method for subspace selection, which is based on feature permutation to measure the importance of features and produce raw feature importance scores. They assess p-values for correlation between the features and response feature and group the features into high, medium and low importance features. This helps to find the cut-off between informative and uninformative features. When splitting a node, a greedy algorithm is used to identify the best split. For a categorical feature, a set of randomized values of a high-cardinal category is made, and a cut-point is selected based on maximum decrease in node impurity. This approach reduces computational complexity and can handle very high cardinality. With reduced feature space this algorithm outperforms many RF algorithms and can perform well on high dimensionality data.

Cerda et. al. [6] introduce two encoding techniques, a minhash encoder for fast approximation of string similarities, and Gamma-Poisson matrix factorization on substring counts. Minhash encoding works by grouping together similar character strings. It is based on locality-sensitive hashing and approximates jaccard coefficient between two strings. The authors claim that this method provides high levels of scalability as it is fast and efficient. Being stateless, it can work in parallel on workers in distributed systems. The drawback is, however, loss in interpretability as the strings get hashed. In order to provide interpretability, authors suggest another encoding technique, Gamma-Poisson matrix factorization. This method relies on substring representation of the string entities in the categorical variables by assuming a Poisson distribution on the n-gram counts of categories, with a Gamma prior on the activations. The authors contend that both their algorithms scale linearly with number of samples and therefore can be used in streaming settings.

Based on the literature, we can broadly classify the most popular encoding techniques in three classes. These are Classic Encoders, Contrast Encoders, and Bayesian Encoders. We discuss these in greater detail in the next section.

# Category Encoders

## Classic Encoders

### Label Encoding or Ordinal Encoding

In this method, the categories are simply mapped to an integer. The models then treat the categories as numeric which inappropriate in most cases. For example, if the categories are Male, Female and Others, these are converted to 1,2 and 3, respectively. If a new category appears later in the test data, it is generally replaced with 0 or -1. While this does not make sense in the case of feature Gender, this encoding may be useful in another feature such as Quality, having factors Excellent, Good and Bad, represented by 1, 2 and 3.

### One-Hot Encoding or Dummy Encoding

This method works by splitting the categorical feature into as many features as are the number of categories in the original feature. Each feature, thus made, represents a category and each observation can be given a binary 1 or 0 value depending on if the observation has that category or not. For instance, a column representing Gender, having values Male, Female and Others, can be split into three columns, Gender\_Male, Gender\_Female and Gender\_Others, each having 1 or 0 values depending on the observation.

In some applications, one of all the columns is omitted as it can be inferred based on other columns. In the previous example, if Gender\_Male and GenderFemale are both 0, then Gender\_Other becomes 1, meaning that Gender\_Other column can be safely removed. Though extensively used in literature, a major limitation of this method is its memory inefficiency. This is because the number of features increase as the number of categories increase. Therefore, the data becomes expensive in terms of memory usage.

### Binary Encoding

In Binary Encoding, the variable is first converted using Label Encoding. The resulting integer is converted to its binary representation. Finally, the binary string is split with each column representing a digit of the binary representation. Binary Encoding helps in tackling the problem of high cardinality as seen in One-Hot Encoding as it substantially reduces the number of variables created. For instance, if there are 16 categories of a variable, One-Hot Encoding would create 15 columns, but a Binary Encoding would take up only four columns because integers from 0 through 15 can be encoded in a four-length string.

### Hashing

Hashing is a technique to replace a string with a fixed-length vector. There could be may hash functions, each of them being common in that a hash function always returns same vector for a given string. Hash encoding therefore gives new columns just like one-hot encoding, but the number of columns can be controlled by the user. Since the length of output vector is fixed, this method solves the problem of new categories appearing in the test data, which is one of the major limitations of one-hot encoding.

## Contrast Encoders

### Helmert Encoding

In Helmert encoding [7], mean of target for a level or category is compared to mean of target for all the subsequent levels taken together. The number of resulting columns from this encoding depends on the number of pairs wherein the difference of the means is found to be statistically significant. This helps in reducing the problem of high cardinality. A variation of Helmert Encoding is reverse helmert encoding, wherein instead of comparing target means of a level and its subsequent levels, the target means of a level and its previous levels are compared.

### Difference Encoding

The mean of target variable for a level is compared to the target mean of the adjacent level. If the subsequent level is considered, it is called as forward difference encoding. If previous level is considered, it is called as backward difference encoding. These encoding could prove useful with either nominal or ordinal features.

### Sum Encoding

Each level is compared to all other levels collectively by comparing target means. This means that the target mean for level k is compared to target mean of all other k-1 levels.

## Bayesian Encoders

### Target Encoding

This method introduces a transformation scheme [1] wherein one maps each instance (value) of a high-cardinality categorical to the probability estimate of the target attribute. In a classification scenario, the numerical representation corresponds to the posterior probability of the target, conditioned by the value of the categorical attribute. In a prediction scenario, the numerical representation corresponds to the expected value of the target given the value of the categorical attribute. In order to avoid overfitting due to small number of observations in a category, smoothening of the means is also applied. Probability estimate for a category within a high cardinality categorical variable can be given by Empirical Bayesian probability, P(Y=1|X=Xi), i.e.

(3)

for all training count and i-th category row count . λ is a function which gives monotonic weight which helps when we have a small number of several categories, increasing with count from 0 to 1.

(4)

Introducing the weighting factor makes sense because when the sample size is large, we should assign more credit to the posterior probability estimate provided by first term above. However, if the sample size is small, then we replace the probability estimate with the null hypothesis given by the prior probability of the dependent attribute (i.e. mean of all Ys). With this transformation, missing values are handled by treating them as just another variable . This has advantage that if nulls have a predictive relevance, then will capture that information, otherwise it will converge towards the prior probability of target, leading to 0 effect of .

### Weight of Evidence Encoding

This method [2], having originated from credit scoring sector, can be used to transform a category of the feature into a numeric value by taking relative ratio of appearance of the category in the dataset.

(5)

TC and TN define the total number of instances of target class c1 versus target class c2; and denote the number of c1 and c2 for the ith value of attribute X.However, in case when values are zero in a particular category, 1 row is added for that category and over all ratio modified so that it is equal to TC/TN. It is recommended that the calculation of WOE be done using a separate part of the training data instead of whole data, in order to avoid overfitting.

### Leave One Out Encoding

This method was introduced to counter the effects of outliers in the training data. Similar to Target Encoding, mean of each level is calculated for the observation’s level in question, but the observation itself is left out. This makes sure that if the observation is an outlier, it does not bring bias in the calculation of category mean.

### James-Stein Encoding

The weight λ in equation 1 is a parameter that needs to be tuned explicitly. Giving more weight to a category would lead to overfitting, while giving more weight to global mean would lead to underfitting. In order to solve this problem, James-Stein encoder gives lesser weight to a category if variance in values of that category is high as compared to overall variance in target.

# References

1. Micci-Barreca, D. (2001). *A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems. ACM SIGKDD Explorations Newsletter, 3(1), 27.*
2. Moeyersoms, J., & Martens, D. (2015). *Including high-cardinality attributes in predictive models: A case study in churn prediction in the energy sector. Decision Support Systems, 72, 72–81*
3. Entity Embeddings of Categorical Variables, Cheng Guo and Felix Berkhahn
4. Automatic Gradient Boosting, Janek Thomas, Stefan Coors, Bernd Bischl
5. An efficient random forests algorithm for high dimensional data classification, Wang, Nguyen
6. Cerda, P., & Varoquaux, G. (2020). *Encoding high-cardinality string categorical variables. IEEE Transactions on Knowledge and Data Engineering, 1–1.* doi:10.1109/tkde.2020.2992529
7. R Library Contrast Coding Systems For Categorical Variables, <https://stats.idre.ucla.edu/r/library/r-library-contrast-coding-systems-for-categorical-variables/#HELMERT>
8. <http://contrib.scikit-learn.org/category_encoders/index.html>
9. <https://towardsdatascience.com/smarter-ways-to-encode-categorical-data-for-machine-learning-part-1-of-3-6dca2f71b159>
10. GBDT
    1. GBDT Overview

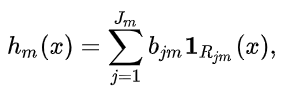
Boosting is an ensemble learning technique, in which many weak learners are trained to make predictions. Collectively, these weak learners can form a single strong learner. The weak learners are classifiers built in iterative fashion, each one learning from the mistakes of the previous learner.

All boosting algorithms are aimed at optimizing a suitable differentiable loss function. The idea was observed first by Leo Breiman [1] in 1997. As such, there have been many applications and implementations of the framework. One of the earliest and effective application was AdaBoost, or Adaptive Boosting, formulated by Freund and Schapire [2], wherein the output of each weak learner is weighted and then combined to form the final output. This algorithm is adaptive in the sense that each subsequent learner learns only from the mistaken observations of the previous learner. It was noted that with AdaBoost, overfitting is contained even when the iterations or number of weak learners increase.

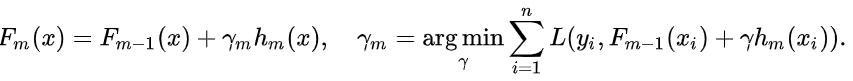
In addition to AdaBoost, several variations in the original idea introduced more algorithms. When logistic function is used for measuring cost of each iteration, the algorithm is known as LogitBoost [3]. The authors of this framework proved statistically that weak learners are additive on a logistic scale. Another boosting algorithm, BrownBoost [4] is specialized in datasets with noisy observations. This algorithm assumes that if learners are repeatedly misclassifying certain observations, then such observations are outliers and it removes such samples from the data. Domingo et. al. [5] noted the noise sensitivity of AdaBoost, and presented a new boosting algorithm called MAdaBoost, by modifying the weighting system of AdaBoost. The main change is simple, MAdaBoost simply places an upper bound on the weights that can be assigned to any datapoint. This proved to be more robust to class-label noise.

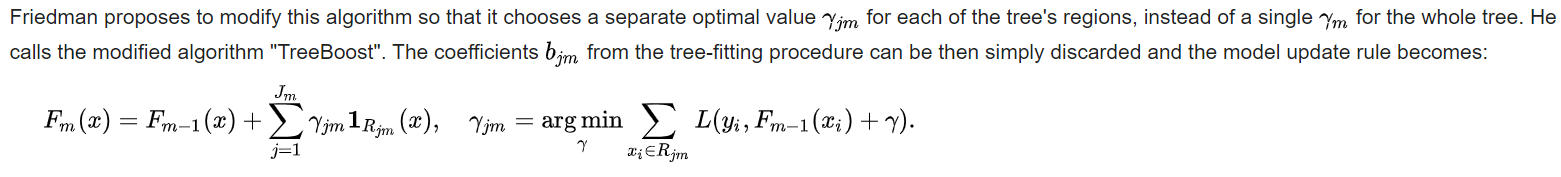
In addition to these frameworks and improvements, there was rising interest in another class of boosting algorithms, called gradient boosting algorithms. Freidman [6,7] proposed modifications to gradient boosted methods to use decision trees and improve on the quality of fit. All the recent advancements in boosting algorithms build on gradient boosting method.

Generic gradient boosting at the m-th step would fit a decision tree hm(x) to pseudo-residuals. Let Jm be the number of its leaves. The tree partitions the input space into Jm disjoint regions R1m,,…,RJ(m)m and predicts a constant value in each region. Using the indicator notation, the output of hm(x) for input x can be written as the sum:



Where bjm are multiplied by some value gamma(m), chosen using line search so as to minimize the loss function, and the model is updated as follows:





In many GBDTs building next tree comprises two steps: choosing the tree structure and setting values in leaves after the tree structure is fixed. To choose the best tree structure, the algorithm enumerates through different splits, builds trees with these splits, sets values in the obtained leafs, scores the trees and selects the best split. Leaf values in both phases are calculated as approximations for gradients or for Newton steps.

Though the underlying principle is same, there have been many variants of GBDTs since its inception. Feng et. al. [8] proposed Soft Gradient Boosting Machine, which wires multiple differentiable base learners together for joint optimization with linear speed-up of the traditional gradient boosting trees training. The authors claimed much higher time efficiency with better accuracy.

In next section, the three state-of-the-art GBDT implementations- XGBoost, LightGBM and CatBoost are explained in great detail, along with the innovations and contributions.

1. *Breiman, L. (June 1997).*[*"Arcing The Edge"*](https://statistics.berkeley.edu/sites/default/files/tech-reports/486.pdf)*. Technical Report 486. Statistics Department, University of California, Berkeley.*
2. Robert E. Schapire. 1999. A brief introduction to boosting. In Proceedings of the 16th international joint conference on Artificial intelligence - Volume 2 (IJCAI'99). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1401–1406.
3. Friedman, Jerome; Hastie, Trevor; Tibshirani, Robert (2000). "Additive logistic regression: a statistical view of boosting". Annals of Statistics. 28 (2): 337–407. CiteSeerX 10.1.1.51.9525. doi:10.1214/aos/1016218223
4. Freund, Y. An Adaptive Version of the Boost by Majority Algorithm. *Machine Learning* **43,**293–318 (2001). https://doi.org/10.1023/A:1010852229904
5. *Domingo, Carlos & Watanabe, Osamu. (2000). MadaBoost: A Modification of AdaBoost. Proceedings of the Thirteenth Annual Conference on Computational Learning Theory.*
6. *Friedman, J. H. (February 1999).*[*"Greedy Function Approximation: A Gradient Boosting Machine"*](https://statweb.stanford.edu/~jhf/ftp/trebst.pdf)*.*
7. *Friedman, J. H. (March 1999).*[*"Stochastic Gradient Boosting"*](https://statweb.stanford.edu/~jhf/ftp/stobst.pdf)*.*
8. Ji Feng, Yi-Xuan Xu, Yuan Jiang, Zhi-Hua Zhou, Soft Gradient Boosting Machine,  [arXiv:2006.04059](https://arxiv.org/abs/2006.04059)
   1. State-of-the-art in GBDT

# Gradient Boosting Decision Trees

Started as a research project in 2014, XGBoost [1], short for eXtreme Gradient Boosting, became famous in 2016 when Chen et. al. successfully used it in Kaggle competitions and formally published it. The main contribution of this implementation of GBDT algorithm is an improved algorithm for finding split value for a feature and distributed computing provision. XGBoost offers two methods for finding the best split, one is exact and the other is approximate.

The exact greedy algorithm is the slower one of the two, as it iterates over all the features and sorts the data according to the feature value before the gradient statistics are calculated for an instance. A split is then chosen so that it maximises the information gain. Analysing every possible split makes this algorithm highly precise; however, this also leads to computational overload and slower processing.

On the other hand, approximate algorithm uses the bins of histograms of the feature values instead of sorting to propose possible splits. The best split is then decided based on aggregate statistics of the proposed splits. If the proposal of splits is given only in the beginning during tree construction, the algorithm is faster and the authors call it global variant, wherein same proposals are used for splitting at all levels. The local variant re-proposes the splits after each split. This could potentially lead to improved performance when building deeper trees.

Another performance improvement in XGBoost comes by means of handling missing values in sparse data. The ‘sparsity-aware algorithm’, which gives missing value instances a default direction, has been reported to perform better than other algorithms which are generally designed for dense data as opposed to sparse data.

Finally, an effective cache-aware block structure helps to scale the algorithm for out-of-core tree learning. This facilitates distributed computing and therefore lets the user build trees for data with billions of instances.

LightGBM [2] is another implementation of GBDT which builds on the innovations proposed in XGBoost and other algorithms, while addressing the limitations of these. The authors argue that the existing algorithms were inefficient and slow in big data applications as their computation time was proportional to the number of instances and number of features because they scan all data instances in order to find the best split. The authors introduce two novel ideas for optimisation, namely, Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB).

GOSS handles the size of data vertically; it is based on the premise that the instances with small gradient can be removed from the data as they already have small training error. In order to compensate for the change in the distribution of data, such low gradient instances are randomly sampled and then a constant multiplier is used when computing information gain with them. This helps in giving higher importance to instances with large gradient.

In order to tackle the problem of large number of features, EFB reduces feature count by bundling them together. The authors claim that many features are mutually exclusive in that they generally do not take non-zero values simultaneously; as such, these features can be bundled together. The features to be bundled together are selected such that the histogram of bundled feature is same as that of the individual features.

CatBoost [3], which derives name from Categorical Boosting, is the most recent in popular implementations of the GBDT algorithm, proposed by Yandex group. The authors contend that the limitations of all existing GBDT implementations is due to a special form of target leakage called prediction shift- a phenomenon caused by the models getting biased towards the training examples because it is based on the targets of all the training examples. This problem affects the overall model as well as categorical features encoded using target statistics methods and is tackled in both cases by ordering of training samples.

Ordering principle is inspired from online learning algorithms, wherein the training samples are obtained sequentially in time. Similarly, CatBoost creates artificial time by creating random permutations of the training samples. Different permutations are required for different steps of gradient boosting to reduce variance. This prevents target leakage during encoding of categorial features because the values of target statistic only rely on the observed history.

The proposed ordered boosting works as follows. The algorithm takes one random permutation of the samples and builds trees such that each tree is built only on as many number of samples as the index number of the that tree. The residuals for next observation is calculated using the previous tree. This makes sure that the residuals are calculated using a model which was not trained on the current observation.

In addition, CatBoost uses oblivious trees as base predictors. This is done in order to simplify the complex models because oblivious trees are simpler, balanced and less prone to overfitting.

# Handling Categorical Features

While CatBoost and LightGBM support categorical features, XGBoost requires pre-processing of the data as it provides support for only numerical features. Therefore, the techniques such as one-hot encoding, target encoding and others are necessary.

LightGBM [4] requires the categorical features to be encoded as integers, and such features need to be specified explicitly using parameter *categorical\_feature*. The categories are split into two subsets. First, the histogram of categorical features are sorted according to its accumulated values (sum\_gradient/sum\_hessian) and then the best split on the sorted histogram is obtained.

In order to handle categorical features, CatBoost [5] provides the option of both one-hot encoding as well as target statistics methods. One-hot encoding is generally used when number of categories in a feature is relatively small. The hyperparameter *one\_hot\_max\_size* can be used to set a value threshold for the number of categories in a feature, above which target statistic method is used. For encoding using target statistics, CatBoost first generates random permutations of the training data in order to facilitate ‘online learning’. For each permutation , a subset of data ={} is taken in the training phase, which ensures that the target statistic is calculated only with the observed history of data. In the testing phase all data is taken. The category value is then calculated as

where p is a prior value and a is the weight of the prior. This is required to minimize the effect of noise from categories which have lower frequency. For regression tasks, the standard technique for calculating prior is to take the average label value in the dataset. For classification task a prior is usually an a priori probability of encountering a positive class.

After first split in a tree, CatBoost combines all categorical features of dataset with the existing features in the tree. This ensures that any strong combination of features is acknowledged. Therefore, a general solution of ordered boosting with ordered Target Statistic is reached, which solves the problem of prediction shift in other gradient boosting tree methods.

# References

[1] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD ’16). Association for Computing Machinery, New York, NY, USA, 785–794. DOI:https://doi.org/10.1145/2939672.2939785

[2] LightGBM: Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: a highly efficient gradient boosting decision tree. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS’17). Curran Associates Inc., Red Hook, NY, USA, 3149–3157.

[3] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. 2018. CatBoost: unbiased boosting with categorical features. In Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS’18). Curran Associates Inc., Red Hook, NY, USA, 6639–6649.

[4] Optimal Split for Categorical Features, <https://lightgbm.readthedocs.io/en/latest/Features.html#optimal-split-for-categorical-features>

[5] Transforming categorical features to numerical features, <https://catboost.ai/docs/concepts/algorithm-main-stages_cat-to-numberic.html> , Last accessed: 20-6-2020

1. **Methodology**
   1. **Datasets**

Table xxx shows the datasets used and the features of each dataset. The datasets were obtained from trusted sources such as UCI ML repository, Kaggle datasets and data.world website. Care was taken so that the chosen datasets have diverse characteristics. Therefore, the observations vary from 1,473 in the Contraceptive dataset, to almost a million in the Physician dataset and features range from 5 in Land process dataset through 32 in the kick dataset. There is a mix of categorical and numeric features, with two datasets being purely categoric. Highest cardinality represents maximum number of categories present in any of the categorical feature of the dataset. Datasets were selected to come from diverse backgrounds and of different number of classes so that the results of the experiment are not biased.

Adult dataset has demographic information of individuals, the task is to predict whether an individual earns more or less than $50,000 a year. Amazon dataset carries information about an employee’s role at the firm, the task is to predict if the employee should be granted the access to a resource or not. These auto-access models seek to minimize the human involvement required to grant or revoke employee access. Bank marketing dataset provides client information for a Portuguese banking institution. The classification goal is to predict if the client will subscribe a term deposit or not. Contraceptive dataset is a subset of the 1987 National Indonesia Contraceptive Prevalence Survey. The samples are married women who were either not pregnant or do not know if they were at the time of interview. The goal is to predict the current contraceptive method choice (no use, long-term methods, or short-term methods) of a woman based on her demographic and socio-economic characteristics. Kick dataset contains details of vehicles being auctioned. The challenge is to predict if the car purchased at the auction is a bad buy or a good buy. Land prices dataset sources information regarding the value per acre of farmland in each state/region in the United States since 1997, as published by National Agricultural Statics Service (NASS). The land prices have been grouped into 10 buckets, which is to be predicted. Physician dataset identifies prescription drugs providers by their National Provider Identifier and summarizes for each prescriber the total number of prescriptions that were dispensed, which include original prescriptions and any refills, and the total drug cost, along with some identifiable information of the physicians. Only this physician information has been taken from the dataset to predict the number of patients visiting the physician. Poker hand dataset consists of examples of five playing cards drawn from a standard deck of 52. Each card is described using two attributes (suit and rank), for a total of 10 predictive attributes.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Observations** | **Features** | **Categorical** | **Numerical** | **Highest**  **Cardinality** | **Classes** |
| **Adult** | 32,561 | 14 | 8 | 6 | 42 | 2 |
| **Amazon** | 32,769 | 9 | 9 | - | 7,518 | 2 |
| **Bank Marketing** | 41,188 | 20 | 10 | 10 | 12 | 2 |
| **Contraceptive** | 1,473 | 9 | 4 | 5 | 34 | 3 |
| **Kick** | 72,983 | 32 | 15 | 17 | 1,063 | 2 |
| **Land Prices** | 3,458 | 5 | 4 | 1 | 60 | 10 |
| **Physician** | 969,617 | 8 | 8 | - | 19,574 | 4 |
| **Poker Hand** | 25,010 | 10 | 5 | 5 | 4 | 10 |

* 1. **Experimental Design**

Along with the three algorithms of interest, namely XGBoost, LightGBM and CatBoost, Random Forest has also been considered, as this is one of the popular, interpretable and reliable method of getting a baseline result. Therefore, we have four algorithms to test their classification performance.

In order to convert the categorical features to numeric, we use four methods. First is the widely used one-hot encoding, where each category of a categorical feature is represented by a binary column. Second is target encoding, where each category is replaced by the probability estimate of the target label for that category. These two methods have been explained in greater detail in earlier section. The third method is a combination of both methods- if the feature cardinality is less than 10, then the feature is transformed using one-hot encoding, while if the cardinality is higher than or equal to 10, then the feature category is replaced using target encoding. We call this method Mixed Encoding. This method helps in keeping check on horizontal explosion of the dataset in case the cardinality is very high. The fourth encoding technique is the one provided by the algorithms themselves. Only LightGBM and CatBoost have the capability to treat the categorical features natively. We call this encoding technique as Native Encoding.

Thus, we have 8 datasets, 4 algorithms and 4 categorical encoding techniques. Considering Random Forest and XGBoost do not have native encoding functionality, we have a total of 112 combinations to test on.

For each combination, the pipeline involves six steps as shown in figure 1. We first read the data in Python using Pandas package. We encode the categorical features using one of the encoding techniques and split it into train and test data with 80:20 ratio of observations. Next, we apply Scikit-learn’s Research method, which takes in a dictionary of permissible values of the hyperparameters and trains the train data. This method helps in finding the best combination of hyperparameters and this best model is then evaluated on the test data.

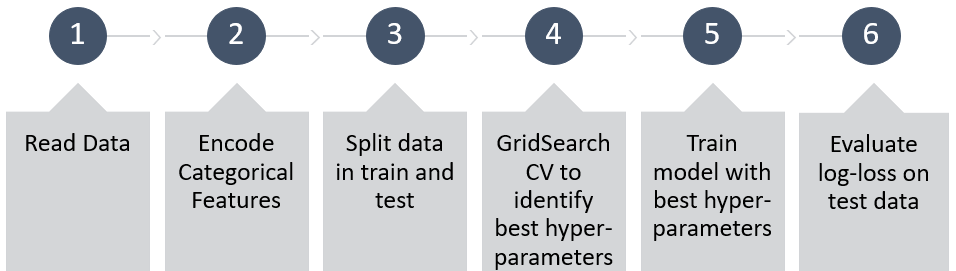


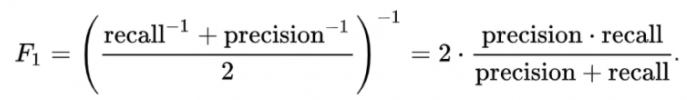
Figure : Steps showing experimental setup

* 1. **Evaluation Metrics**

Building a model is only half the work done. There needs to be an evaluation metric to measure the performance of the model. In the industry, different kinds of metrics are used to evaluate models. The choice of metric is a function of the model type and objective. For classification tasks, confusion matrix provides several metrics to select from, such as accuracy, precision and recall.

While classification accuracy is the most widely used metric, it is arguably the most misleading one. Accuracy is defined as the ratio of correct predictions to the total predictions. In reality, accuracy does not give information about incorrect predictions being made. Specifically in two cases, accuracy metric fails. First, when there are more than two classes. In such scenario, the metric does not give information about the individual accuracy for each class. It is possible the model ignore one or more classes. Second, when the data has imbalanced classes. For instance, if a binary classification dataset has class imbalance ratio of 90:10, then a model with 90% classification accuracy is a non-informative one.

In order to tackle these limitations, f1 score is another metric, which is defined as the harmonic mean of precision and recall. Precision is the proportion of positive cases that were correctly identified, while recall is the proportion of actual positive cases which are correctly identified. Improving F1 score helps in increasing precision and recall a t the same time. Taking a harmonic mean makes sense because as compared to arithmetic mean, harmonic mean punishes extreme values more.



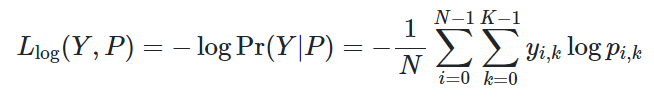
Despite being better than accuracy as a metric, F1 score still suffers a drawback in classification models where the class prediction is based on class probability. Consider two models wherein for a particular observation, the first model predicts a class with probability 0.6, while the other predicts class with probability 0.9. F1 score will treat each of these models as equal, because both predict same class for this observation. It does not take into account the certainty of prediction.

In order to account for the class probability, log-loss can be used. Log-loss is negative average of the log of corrected predicted probabilities for each instance.

For binary classification with a true label y∈{0,1} and a probability estimate p=Pr(y=1), the log loss per sample is the negative log-likelihood of the classifier given the true label:



This extends to the multiclass case as follows. Let the true labels for a set of samples be encoded as a 1-of-K binary indicator matrix Y, i.e., yi,k=1 if sample i has label k taken from a set of K labels. Let P be a matrix of probability estimates, with pi,k=Pr(ti,k=1). Then the log loss of the whole set is



In this thesis, a python API provided by Sklearn library has been used to calculate the log loss, which takes as input the actual labels and estimated probability of each class. Baseline for each dataset would be different as it is a function of the number of classes as well as their class ratio. The same has been calculated using 10,000 random samples generated in same ratio as the classes for each dataset.

1. **Results**

For each dataset, 14 combinations of encoding technique and classification algorithm was run. It was observed that at least one combination always performed better than random guess baseline for all datasets. Figure 2 shows that highest predictive power was gained in the Land prices dataset, where log-loss reduced by 82.7%, while least in the Physician dataset, where log-loss reduced by only 6.4%. However, it should be noted that these predictive powers can not be compared among datasets as log-loss is a function of number of classes and the class ratio of the dataset.

Figure : Comparison of best model for each dataset and random guessing log-losses

Figures 3,4,5,6 show performance of each of the four encoding techniques across datasets for all the four algorithms. It can be noted that one-hot encoding performs better than random guess baseline in all combinations. This is on contrary to target encoding, where XGBoost performs worse than baseline for Amazon, Kick and Physician datasets. However, it should be noted that otherwise, performance of Target encoding is much better than one-hot encoding. Results for Mixed Encoding are comparable to the Target Encoding. Native encodings perform reasonably well as compared to other three encoding techniques. Not only do they always perform much better than the baseline, they do so with a wide margin.

From this comparison of encoding techniques, it can be inferred the one-hot encoding, though reliable, performs worse as compared to other encoding techniques. On the other hand, native encoding performs reliably as well gives good performance. Target Encoding is relatively not as reliable. However, with more tuning and cross-validation, target encoding can potentially outperform native encodings techniques.

Figure : Performance of One-hot Encoding across datasets and algorithms

Figure : Performance of Target Encoding across datasets and algorithms

Figure : Performance of Mixed Encoding across datasets and algorithms

Figure : Performance of Native Encoding across datasets and algorithms

Next, we compare the performances of the gradient boosting decision tree algorithms. Figures 7-10 compare each of the algorithms against encoding techniques. It can be seen in figure 7 that while Random Forest always give better results than the random guessing baseline, it should be noted the performance is only marginally better in most datasets. In XGBoost, the results are staggered. While the algorithm always performs consistently with one-hot encoding, the results are unreliable with target encoding and native encoding. For LightGBM, the results are relatively reliable but it performs worse as compared to CatBoost, which delivers the best performance overall.

This comparison establishes superiority of CatBoost over other algorithms. In addition, it reveals that XGBoost performs best with one-hot encoding technique. This is possibly because it is designed to expertly handle only numeric features with histogram-based splitting. On the other hand, CatBoost relies on ordered boosting and handles categorical features in a similar fashion as mixed encoding. Coupling of these two features may account for the performance lift.

Figure : Performance of Random Forest across datasets and encodings

Figure : Performance of XGBoost across datasets and encodings

Figure : Performance of LightGBM across datasets and encodings

Figure : Performance of CatBoost across datasets and encodings

Further in order to investigate the performance on a deeper level, we dive into the Amazon dataset and Physician dataset. All the features of these two datasets are categorical, as such, it should be interesting to see how the combination of algorithms perform on them. have We compare and contrast the feature importance in these two datasets across target encoding and native encoding.

Figure 11 shows the relative performance of the 14 models ran on this dataset. One-hot encoding performs much better than target and mixed encodings for XGBoost, which is in agreement with our previous observation. On the other hand, native encoding performs better than all other encoding techniques in LightGBM and CatBoost. In LightGBM, native encoding beats the next-best one-hot by 6%, while in CatBoost, the performance improvement of native encoding over one-hot is close to 11%. However, one-hot encoding with XGBoost beats one-hot with LightGBM by a 5.6% as well. On contrasting the feature importance of the models with target encoding versus native encodings from figure 12 and 13, it can be easily seen that target encoding only gives importance to two of the features. Interestingly, native encoding extracts information from all the features. With this information, better performances of native encodings make sense.

Figure : Model performances for Amazon dataset

Figure : Feature importances with Target Encoding in Amazon dataset

Figure : Feature importances with Native encoding in Amazon dataset

For Physician dataset, figure 14 shows relative performance of the 14 models. While most of the models perform only marginally better than baseline, target encoding in LightGBM and CatBoost outperform other combinations. In fact, even Random Forest with target encoding beats LightGBM with target encoding by 1.4%, while CatBoost beats RF by only 0.6%. Before comparing the performances of algorithms on this dataset with Amazon dataset, it should be noted that Physician dataset has 4 classes as opposed to only two classes in Adult, and has much higher cardinality of 19,574 as compared to 7,518 of Adult dataset. These two characteristics make prediction task in Physician dataset much more difficult as compared to that in the Adult dataset.

Looking at figures 15 and 16 gives an indication of how the algorithms deal with the features in Physician dataset. While those in target encoding take full advantage of the features, as is evident from high importance scores, the same is not the case in native encoding. It can be seen that only two features are being used more in native encoding.

While comparing the feature importance in Amazon versus Physician dataset, we see quite reversal in the behavior of target encoding and native encoding. This could possible be because Amazon dataset is a binary classification task while Physician is four-class classification. This means that the target encoding increases the number of features In the Physician dataset. For each class of the target, we have one new feature. This happens for every categorical feature in the Physician dataset. Thus we have more information in a target encoded dataset of multiclass dataset, as compared to native encoded dataset.

This behavior of the multiclass dataset insinuates that the category-handling GBDT algorithms LightGBM and CatBoost are optimized for binary classification tasks.

Figure : Model performances for Physician dataset

Figure : Feature importances with Target Encoding in Physician dataset

Figure : Feature importances with Native encoding in Physician dataset

Finally, we compare the performance of all the 112 models simultaneously. With 8 datasets, 4 algorithms and 4 encoding techniques, we have a three-dimensional matrix to visualize. In order to simplify, we look at relative performance of the models as shown in figure 17. The datasets are stacked vertically, while algorithms are stacked horizontally. Each cell gives name of the best performing encoding technique for a given combination of the dataset and GBDT algorithm. Once we have this information for all four GBDT algorithms for one dataset, we compare the four best combinations. Therefore, the first cell in figure 17 denotes that for Random Forest in Adult dataset, Target encoding had the lowest log-loss among the four encoding techniques. Similarly, for XGBoost in Adult dataset, Target encoding had the lowest log-loss among the four encoding techniques. Next we look at colors in cells of each row. Looking at the first row, we see that Target encoding with Random Forest performed the worst and had highest log-loss, while Native Encoding with CatBoost performed the best with least log-loss across the row.

Looking at the results in figure 17 collectively, it is easily seen that CatBoost performs the best for all datasets with one of the four encoding techniques. In fact, it never performs good with one-hot encoding. This is in contrast to XGBoost, which frequently shows low log-losses with one-hot encoding.

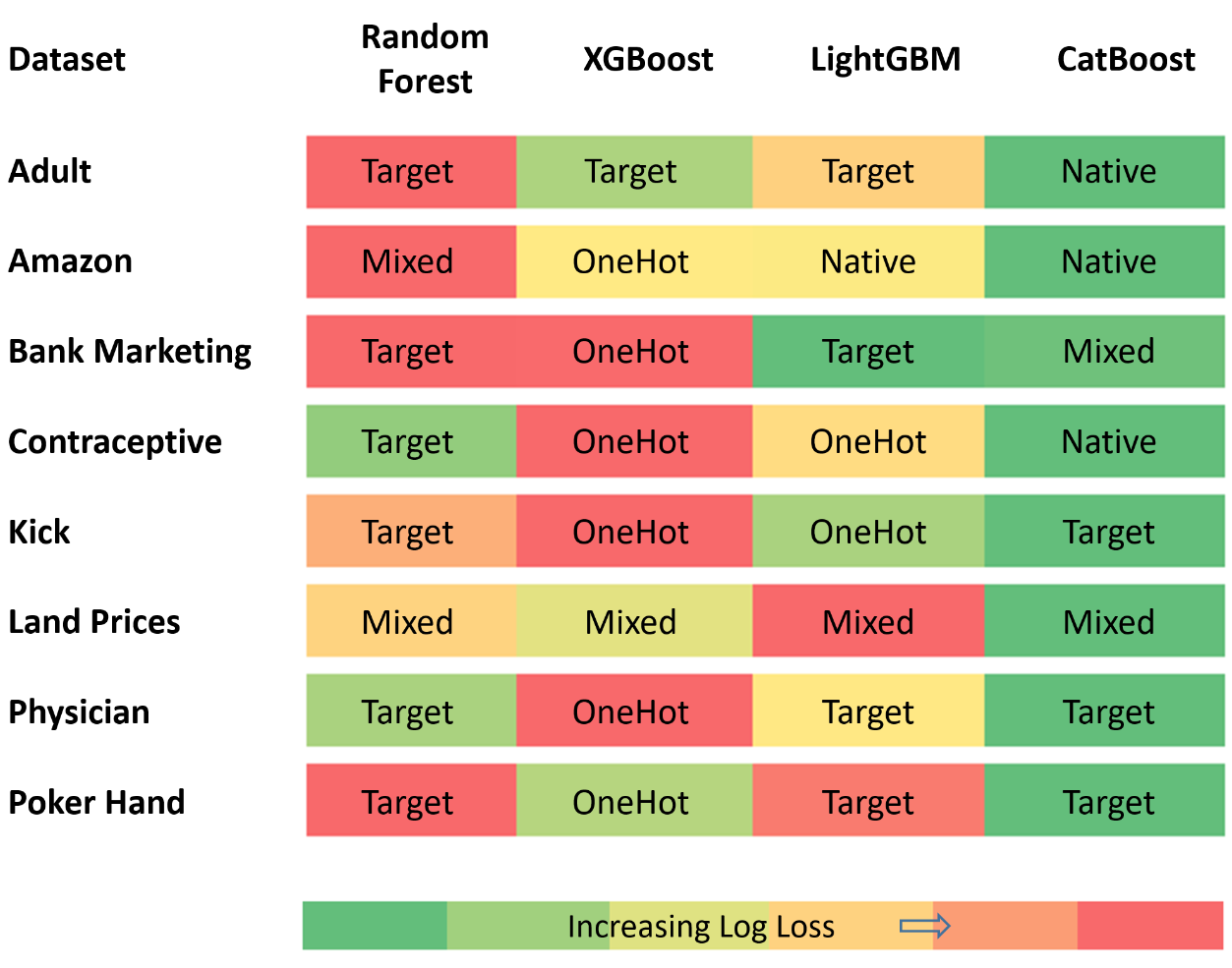
****

Figure : A cross-tab showing best models. Text in each box shows best performing categorical encoding technique for that algorithm. Color shows relative log-loss across algorithms for the dataset.

1. **Security and Privacy Considerations**

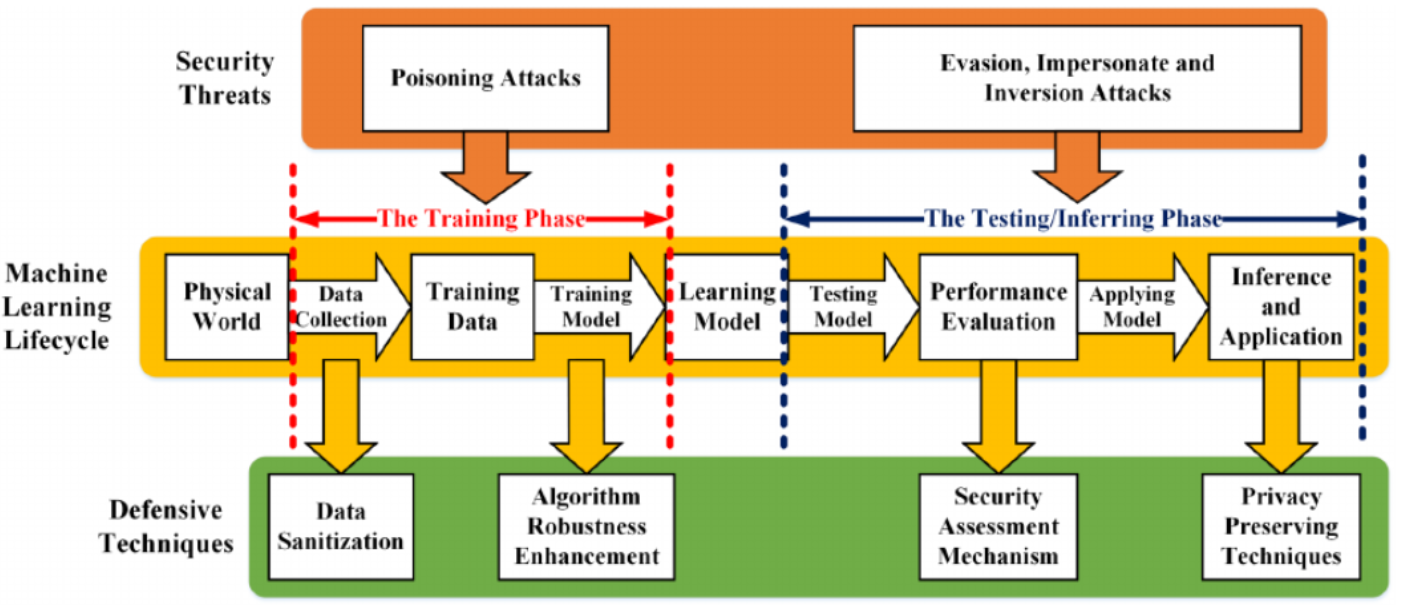


Figure : Threats and defenses during ML research [1]

* 1. **When Data is problem for the Model: Adversarial Data**

During the data collection and cleaning phase of the project, issues related to data security can arise at multiple points. This is because I intend to obtain data from open-source repositories such as UCI ML repository, Kaggle datasets, data.world and other similar websites, wherein some users publish their own datasets as well.

Here, the dataset is susceptible to Poisoning attacks. A Poisoning attack disrupts the integrity and performance of a model by injecting adversarial samples in the dataset. An Adversarial attack is when data points are intentionally added to the dataset so that the performance of the classifier is reduced, and misclassification rate is increased. This is done by including false labels in the dataset. There are numerous security issues that arise as a result of this attack. A self-driving car using the algorithm designed as a result of this dissertation, for example, could identify a stop sign as something else, leading to accidents. Though I will make every effort to download the data only from authentic websites and dataset publishers, it is possible that even these websites have doctored data uploaded by individuals with malicious intent.

Facebook recently published their work [2] wherein they tag the training images with imperceptible changes, much in the same way as a watermark. They call this “radioactive dataset”. Numerous other successful works exist in literature. I can potentially make use of such techniques in order to improve the security of my datasets. This ensuring the purity of training data is called as sanitization of data. In order to improve the robustness of my model, I can utilise Bootstrap Aggregating and Random Subspace methods, which improve the stability of model, while making them more reliable.

* 1. **When Data is problem for the System: Malware**

I will download the data on the SCSS server and interact with it through python codes on my Personal Laptop Computer using command line interface (SSH) and open-source FTP clients such as FileZilla. Evidently, there are multiple nodes where the data is vulnerable to security issues (Figure 2). It is technically possible to inadvertently download malware along with the dataset.

* **Virus**: There could be virus taking advantage of the immense computing resources of the SCSS. Virus are programs that multiply and consume system resources leading to Denial of Service attacks. If the virus does not replicate itself outside the resources allocated to each user, whole of the system would not go down because access restrictions are put in place for SCSS resources. If accidentally downloaded on my PC, a malware could lead to unexpected lag in processing and delay in meeting deadlines.
* **Ransomware**: In case of malware being a ransomware, whole system could get affected. Ransomwares are programs that restrict access to files stored on the system. Such an attack could potentially lead to lockout of all the users from using SCSS resources. When such situations arise on my PC, I find doing a factory reset as the quickest and easiest method to revive the PC, which of course may lead to loss of data and work, in case it has not been backed up.
* **Trojans**: Similarly, Trojan horses are computer programs that are installed without the knowledge of the user and may provide a backdoor control of the system to the attacker. The system can then be potentially used to perform illegal activity over the internet. If attacked with a trojan, this may have legal implications for either the SCSS or me or both.
* **Spyware and Keylogger**: Spyware and Keylogger are malwares that monitor the activities and keystrokes of a user. If installed, such systems can steal my passwords through which I access SCSS resources and personal sites as well. My home router could be hacked to gain access to the data. A combination of a Keylogger and trojan horse attack could potentially be a nightmare.

### **The Solution**

Thankfully, there are checks in place at various levels within the SCSS server system. Available storage and computing resources are both limited per user. All these resources are secured with passwords. The slurm-master job management system allows only one job to be scheduled by a user so that memory hogging does not take place. Windows 10 security in my personal computer is enough to restrict most attacks. I have my home wi-fi hidden and masked so that any amateur hacker may not gain unauthorized access.

* 1. **In Production**

Tree-based models, including the one I propose in my dissertation, are deployed in a wide range of applications. The vulnerabilities of models may be exploited when used in practice to classify. Attackers can generate synthetic samples to evade detection and impersonate victims to obtain unauthorized access.

* **Evasion**: Evasion attack is when an attacker aims to compromise the machine learning in information security. For example, in a spam detection application, an adversary may learn of the model’s vulnerability and produce fake data to fool the algorithm, and successfully send a spam to a victim. Similarly, an attacker may target PDF malware detection. Gradient algorithms, on which most boosting algorithms including the ones I have selected for my work are based, have been shown to be vulnerable to such evasion attacks [1].
* **Impersonation**: Impersonate attacks are when the attacker generates adversarial data keeping in mind the model’s limitations and forces the model to make a misclassification. Doing so, the attacker can potentially gain access to a victim’s private data or a practical access control system. Generative Adversarial Networks (GANs) have been highly successful in manipulation of image and sound data. Though my dissertation is not connected to GANs, in practical applications there could be a combination of the proposed model in conjunction with GANs, that can potentially be exploited.
* **Inversion**: In applications where machine learning models have a front-end API, an attacker could access the model and reverse engineer it to obtain information about the training data. This can have serious implications in patient health applications, for example, because medical healthcare data is considered Protected Health Information.

### **The Solution**

Contrary to the defensive techniques employed in training phase, in order to avoid the above security risks in testing phase, countermeasures are focussed on improving the robustness of models. In my work, I am going to going to build a model which can predict the best tree-based method for a particular application. Therefore, I cannot guarantee the security of the proposed tree-based algorithm. However, I can work to make sure that the algorithm I use to suggest the model, is robust and cannot be exploited for reverse engineering for instance.

**Conclusion**

I presented some security and privacy risks that can arise out of this dissertation. Although the situation may look grim considering so many concerns in the above discussion, practically the risk is low. This is because the datasets I am using in this research are all open-source and it seems unlikely that attackers would want to invest their time and effort in designing such mechanisms. These days, big firms such as Google and Facebook organise open hacking contests and reward those who can spot security flaws in their frameworks. I believe hackers would rather spend their time where there is a higher probability for reward. The SCSS server system is robust and scalable enough to thwart any small-scale attack. Security of my personal Windows 10 system is enough and strong with the regular updates from windows. Despite low risks for my research, the possibility of exploitation when being used in practical applications cannot be negated.

[1] Liu, Qiang & Li, Pan & Zhao, Wentao & Cai, Wei & Yu, Shui. (2018). A Survey on Security Threats and Defensive Techniques of Machine Learning: A Data Driven View. IEEE Access. 6. 12103-12117. 10.1109/ACCESS.2018.2805680.

[2] Sablayrolles, Alexandre & Douze, Matthijs & Schmid, Cordelia & Jégou, Hervé. (2020). Radioactive data: tracing through training. ([link](https://arxiv.org/pdf/2002.00937.pdf))

1. **Conclusion**

In this thesis, the performance of four gradient boosting decision trees algorithms, namely Random Forest, XGBoost, LightGBM and CatBoost, was compared for classification on eight datasets from varying domains, having a mixture of numerical and high-cardinality categorical features. The algorithms were compared for four different categorical encoding techniques, namely, one-hot, target, mixed and native encodings.

It was observed that CatBoost generally performed the best, with LightGBM coming in a close second position. Among XGBoost and Random Forest, the performance depends on the dataset and the type of encoding technique used and, as such, has no clear pattern. However, there is no clear winner for the encoding technique to be used. The choice of encoding technique really depends on the number of classes in the target label and the GBDT algorithm being used. The best performing combinations were target or native encodings with CatBoost.

Interestingly, CatBoost never performed well with one-hot encoding. On the contrary, XGBoost generally performed the best with one-hot encoding. Also, CatBoost performance was best with native encoding on binary classification tasks, and with target encoding on multi-class classification tasks. The performance of LightGBM did not show a pattern with any encoding techniques.

These observations led to an inference that GBDT algorithm designed to handle categorical features natively, CatBoost, has been optimised for binary classification tasks, as such, it does not perform relatively well on multi-class datasets. However, with target encoding, these algorithms can beat other GBDTs.

1. **Future Work**

All the models and combinations of algorithms in this work have been evaluated using only the log-loss metric. However, there may be applications where other metrics or loss functions need to be optimised. There could be a comparison of algorithms across metrics to be optimised.

Also, the performances have been reported irrespective of the training time it took for each combination. The models can be compared for training speed. There could be interesting patterns to be seen when binary classification tasks are compared with multi-class classification tasks.

Inclusion of performance metric and training times could make such study a much more exhaustive 5-dimensional comparison, instead of a 3-dimensional one, as in the present work.

1. **Bibliography**