Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

A Comparison of Gradient Boosting Decision Tree Algorithms on High Cardinality Data

Nishant Mohan

A Dissertation

Presented to the University of Dublin, Trinity College

in partial fulfilment of the requirements for the degree of

Master of Science in Computer Science (Data Science)

Supervisor: Dr. Mimi Zhang

September 2020

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Nishant Mohan

September 4, 2020

Acknowledgments

I would like to express my sincere gratitude to my advisor, Dr. Mimi Zhang, for her valuable and constructive suggestions throughout the course of this research work. I am also grateful to Computer Science and Statistics Department of Trinity College Dublin to provide me with infrastructure and good environment to work on my thesis.

I would also like to thank my second reader Dr. Andrew Butterfield for his time and suggestions.

I must express my profound gratitude to my parents for providing me with unfailing support and motivation throughout my year of study.

Nishant Mohan

University of Dublin, Trinity College
September 2020

A Comparison of Gradient Boosting Decision Tree Algorithms on High Cardinality Data

Nishant Mohan, Master of Science in Computer Science
University of Dublin, Trinity College, 2020

Supervisor: Dr. Mimi Zhang

Recent advancements in classification tasks, such as using decision trees as base learners in boosting algorithms coupled with gradient descent method, have significantly improved the performance of gradient boosted decision tree (GBDT) algorithms. Currently, there are several classification algorithms that deliver high accuracy. Out of these, CatBoost, LightGBM and XGBoost are the most popular and reliable ones. In addition to new classification algorithms, diverse methods of encoding have made it possible to encode categorical information to numeric information efficiently in many ways. Improvements in methods such as target encoding have made the performance of existing classification algorithms even better.

Since the gradient decision boosted decision tree algorithms are fairly recent, their performance with various encoding methods have not been compared across a large number of data sets. This work compares the performance of said classification algorithms across data sets using different encoding techniques. Each of the GBDT algorithms, namely, XGBoost, LightGBM, and CatBoost, are tested on eight diverse data sets using four categorical encoding techniques each. This gives a three-dimensional view of the performance and reliability of the GBDT algorithms.

The comparison of the GBDTs shows that CatBoost significantly performed better than XGBoost and LightGBM on all the data sets, there is no one categorical encoding technique that performed the best on all data sets.

Summary

This work gives an overview of the gradient boosting decision tree algorithms, including their evolution and concepts involved. It also presents an in-depth classification of various categorical encoding methods.

The research compares the performance of state-of-the-art gradient boosting decision tree algorithms on various categorical encoding techniques. The selected algorithms are Random Forest, XGBoost, LightGBM, and CatBoost. These are compared across one-hot encoding, target encoding, mixed encoding, and native encoding. All combinations of these are trained on eight data sets, namely, adult, amazon, contraceptive, bank marketing, kick, land prices, physician, and poker hand. The data sets are sourced from varying background, and having a diverse number of observations, features, and number of classes, for a fair comparison.

The results prove that with minimum log loss, CatBoost outperforms all other algorithms. However, there is not one encoding technique that performs better than others. While native encoding performs well on binary classification, target encoding did better with multi class data sets.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

With the 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 the internet of things. The International Data Corporation predicts that by 2025, total digital data created would be a staggering 163 zettabytes, all attributing to a growing number of devices and internet [1]. In fact, data has been aptly called the new oil of the 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 analyze trends in customer behavior 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 such applications depend on the ability of a system to capture the complex inter-dependencies 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]. Analyzing 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 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. The ensemble of decision trees, Random Forest, has helped in limiting the problem of over-fitting. Though Random Forest is still a popular algorithm, gradient tree boosting, or more popularly, gradient boosting decision trees (GBDT) has 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 the presence of foodborne pathogens in the soil and found that GBDT outperformed RF significantly. Rao et. al. [6] used the Ant bee colony method coupled with GBDT for feature selection in six high dimensional data sets 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 the 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) [8], LightGBM [9] and CatBoost [10]. 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.

1.2 Motivation

It is a widely accepted fact that in designing a machine learning solution, the data preprocessing 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 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 data set.

Most state-of-the-art algorithms are all gradient boosting decision trees. Though the underlying method of building trees is the same in all, the method of split and feature selection is varied. In such a 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, others may perform better with data having more numerical features. In such a situation, the need for an extensive comparison of the combination of encoding techniques with classification algorithms cannot be overemphasized.

1.3 Objective

The main objective of this dissertation 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 data sets, while not on others.

1.4 Challenges

While there is no dearth of data in present times, finding several suitable open-access data sets is still a task. In order to make the comparison of algorithms fair, it is imperative that the data sets 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 dissertation, the data sets were carefully selected from reliable sources such as UCI Machine Learning Repository, Kaggle data sets and data.world, and also from government websites publishing open-access data for research.

All the GBDT algorithms considered in the dissertation have a tens of hyper-parameters that can be adjusted for best fit of the model to the data set. However, searching for these is computationally expensive and time-taking task. GridSearchCV is a module provided by Scikit-learn [11] which iterates over a list of hyper-parameters to find the optimum ones. This was used in the dissertation.

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

Chapter 2

Handling Categorical Features

Machine Learning algorithms require the data set features to be numeric. However, in the real world, the data generally contains a mixture of numeric and categorical features. Encoding techniques are used to address this problem. This chapter first discusses some previous works attempting to encode the categorical data. After a review of the literature, we present a classification of the encoding techniques for a systematic selection of the technique.

2.1 Literature Review

This section discusses some of the attempts made by researchers in encoding.

A transformation scheme was proposed by Barreca et. al. [12] 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. [13] 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 a couple of techniques based on the transformation of identified features to a target statistic. The first method is Weight of Evidence, which replaces churners with the log of the 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 a social network perspective, given by

$$SR_i^X = \frac{C_i^X}{C_i^X + N_i^X} \tag{2.1}$$

where C_i^X and N_i^X are again the number of churners and the number of non-churners for the i-th 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. [14] 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 a one-hot encoded vector, but shorter in length. It is given by:

$$x_i \equiv \sum_\alpha \omega_{\alpha\beta}\delta_{x_i\alpha} = \omega_{x_i\beta} \tag{2.2}$$

where $\delta_{x_i\alpha}$ is a vector of length equal to the number of categories in the variable x_i, in which the element is only non-zero when complexity parameter $\alpha = x_i$. $\omega_{\alpha\beta}$ 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 the 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 are concatenated, and the merged layer is treated as any normal Neural net layer. Through experimentation, the authors show that this method can improve the 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 the gradient boosting model, Janek et. al. [15] 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 GBDT hyper-parameters.

While designing an alternative method of building trees, a different approach is taken by Nguyen et. al. [16] for 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 the 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. [17] 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 the 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 with workers in distributed systems. The drawback is, however, loss in interpretability as the strings get hashed. To provide interpretability, authors suggest another encoding technique, Gamma-Poisson matrix factorization. This method relies on sub-string 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 the 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.

2.2 Classifying Category Encoders

2.2.1 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 Gender_Female are both 0, then Gender_Other becomes 1, meaning that the 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 of the number of features increases as the number of categories increases. 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 the 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 the 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.

2.2.2 Contrast Encoders

Helmert Encoding

In Helmert encoding [18], the mean of the target for a level or category is compared to the mean of the 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 the target variable for a level is compared to the target mean of the adjacent level. If the subsequent level is considered, it is called forward difference encoding. If the 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 the target mean of all other k-1 levels.

2.2.3 Bayesian Encoders

Target Encoding

This method introduces a transformation scheme [12] 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. To avoid over-fitting due to a small number of observations in a category, smoothing of the means is also applied. Probability estimate for a category within a high cardinality categorical variable can be given by Empirical Bayesian probability, $P(Y = 1|X = X_i)$, i.e.

$$S_i = \lambda\left(n_i\right)\frac{n_{iY}}{n_i} + \left(1 - \lambda\left(n_i\right)\right)\frac{n_Y}{n_{TR}} \tag{2.3}$$

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

$$\lambda(n) = \frac{1}{1 + e^{\frac{-(n-k)}{f}}} \tag{2.4}$$

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 the 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 X_0. This has an advantage that if nulls have a predictive relevance, then X_0 will capture that information, otherwise, it will converge towards the prior probability of target, leading to 0 effect of X_0.

Weight of Evidence Encoding

This method [13], having originated from the credit scoring sector, can be used to transform a category of the feature into a numeric value by taking a relative ratio of appearance of the category in the data set.

$$WOE_i^X = ln(\frac{C_i^X/TC}{N_i^X/TN}) \tag{2.5}$$

TC and TN define the total number of instances of target class c1 versus target class c2; C_i^X and N_i^X denote the number of c1 and c2 for the i^{th} value of attribute X. However, in the case when values are zero in a particular category, 1 row is added for that category, and the overall ratio is 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 over-fitting.

Leave One Out Encoding

This method was introduced to counter the effects of outliers in the training data. Similar to Target Encoding, the 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 over-fitting while giving more weight to the global mean would lead to under-fitting. In order to solve this problem, the James-Stein encoder gives a lesser weight to a category if the variance in values of that category is high as compared to the overall variance in the target.

2.3 Summary

This chapter provided an overview of the encoding techniques by reviewing the literature and a classification to put the methods in context. We saw that the encoding techniques can be broadly classified as Classic Encoders, Contrast Encoders, and Bayesian Encoders. However, for the purpose of the dissertation, we only pick one-hot encoding and target encoding, as these are the most popular methods.

Chapter 3

Gradient Boosting Decision Trees

This chapter starts with the history and development of boosting as an ensemble technique, which led to the development of GBDTs and their present state. Section 3.1 describes several implementations of GBDTs and then describes the concept of GBDTs in detail.

Next, we present the three GBDT algorithms being compared in this dissertation, namely, XGBoost, LightGBM, and CatBoost. Section 3.2 reviews the concepts that these three algorithms use, the innovations and how they handle categorical variables.

3.1 Literature Review of GBDTs

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 an iterative fashion, each one learning from the mistakes of the previous learner.

3.1.1 Early Developments

All boosting algorithms are aimed at optimizing a suitable differentiable loss function. The idea was observed first by Leo Breiman [19] 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 [20], 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, over-fitting 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 a logistic function is used for measuring the cost of each iteration, the algorithm is known as LogitBoost [21]. The authors of this framework proved statistically that weak learners are additive on a logistic scale. Another boosting algorithm, BrownBoost [22] is specialized in data sets with noisy observations. This algorithm assumes that if learners are repeatedly mis-classifying certain observations, then such observations are outliers and it removes such samples from the data. Domingo et. al. [23] 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 data point. This proved to be more robust to class-label noise.

3.1.2 Building a GBDT Model

During the improvements of existing frameworks, there was a rising interest in another class of boosting algorithms, called gradient boosting algorithms. Freidman [24, 25] 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 the gradient boosting method. Conventionally, at the m^{th} step in gradient boosting, a decision tree $h_m(x)$ is fitted to pseudo-residuals. Assuming J_m as the number of leaves, the tree partitions the input space into J_m disjoint regions $R_{1m}, \cdots, R_{J(m)m}$. Finally, a constant value is predicted in all the regions. In the indicator notation, for input x, $h_m(x)$ gives an the output as the sum:

$$h_m(x) = \sum_{j=1}^{J_m} b_{jm} \mathbf{1}_{R_{jm}}(x) \tag{3.1}$$

where b_{jm} are multiplied by some value $\gamma(m)$, which can be selected using line search with the goal of minimizing the loss function. Then we can update the model as below:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x) \tag{3.2}$$

$$\gamma_m = arg\, min_\gamma \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)) \tag{3.3}$$

Freidman proposed a modified version of this approach and called it ‘TreeBoost’ [25]. In his algorithm, a distinct optimal value is chosen for each region of the tree. This value is γ_{jm}. Then the coefficients b_{jm} become obsolete and model can be update as:

$$F_m(x) = F_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} \mathbf{1}_{R_{jm}}(x) \tag{3.4}$$

$$\gamma_{jm} = arg\, min_\gamma \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(x_i) + \gamma) \tag{3.5}$$

In many GBDTs, building the 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 leaves, scores the trees, and selects the best split. Leaf values in both phases are calculated as approximations for gradients or for Newton steps.

3.1.3 Implementations

Though the underlying principle is the same, there have been many variants of GBDTs since its inception.

Feng et. al. [26] proposed sGBM (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 their implementation called PRGBM (Partially Randomized Gradient Boosting Machine), Konstantinov et. al. [27] attempted to take into account cases when regions of training data are not densely covered by observations. They used partially randomized trees to solve the problem and demonstrated that their model performed better than all existing GBDT algorithms. The proposed solution also reduced the computational complexity of the models.

The multi-layered GBDT forest (mGBDTs) [28] were an interesting development, which combined the concept of layers from neural networks, with classic GBDTs. Feng et.a l. emphasized explicitly on exploring the ability to learn hierarchical distributed representations by stacking several layers of regression GBDTs as its building block. The model can be jointly trained by a variant of target propagation across layers, without the need to derive back-propagation nor differentiability. Experiments confirmed the effectiveness of the model in terms of performance and representation learning ability.

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

3.2 State Of The Art in GBDTs

3.2.1 XGBoost

Started as a research project in 2014, XGBoost [29], 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 the 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 maximizes the information gain. Analyzing every possible split makes this algorithm highly precise; however, this also leads to computational overload and slower processing.

On the other hand, the 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 a global variant, wherein the 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 employing 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.

3.2.2 LightGBM

LightGBM [30] 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 to find the best split. The authors introduce two novel ideas for optimization, 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 a small gradient can be removed from the data as they already have small training errors. 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 a large gradient.

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

3.2.3 CatBoost

CatBoost [31], which derives the name from Categorical Boosting, is the most recent in popular implementations of the GBDT algorithm, proposed by the Yandex group. The authors contend that the limitations of all existing GBDT implementations are 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.

The ordering principle is based on the concept of online learning algorithms, in which the training samples are provided to the algorithm in time, sequentially. Similarly, CatBoost creates artificial time by creating random permutations of the training samples. Different permutations are used for the training in different steps of gradient boosting so that the variance can be reduced. This prevents target leakage during encoding of categorical 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 samples as the index number of that tree. The residuals for the next observation are calculated using the previous tree. This makes sure that the residuals are calculated using a model that was not trained on the current observation.

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

3.2.4 Handling Categorical Features

XGBoost

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

LightGBM

LightGBM [32] 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 is sorted according to its accumulated values (sum_gradient/sum_hessian) and then the best split on the sorted histogram is obtained.

CatBoost

To handle categorical features, CatBoost [33] provides the option of both one-hot encoding as well as target statistics methods. One-hot encoding is generally used when the number of categories in a feature is relatively small. The hyper-parameter one_hot_max_size can be used to set a value threshold for the number of categories in a feature, above which the target statistic method is used. For encoding using target statistics, CatBoost first generates random permutations of the training data to facilitate 'online learning'. For each permutation σ, a subset of data $\mathcal{D}_k = \{x_j : \sigma(j) < \sigma(k)\}$ 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 $\mathcal{D}$ is taken. The category value is then calculated as

$$\hat{x}_k^i = \frac{\sum_{x_j \in \mathcal{D}_k} 1_{x_j^i = x_k^i} . y_j + \ a.p}{\sum_{x_j \in \mathcal{D}_k} 1_{x_j^i = x_k^i} . y_j + \ a} \tag{3.6}$$

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 that have a lower frequency. For regression tasks, the standard technique for calculating prior is to take the average label value in the data set. For classification tasks, a prior is usually an a priori probability of encountering a positive class. After the first split in a tree, CatBoost combines all categorical features of the data set 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.

3.3 Summary

This chapter explained in detail, the evolution of boosting as an ensemble technique, to finally various implementations of GBDTs. We also examined a basic method of constructing a GBDT algorithm. The state-of-the-art implementations were discussed and their innovations noted.

Chapter 4

Methodology

4.1 Data sets

Table 4.1 shows the data sets used and the features of each data set. The data sets were obtained from trusted sources such as data.world, Kaggle data sets, and UCI ML repository. Care was taken so that the chosen data sets have diverse characteristics. Therefore, the observations vary from 1,473 in the Contraceptive data set to almost a million in the Physician data set and features range from 5 in Land process data set through 32 in the kick data set. There is a mix of categorical and numeric features, with two data sets being purely categorical. Highest cardinality represents the maximum number of categories present in any of the categorical features of the data set.

The data sets were selected to come from diverse backgrounds and of a different

Name	Observations (thousand)	Features	Categorical Features	Highest Cardinality	No. of Classes
Adult	32.5	14	8	42	2
Amazon	32.8	9	9	7518	2
Bank Marketing	41.1	20	10	12	2
Contraceptive	1.5	9	4	34	3
Kick	72.9	32	15	1063	2
Land Prices	3.5	5	4	60	10
Physician	969	8	8	19574	4
Poker Hand	25	10	5	4	10

Table 4.1: Data sets and their characteristics

number of classes so that the results of the experiment are not biased. Following is the description of the data sets:

1. The Adult data set [34] has demographic information of individuals, the task is to predict whether an individual earns more or less than $50,000 a year.

2. Amazon [35] data set carries information about an employee's role at the firm, the task is to predict if the employee should be granted access to a resource or not. These auto-access models are expected to revoke or grant access to employees and thus minimize the involvement of humans.

3. Bank marketing [36] data set provides client information for a Portuguese banking institution. The classification goal is to predict if the client will subscribe to a term deposit or not.

4. The Contraceptive [37] data set is a subset of the 1987 National Indonesia Contraceptive Prevalence Survey. The samples of this data set are married who did not know about their pregnancy when they were interviewed or those who were not pregnant at all. The goal in this data set is to predict a woman's current preferred contraceptive method (short-term, long-term methods, no use) based on her socio-economic and demographic characteristics.

5. Kick [38] data set 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.

6. Land prices [39] data set sources information regarding the value per acre of farmland in each state/region in the United States since 1997, as published by the National Agricultural Statics Service (NASS). The land prices have been grouped into 10 buckets, which is to be predicted.

7. The Physician [40] data set identifies prescription drug 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 data set to predict the number of patients visiting the physician.

8. Poker hand [41] data set 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.

4.2 Experimental Design

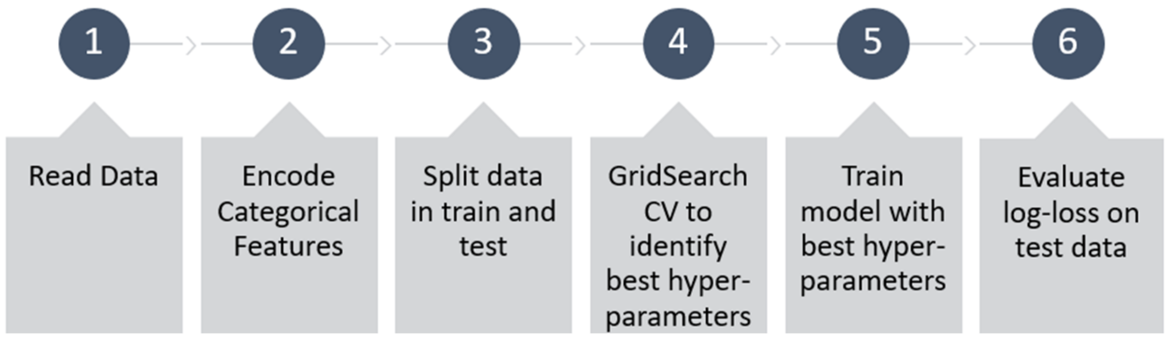
Along with the three algorithms of interest, namely XGBoost, LightGBM, and CatBoost, Random Forest has also been considered, as this is one of the popular, interpretable, and reliable method of getting a baseline result. Therefore, we have four algorithms to test their classification performance.

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 chapter 2. 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 a check on the horizontal explosion of the data set 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 data sets, 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 4.1. We first read the data in Python using the Pandas package [42]. We encode the categorical features using one of the encoding techniques and split it into train and test data with an 80:20 ratio of observations. Next, we apply Scikit-learn′s GridSearch method [11], which takes in a dictionary of permissible values of the hyper-parameters and trains the train data. This method helps in finding the best combination of hyper-parameters and this best model is then evaluated on the test data.

Figure 4.1: Steps showing experimental set-up


1
Read Data
2
Encode Categorical Features
3
Split data in train and test
4
GridSearch CV to identify best hyper-parameters
5
Train model with best hyper-parameters
6
Evaluate log-loss on test data

4.3 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, the 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, the accuracy metric fails. First, when there are more than two classes. In such a scenario, the metric does not give information about the individual accuracy for each class. It is possible the model ignores one or more classes. Second, when the data has imbalanced classes. For instance, if a binary classification data set has a class imbalance ratio of 90:10, then a model with 90% classification accuracy is a non-informative one.

In order to tackle these limitations, the 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 that are correctly identified. Improving the F1 score helps in increasing precision and recall at the same time. Taking a harmonic mean makes sense because as compared to the arithmetic mean, harmonic mean punishes extreme values more.

$$F_1 = \left(\frac{recall^{-1} + precision^{-1}}{2}\right)^{-1} = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{4.1}$$

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 the 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 a negative average of the log of corrected predicted probabilities for each instance.

For binary classification with a true label $y \in \{0, 1\}$ and a probability estimate $p = Pr(y = 1)$, the log loss [43] per sample is the negative log-likelihood of the classifier given the true label:

$$L_{\log}(y, p) = -\log Pr(y|p) = -(y \log(p) + (1 - y) \log(1 - p)) \tag{4.2}$$

This extends to the multi-class 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. $y_{i,k} = 1$ if sample i has label k taken from a set of K labels. Let P be a matrix of probability estimates, with $p_{i,k} = Pr(t_{i,k} = 1)$. Then the log loss of the whole set is

$$L_{\log}(Y, P) = -\log Pr(Y|P) = -\frac{1}{N} \sum_{i=0}^{N-1} \sum_{k=0}^{K-1} y_{i,k} \log p_{i,k} \tag{4.3}$$

In this dissertation, a python API provided by Sklearn library [43] has been used to calculate the log loss, which takes as input the actual labels and estimated probability of each class. Baseline for each data set 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 data set. As can be seen in table 4.2, the random guessing log loss varies considerably across data sets.

Name	No. of Classes	Class Prevalence	Random Guessing Log Loss
Adult	2	[0.7592, 0.2408]	0.552
Amazon	2	[0.9421, 0.0579]	0.221
Bank Marketing	2	[0.8873, 0.1127]	0.352
Contraceptive	3	[0.427, 0.3469, 0.2261]	1.067
Kick	2	[0.877, 0.123]	0.37286
Land Prices	10	[0.4378, 0.3242, 0.1217, 0.0468,...]	1.418
Physician	4	[0.252, 0.2502, 0.2497, 0.2481]	1.386
Poker Hand	10	[0.4995, 0.4238, 0.0482, 0.0205, ...]	0.986

Table 4.2: Data sets and their Random Guessing log loss

4.4 Summary

This chapter described the data and steps being followed for the analysis. We also discussed various evaluation metrics and concluded that log-loss is the best choice for our purpose.

We discuss the results in the next chapter.

Chapter 5

Results and Discussion

5.1 Introduction

For each data set, 14 combinations of encoding technique and classification algorithm was run. It was observed that at least one combination always performed better than the random guess baseline for all data sets. Figure 5.1 shows that the highest predictive power was gained in the Land prices data set, where log-loss reduced by 82.7%, while least in the Physician data set, where log-loss reduced by only 6.4%. However, it should be noted that these predictive powers can not be compared among data sets as log-loss is a function of the number of classes and the class ratio of the data set.

Figure 5.1: Comparison of best model and random guessing log-losses

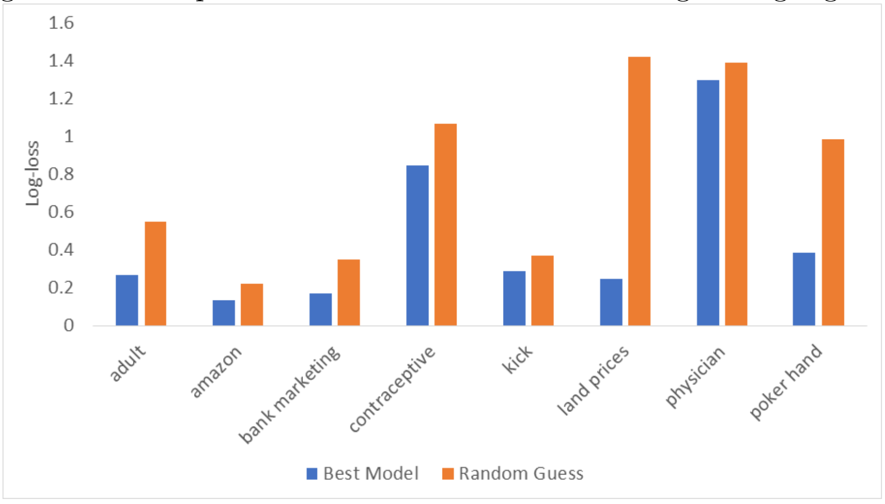
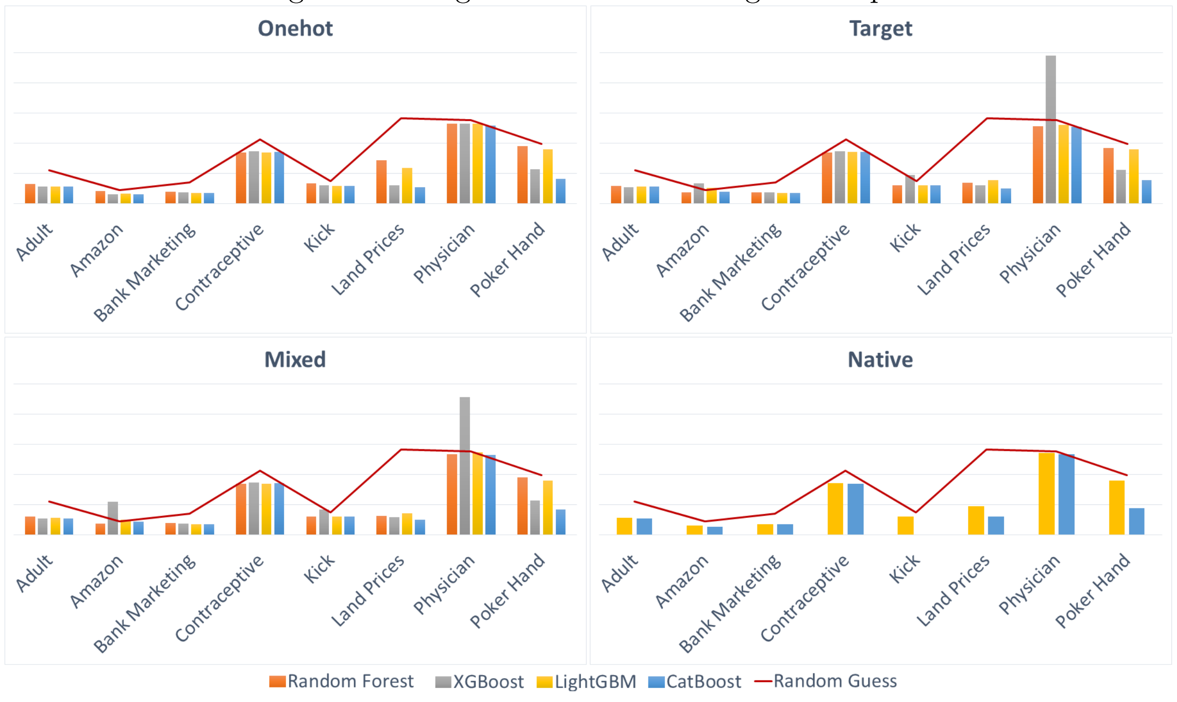


Figure 5.2: Log loss across Encoding Techniques

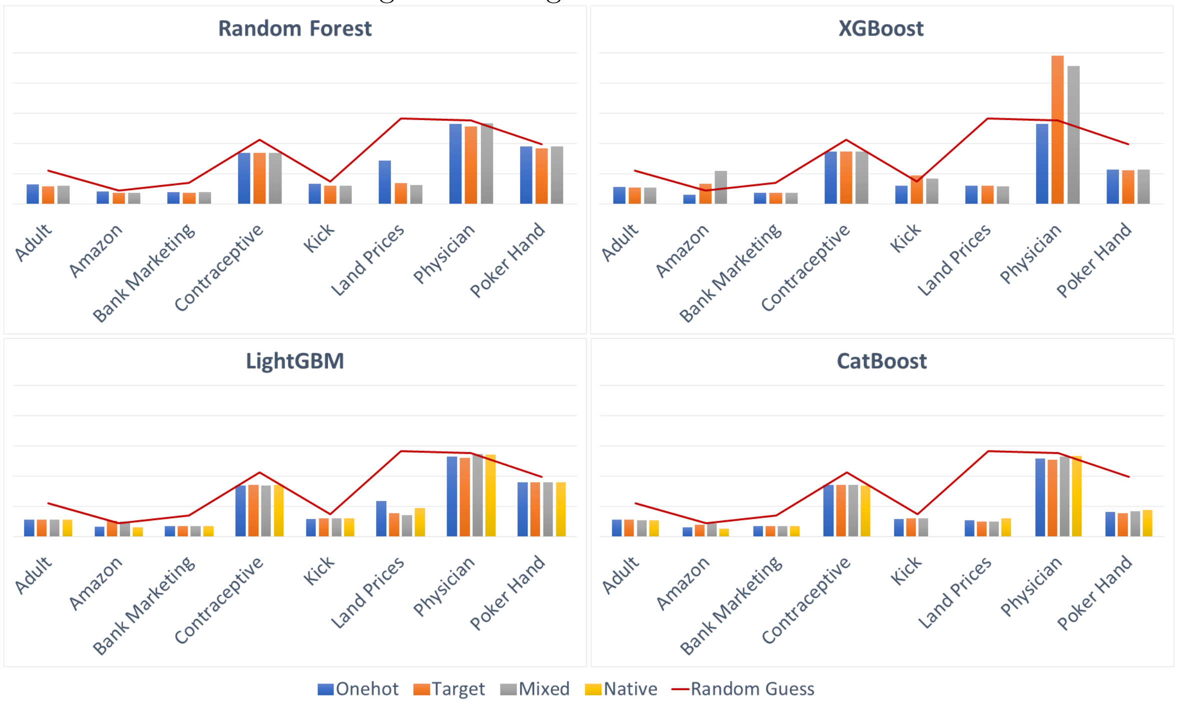


5.2 Comparing Encoding Algorithms

Figure 5.2 shows the performance of each of the four encoding techniques across data sets for all the four algorithms. It can be noted that one-hot encoding performs better than the random guess baseline in all combinations. This is contrary to target encoding, where XGBoost performs worse than baseline for Amazon, Kick, and Physician data sets. However, it should be noted that otherwise, the 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 the other three encoding techniques. Not only do they always perform much better than the baseline, but they also 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 as gives a 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 5.3: Log loss across GBDTs



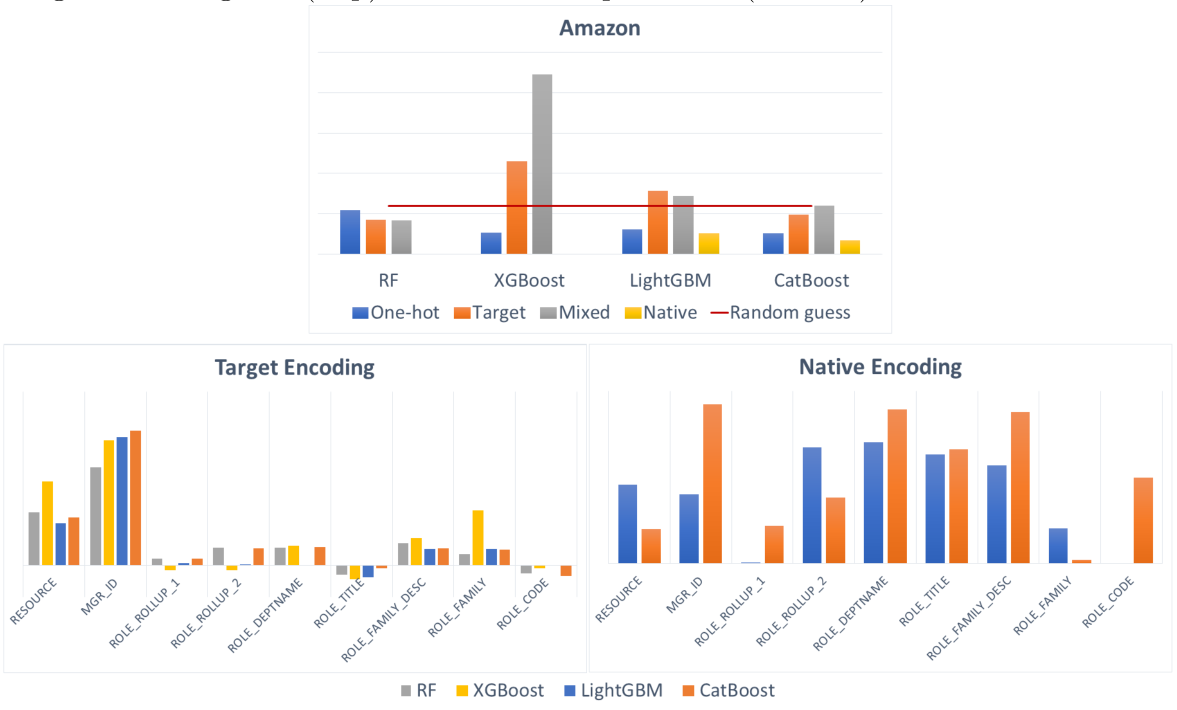
Next, we compare the performances of the gradient boosting decision tree algorithms.

5.3 Comparing GBDT Algorithms

Figure 5.3 compares each of the algorithms against encoding techniques. It can be seen in figure 5.3 that log loss in RF is always less than random guessing baseline, meaning RF performs better; however, it should be noted the performance is only marginally better in most data sets. 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 the superiority of CatBoost over other algorithms. Besides, it reveals that XGBoost performs best with the one-hot encoding technique. This is possibly because it is designed to expertly handle only numeric features with

Figure 5.4: Log loss (Top) and Feature Importances (Bottom) in Amazon Data set



histogram-based splitting. On the other hand, CatBoost relies on ordered boosting and handles categorical features in a similar fashion as mixed encoding. The coupling of these two features may account for the performance lift.

5.4 Feature Importance in Single Class vs Multi Class

To investigate the performance on a deeper level, we dive into the Amazon data set and Physician data set. All the features of these two data sets are categorical, as such, it should be interesting to see how the combination of algorithms performs on them. We compare and contrast the feature importance in these two data sets across target encoding and native encoding. The chart on top in Figure 5.4 shows the relative performance of the 14 models run on this data set. One-hot encoding performs much better than the target and mixed encodings for XGBoost, which is in agreement with

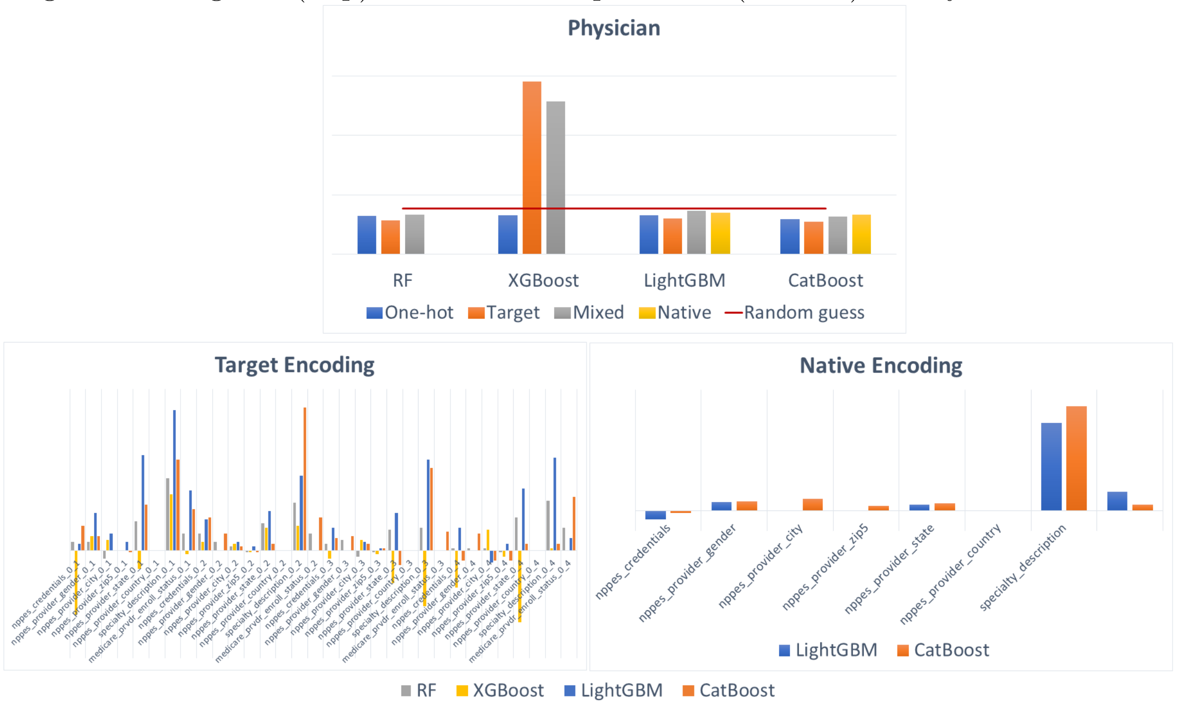


Figure 5.5: Log loss (Top) and Feature Importances (Bottom) in Physician Data set

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 5.6% as well.

Looking at the charts in bottom in figure 5.4, one can contrast in the feature importance of the models with target encoding versus native encodings; 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.

For the Physician data set, top chart in figure 5.5 shows the 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 data set with the Amazon data set, it should be noted that the Physician data set has 4 classes as opposed to only two classes in Adult, and has a much higher cardinality of 19,574 as compared to 7,518 of the Adult data set. These two characteristics make the prediction task in the Physician data set much more difficult as compared to that in the Adult data set.

Charts in the bottom of figure 5.5 indicate how the algorithms deal with the features in the Physician data set. 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 the native encoding. It can be seen that only two features are being used more in the native encoding.

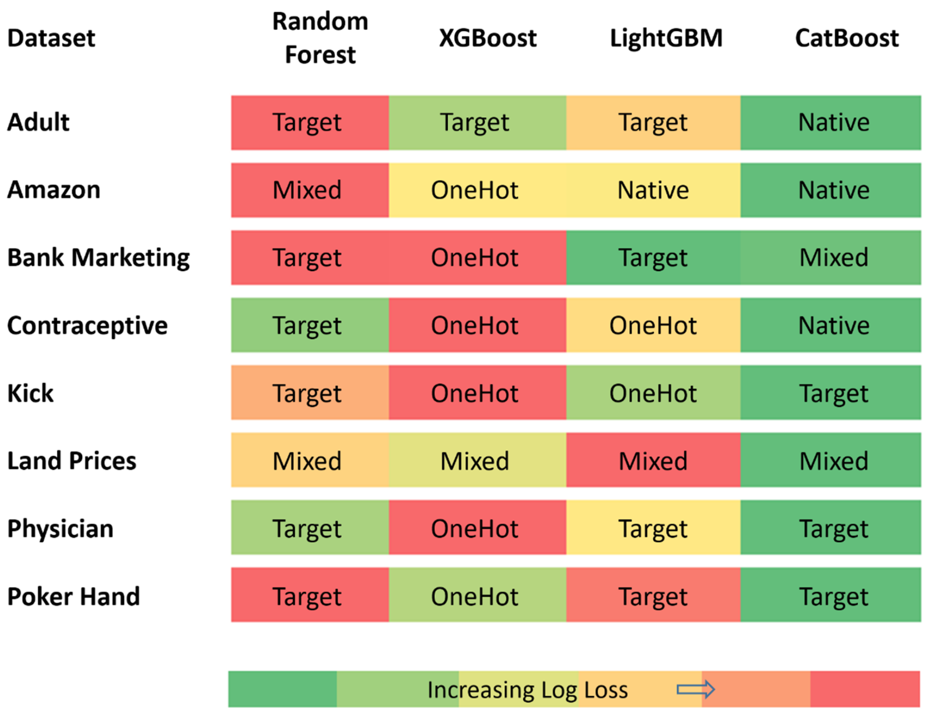
While comparing the feature importance in Amazon versus Physician data set, we see quite a reversal in the behavior of target encoding and native encoding. This could possibly be because the Amazon data set is a binary classification task while Physician is a four-class classification. This means that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

This behavior of the multi-class data set insinuates that the category-handling GBDT algorithms LightGBM and CatBoost are optimized for binary classification tasks.

5.5 Summary

Finally, we compare the performance of all the 112 models simultaneously. With 8 data sets, 4 algorithms, and 4 encoding techniques, we have a three-dimensional matrix to visualize. In order to simplify, we look at the relative performance of the models as shown in figure 5.6. The data sets are stacked vertically, while algorithms are stacked horizontally. Each cell gives the name of the best performing encoding technique for a given combination of the data set and GBDT algorithm. Once we have this information for all four GBDT algorithms for one data set, we compare the four best combinations.

Figure 5.6: 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 data set.



Dataset	Random Forest	XGBoost	LightGBM	CatBoost
Adult	🟥 Target	🟩 Target	🟨 Target	🟩 Native
Amazon	🟥 Mixed	🟨 OneHot	🟨 Native	🟩 Native
Bank Marketing	🟥 Target	🟥 OneHot	🟩 Target	🟩 Mixed
Contraceptive	🟩 Target	🟥 OneHot	🟨 OneHot	🟩 Native
Kick	🟧 Target	🟥 OneHot	🟩 OneHot	🟩 Target
Land Prices	🟨 Mixed	🟨 Mixed	🟥 Mixed	🟩 Mixed
Physician	🟩 Target	🟥 OneHot	🟨 Target	🟩 Target
Poker Hand	🟥 Target	🟩 OneHot	🟧 Target	🟩 Target

🟩 🟩 🟨 Increasing Log Loss ⇨ 🟧 🟥

Therefore, the first cell in figure 5.6 denotes that for Random Forest in the Adult data set, Target encoding had the lowest log-loss among the four encoding techniques. Similarly, for XGBoost in the Adult data set, 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 the highest log-loss, while Native Encoding with CatBoost performed the best with the least log-loss across the row.

Looking at the results in figure 5.6 collectively, it is easily seen that CatBoost performs the best for all data sets with one of the four encoding techniques. In fact, it never performs well with one-hot encoding. This is in contrast to XGBoost, which frequently shows low log-losses with one-hot encoding.

Chapter 6

Security and Privacy

The dissertation work is done in two phases. The first phase, training, includes collecting data sets, cleaning of data, and then model training. The second phase, testing, comprises evaluating the performance and then measuring the model performances against each other. This chapter discusses possible security issues and their solutions, during the course of this work and henceforth.

6.1 Training Phase

The data for this work was obtained from open-source providers. Such open data could be poisoned. This means that the data could be maliciously doctored and injected with observations to explicitly cause misclassification in models [44]. Such models could be a threat to the security of systems where the models are deployed. For instance, in an application using the Amazon data set, incorrectly granting access of resources to an employee could cause major business havoc. Such an attack is called an adversarial attack.

Another threat during the training phase could be the infection of the system with malware. While downloading the data sets from the internet, downloading malware inadvertently is entirely plausible. These malware include viruses, ransomware, trojans, spyware, and keyloggers.

The attacks discussed above can be easily thwarted by exercising caution while interacting with online resources and only downloading from trusted sources, which have been used extensively by other researchers. The effects of poisoning attacks can be negated by bootstrap aggregation and feature sampling methods. These measures are already taken into account by GBDT algorithms. Also, keeping the system up-to-date with the latest firmware and antivirus software should be enough to minimize the possibility of malware attacks.

6.2 Testing Phase

When the models developed with the help of this work are deployed in the real world, they are still susceptible to attacks. When an attacker learns of the weakness or vulnerability of a model, he can launch what is called an Evasion attack. Generating data similar to actual data and using it to gaining access to systems is called an Impersonation attack. Reverse engineering the model to learn about the data used for training is called an Inversion attack.

The scope of this dissertation is limited to comparing the performance of GBDT algorithms. As such, the security capabilities of the algorithms cannot be determined and is out of the scope of this work.

6.3 Summary

While the security concerns presented in this chapter are real, the risk is low. This is because it is unlikely that an attacker would invest their resources in poisoning open-access data sets on the internet. The risk of malware is considerably low considering the strong security of genuine windows software.

Chapter 7

Conclusion

In this dissertation, the performance of four gradient boosting decision trees algorithms, namely Random Forest, XGBoost, LightGBM, and CatBoost, was compared for classification on eight data sets 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 data set 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 the 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 the GBDT algorithm designed to handle categorical features natively, CatBoost, has been optimized for binary classification tasks, as such, it does not perform relatively well on multi-class data sets. However, with target encoding, these algorithms can beat other GBDTs.

Chapter 8

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 optimized. There could be a comparison of algorithms across metrics to be optimized.

The model 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.

The 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.

Bibliography

[1] N. Martin, "How much data is collected every minute of the day." `https://www.forbes.com/sites/nicolemartin1/2019/08/07/how-much-data-is-collected-every-minute-of-the-day/#29902a923d66`, 8 2019. Accessed: 2020-08-30.

[2] D. L. Donoho, "High-dimensional data analysis: The curses and blessings of dimensionality," in *AMS CONFERENCE ON MATH CHALLENGES OF THE 21ST CENTURY*, 2000.

[3] J. Ogutu, H.-P. Piepho, and T. Schulz-Streeck, "A comparison of random forests, boosting and support vector machines for genomic selection," *BMC proceedings*, vol. 5 Suppl 3, p. S11, 05 2011.

[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," *IEEE Geoscience and Remote Sensing Letters*, vol. PP, pp. 1–5, 02 2018.

[5] C. Golden, M. Rothrock, and A. Mishra, "Comparison between random forest and gradient boosting machine methods for predicting listeria spp. prevalence in the environment of pastured poultry farms," *Food Research International*, vol. 122, 03 2019.

[6] H. Rao, X. Shi, R. Ahoussou, J. Feng, Y. Xia, M. Elhoseny, X. Yuan, and L. Gu, "Feature selection based on artificial bee colony and gradient boosting decision tree," *Applied Soft Computing*, vol. 74, 11 2018.

[7] P. Yuliang, D. Liu, and L. Deng, "Accurate prediction of functional effects for variants by combining gradient tree boosting with optimal neighborhood properties," *PLOS ONE*, vol. 12, p. e0179314, 06 2017.

[8] "Xgboost documentation." `https://xgboost.readthedocs.io/en/latest/\#`. Accessed: 2020-08-30.

[9] "Lightgbm documentation." `https://lightgbm.readthedocs.io/en/latest/`. Accessed: 2020-08-30.

[10] "Catboost documentation." `https://catboost.ai/docs/`. Accessed: 2020-08-30.

[11] "Scikitlearn documentation." `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html`. Accessed: 2020-08-30.

[12] D. Micci-Barreca, "A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems.," *SIGKDD Explorations*, vol. 3, pp. 27–32, 07 2001.

[13] J. Moeyersoms and D. Martens, "Including high-cardinality attributes in predictive models a case study in churn prediction in the energy sector," *Decision Support Systems*, vol. 72, 02 2015.

[14] C. Guo and F. Berkhahn, "Entity embeddings of categorical variables," 04 2016.

[15] J. Thomas, S. Coors, and B. Bischl, "Automatic gradient boosting," 07 2018.

[16] T.-T. Nguyen, J. Huang, T. Nguyen, and w. qiang, "An efficient random forests algorithm for high dimensional data classification," *Advances in Data Analysis and Classification*, vol. 12, 04 2017.

[17] P. Cerda and G. Varoquaux, "Encoding high-cardinality string categorical variables," 07 2019.

[18] "R library contrast coding systems for categorical variables." `https://stats.idre.ucla.edu/r/library/r-library-contrast-coding-systems-for-categorical-variables/#HELMERT`. Accessed: 2020-08-30.

[19] L. Breiman, "Arcing the edge," 1997.

[20] R. E. Schapire, "A brief introduction to boosting," in *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'99, (San Francisco, CA, USA), p. 1401–1406, Morgan Kaufmann Publishers Inc., 1999.

[21] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: A statistical view of boosting," *The Annals of Statistics*, vol. 28, pp. 337–407, 04 2000.

[22] Y. Freund, "An adaptive version of the boost by majority algorithm," in *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, COLT '99, (New York, NY, USA), p. 102–113, Association for Computing Machinery, 1999.

[23] C. Domingo and O. Watanabe, "Madaboost: A modification of adaboost," *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, 06 2000.

[24] J. Friedman, "Greedy function approximation: A gradient boosting machine," *The Annals of Statistics*, vol. 29, 11 2000.

[25] J. Friedman, "Stochastic gradient boosting," *Computational Statistics & Data Analysis*, vol. 38, pp. 367–378, 02 2002.

[26] J. Feng, Y.-X. Xu, Y. Jiang, and Z.-H. Zhou, "Soft gradient boosting machine," 06 2020.

[27] A. Konstantinov and L. Utkin, "Gradient boosting machine with partially randomized decision trees," 06 2020.

[28] J. Feng, Y. Yu, and Z.-H. Zhou, "Multi-layered gradient boosting decision trees," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, (Red Hook, NY, USA), p. 3555–3565, Curran Associates Inc., 2018.

[29] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery*

and Data Mining, KDD '16, (New York, NY, USA), p. 785–794, Association for Computing Machinery, 2016.

[30] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, (Red Hook, NY, USA), p. 3149–3157, Curran Associates Inc., 2017.

[31] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: Unbiased boosting with categorical features," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, (Red Hook, NY, USA), p. 6639–6649, Curran Associates Inc., 2018.

[32] "Optimal split for categorical features." `https://lightgbm.readthedocs.io/en/latest/Features.html\#optimal-split-for-categorical-features`. Accessed: 2020-08-30.

[33] "Transforming categorical features to numerical features." `https://catboost.ai/docs/concepts/algorithm-main-stages_cat-to-numberic.html`. Accessed: 2020-08-30.

[34] "Adult income." data retrieved from UCI Machine Learning Repository, `https://archive.ics.uci.edu/ml/datasets/Adult/`.

[35] "Amazon access." data retrieved from Kaggle data sets, `https://www.kaggle.com/c/amazon-employee-access-challenge`.

[36] "Bank marketing." data retrieved from World Development Indicators, `https://data.world/data-society/bank-marketing-data`.

[37] "Contraceptive." data retrieved from UCI Machine Learning Repository, `https://archive.ics.uci.edu/ml/datasets/Contraceptive+Method+Choice`.

[38] "Dont get kicked." data retrieved from Kaggle data sets, `https://www.kaggle.com/c/DontGetKicked`.

[39] "Land prices." data retrieved from Kaggle data sets, `https://www.kaggle.com/jmullan/agricultural-land-values-19972017/home`.

[40] "Prescriber summary." data retrieved from World Development Indicators, https://data.cms.gov/Medicare-Part-D/Medicare-Provider-Utilization-and-Payment-Data-Par/psut-35i4.

[41] "Poker hand." data retrieved from UCI Machine Learning Repository, https://archive.ics.uci.edu/ml/datasets/Poker+Hand.

[42] "Pandas documentation." https://pandas.pydata.org/. Accessed: 2020-08-30.

[43] "Log-loss." https://scikit-learn.org/stable/modules/model_evaluation.html#log-loss. Accessed: 2020-08-30.

[44] Q. Liu, P. Li, W. Zhao, W. Cai, S. Yu, and V. C. M. Leung, "A survey on security threats and defensive techniques of machine learning: A data driven view," *IEEE Access*, vol. 6, pp. 12103–12117, 2018.

Appendix: List of Abbreviations

AdaBoost	Adaptive Boosting
AUC	Area Under Curve
CatBoost	Categorical Boosting
EFB	Exclusive Feature Bundling
GBDT	Gradient Boosting Decision Tree
GOSS	Gradient-based One-Side Sampling
PR	Perlich's Ratio
RF	Random Forest
SR	Supervised Ratio
SVM	Support Vector Machine
TC	Total Churners
TN	Total Non-churners
TPR	True Positive Rate
WOE	Weight Of Evidence
XGBoost	Extreme Gradient Boosting