International Journal of Advanced Technology & Engineering Research
(IJATER) www.ijater.com

Predictive Analysis of Opiod & Non-Opiod Prescriber Using XGBoost and Improved XGBoost System

Mr. Himalaye Goiya[1], Mr. Harsh Lohiya[2], Mr. Kailash Patidar[3],
Department of Computer Science & Engineering
Sri Satya Sai University of Technology & Medical Science, Sehore

Abstract- Tree boosting has exactly ended up being a profoundly effective way to deal with predictive modeling. It has indicated significant outcomes for an immense range of issues. We will demonstrate that XGBoost utilizes a boosting calculation which we will term Newton boosting. This boosting calculation will additionally be contrasted and the gradient boosting calculation that MART utilizes. In addition, we will examine the regularization procedures that these strategies offer along with the effect these have on the models. We examine the XGBoost and Improved XGBoost claasifiers for both Opiod as well as Non – Opiod prescribers in terms of Accuracy and AUC score.

Keywords: XGBoosting, Machine Learning, Accuracy, Tree Model, Predictive Modeling

1. Introduction

Gradient boosting is an intense machine learning method presented by Fried-man [2]. The procedure was propelled similar to a gradient descent strategy in work space, equipped for fitting nonexclusive nonparametric prescient models. Gradient boosting has been especially fruitful when connected to tree models, in which case it fits additive tree models.

1.1 Risk Minimization: Defining the Target

In this area, we will present the loss function. The loss function is the measure of forecast accuracy that we characterize for the current issue. We are at last intrigued by limiting the normal loss, which is known as the risk. The function which limits the risk is known as the target function. This is the ideal forecast function we might want to get.

1.1.1 The Loss Function

Loss functions assume a focal part in decision hypothesis [12]. Statistical decision hypothesis can be seen as an amusement against nature, instead of against other vital players [13]. In this diversion, we need to pick a move a to make from the set of admissible activities, i.e. the activity space A. This activity is hence judged in setting of the genuine result y ∈ Y, which is picked by nature. The loss function

$$L: Y \times A \rightarrow R+$$

gives a quantitative measure of the loss brought about from picking activity a when the genuine result winds up being y. The lower the loss caused the better. Loss functions can be utilized for estimating the nature of a parameter appraise.

1.1.2 The Risk Function

The loss function measures the accuracy of a prediction after the result is ob-served. At the time we make the prediction be that as it may, the genuine result is as yet obscure, and the loss caused is thus an irregular variable L(Y, a). It would subsequently be helpful to have an idea of an optimal action under vulnerability.

The risk of action a is characterized as the normal loss

$$R(a) = E[L(Y, a)]$$

The optimal action is characterized to be the risk minimizer $a*$ (Murphy, 2012),

$$a* = \arg\min R(a), \quad a \in A$$

1.1.3 The Model

The make predictions relying upon the input X, we will utilize a model

$$f : X \rightarrow A,$$

mapping each input $x \in X$ to a comparing prediction $a \in A$. Hence, for a given x, we would make the prediction

$$a = f(x),$$

furthermore, would thus acquire a loss of L(y, f(x)). The model is additionally alluded to as a theory, a prediction function, a decision function or a decision run the show.

2. Statistical Learning

The term"learning" is closely related to generalization. The goal in statistical learning is to find patterns in data that will generalize well to new, unobserved data. If one is able to find patterns that generalize well, one can make accurate predictions. This is indeed the goal in supervised learning, the part of statistical learning concerned with establishing the relationship between a response variable Y and a set of predictor variables X. Unsupervised learning, on the other hand, is concerned with finding patterns in data where there is no predefined response variable Y and the goal is to find structure in the set of variables X. In this chapter we will discuss some core concepts in supervised learning which provides the basis needed for discussing tree boosting.[14] Discusses the distinction between explanatory modeling and predictive modeling. In explanatory modeling we are interested in understanding the causal relationship between X and Y, whereas in predictive modeling we are inter-ested in predicting Y and our primary interest in the predictors X is to aid us in this goal. In this thesis, we will concern ourself with predictive modeling.

2.1. Boosting

Boosting refers to a class of learning algorithms that fit models by combining many simpler models [15]. These simpler models are typically referred to as base models and are learnt using a base learner or weak learner. These simpler models tend to have limited predictive ability, but when selected carefully using a boosting algorithm, they form a relatively more accurate model. This is sometimes referred to as an ensemble model as it can be viewed as an ensemble of base models. Another way to view it is that boosting algorithms are learning algorithms for fitting adaptive basis function models.

2.1.1 Tree Boosting Methods

Using trees as base models for boosting is a very popular choice. Seeing how trees have many benefits that boosted trees inherit while the predictive ability is greatly increased through boosting, this is perhaps not very surprising. The main drawback of boosted tree models compared to single tree models is that most of the interpretability is lost.

Boosted tree models can be viewed as adaptive basis function models of the form in given equations, where the basic functions are regression trees. Regression trees can however further be viewed as adaptive basis function models. We can thus collect the constant terms as

$$f(x) = \theta 0 + \sum_{m=1}^{M} \theta m \varphi m(x)$$

$$\theta 0 + \sum_{m=1}^{M} \theta m \sum_{j=1}^{Tm} w\tilde{}jmI(x \in Rjm)$$

$$\theta 0 + \sum_{m=1}^{M} \sum_{j=1}^{Tm} wjmI(x \in Rjm)$$

$$\theta 0 + \sum_{m=1}^{M} fm(x).$$

As seen from this, boosting tree models results in a sum of multiple trees $f_1, ..., f_M$. Boosted tree models are therefore also referred to as tree ensembles or additive tree models. Assume that we are interested in building a model for predicting the response variable $Y \in Y$ using a set of covariates $X = (X_1, ..., X_p) \in X$. Assume further that we have a data set at our disposal to solve the task at hand. The data set

$$D = \{(Y_1, X_1), (Y_2, X_2), ..., (Y_n, X_n)\}$$

is assumed to be an sample of size n from a joint distribution $P_{Y,X}$.

The response $Y \in Y$ is also referred to as the dependent variable or the output variable. When $Y \in Y$ can only take on a finite number of values or classes, i.e. |Y| is finite; we are dealing with a classification task. Otherwise, we are dealing with a regression task. The covariates $X = (X_1, ..., X_p) \in X$

are also referred to as the predictors, the explanatory variables, the features, the attributes the independent variables or the input variable.

3 Literature Survey

The ML– XGBoost is a prevailing statistical system of classification which recognizes nonlinear patterns inside datasets through missing qualities. It indicate essential potential expected for grouping patients among epilepsy base on the scholarly locale, preparing and side of the equator of their dialect showing. One subset, or else a point by point gathering of highlights, was the most predominant, implied for distinguish patients. The criticalness of this careful subset is conceivable given the intellectual alongside clinical clarification made through these patients.[1]

It's helpful to differentiate how LambdaRank and in addition LambdaMART overhaul their parameters. LambdaRank refresh each one the weights following each inquiry is inspected. The choices (part by the nodes) inside LambdaMART, on another side, are figure utilizing each one data that falls toward that hub, and additionally so LambdaMART refreshes only a little parameters on a period (to be specific, the gap esteems planned for the current leaf nodes), however utilizing each datum (since every xi arrives in a couple of leaf). This implies Lambda Shop is fit to choose parts alongside leaf esteems that may diminish the value for different questions, as long as the general utility increase.[3]

LIBLINEAR is easy along with easy-to-use open source package for huge linear classification. Experiments as well as analysis in Lin et al., Hsieh et al. along with Keerthi et al. (2008) conclude that solvers within LIBLINEAR execute well in practice with have good theoretical property. LIBLINEAR is still being enhanced by latest research results as well as suggestions as of users. The ultimate objective is to make easy knowledge with enormous data possible.[5]

Tree boosting methods contain empirically proven to be an extremely effective along with adaptable approach toward predictive modeling. For several years, MART has been a well-liked tree boosting method. In further recent years, a new tree boosting method through the name XGBoost has gain popularity in winning many machine learning competitions. In this theory, we compare these tree boosting methods as well as provided arguments intended for why XGBoost seems to win so a lot of competitions. We first show that XGBoost employ a special form of boosting than MART, whereas MART employ a form of gradient boosting, which is healthy known for its explanation as a gradient descent method within function space, we show that the boosting algorithm employ through XGBoost preserve be interpreted as Newton's method during function space. We so termed it Newton boosting. Furthermore, we compare the property of these boosting algorithms. We establish that gradient boosting is additional generally appropriate as it does not need the loss function to be severely convex. When appropriate however, Newton boosting is a influential alternative as it uses a higher-order estimate to the optimization problem to be solve at every boosting iteration. It also avoid the require of a line investigate step, which we can able to engage difficult calculations in a lot of situations.[6]

In this article, we show our result to Higgs Machine Learning opposition. We utilize a regularized edition of gradient boosting algorithm through a highly proficient implementation. We also obtain advantage of characteristic engineering base on physics to take out more information of the fundamental physical process. Experimental outcome on the match data express the accuracy as well as effectiveness of the technique proposed through this paper. One of the challenges for unit physics is the huge volume of the data. To deal with this problem, the new completion that deploys

XGBoost to a group of nodes is below development. The scalability will be additional improved along with it will be appropriate for much better data set. It is moreover interesting to discover other function classes that are extra physically significant.[7]

An extremely practical GPU-accelerated tree structure algorithm is devised as well as evaluate within the XGBoost documentation. The algorithm is build on top of proficient parallel primitives along with switches between two modes of process depending on tree strength. The 'interleaved' form of operation show that multi-scan as well as multi-reduce operations through a limited amount of buckets can be used to avoid costly sorting operations at tree depths under six.[8]

The most important objective of this theory has been to afford understanding lying on how to approach a supervised learning prognostic problem as well as illustrate it by the tree boosting method. To achieve this aim, an clarification of a supervised problem has been provide as well as a analysis of the dissimilar tree methods developed since this method was introduce in Breimanet al. (1984). Reviewing the tree method evolution helps to recognize the current tuning parameters technique. Tree boosting along with the XGBoost implementation is the present state-of-the-art predicting technique for many problems; a obvious signal of its usefulness it the fact that is the mainly used algorithm for data after that competitions Chen and Guestrin (2016). In the scope of competition, algorithms require to take into description deep learning LeCun et al. (2015), when the features are text or else images.[9]

All of the useful algorithms were capable to achieve the rest task, provided that several predictive value, when it came to order contracts through their churn probability. For the use validation method, XGBoost prove to be the mainly effective one, through RF and ERT exhibit similar performance as well as CART being the worst. It was probable that the assembly method would better a single decision tree through the CART algorithm which was the case. This is in line among existing literature as well as the theory following the applied modeling technique. When it came to analyze the outcome, it was exciting to note how much produce early false prediction can have as well as how early these are trapped through the models. Now, every model are punish severely for false early on prediction, even though lots of the variables will not modify much over time as of their design. In the current validation method, even if the models are properly predicting lots of months ahead that a service convention is likely to be cancelled, such prediction will be penalize, no issue what the conclusion.[11]

4 Proposed Work

Here I have utilized three datasets which are prescriber-info, overdoses and opioids. The primary dataset prescriber info demonstrates the detail of the pharmaceutical which is recommended by various prescriber. The second one is an overdose which demonstrates the impacts of the overdose utilization of the solution on various zones and the last one is about the opiods that is fundamentally the medications contained in the prescription.

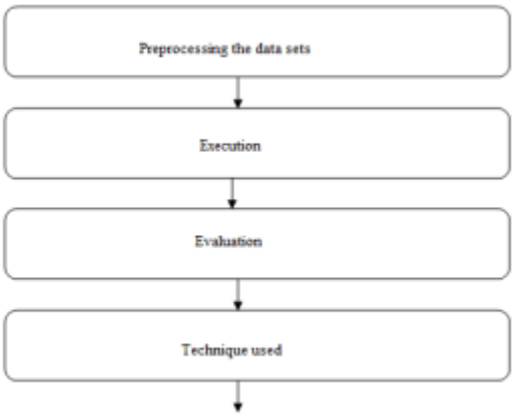

Figure 1 show that the inspection which is think about, none of the datasets required pre-processing as each dataset contained just numeric data and

none of the datasets had any missing qualities. Thus, no pre-processing was required. The XGBoost calculation has been executed in python in an i3 framework having 4 cores. The code for the execution of the calculation on all the four datasets has been made accessible in the GitHub archive. Two strategies were decided with the end goal of assessment of the models viz. train and test sets technique and the k-fold cross-validation methods. The train and test sets show is one of the least difficult models accessible where we split the whole dataset into training set and testing set. This strategy is especially helpful when the dataset included is vast in light of the fact that we isolate the trained and testing sets already. This strategy is imperative when the calculation included is moderate in the preparation procedure. In this effort we have exposed how we can develop the xgboost classifier precision.

Gradient boosting algorithm was developed for very high predictive capability. Still its adoption was very limited because the algorithm requires one decision tree to be created at a time in order to minimize the errors of all previous trees in the model. So it took a large amount of time to train even those models that were small in size. Then come a new algorithm called eXtreme Gradient Boosting (XGBoost) which changed the way gradient boosting was done. In XGBoost, individual trees are created using multiple cores and data is organized in order to minimize the lookup times. This decreased the training time of models which in turn increased the performance. This research study strives to create a quantitative comparison of the accuracy and speed of XGBoost with its improved version algorithm in multi-threaded single-system mode and Gradient Boosting with different datasets.

The concept of boosting came to limelight when it was examined whether a "weak learner" could be made a "better learner" by using some kind of modifications. From statistics point of view, this process was similar to creating a "good hypothesis" from a relatively "poor hypothesis". According to Jason Brownlee, , a poor learner or a "weak hypothesis" is a model whose performance is slightly better than random chance. Hypothesis boosting involves the idea of filtering the observations. Those observations which the weak learner can handle is left as it is and those observations that the weak learner cannot handle are focused on. "The idea is to use the weak learning method several times to get a succession of hypotheses, each one refocused on the examples that the previous ones found difficult and misclassified. … Note, however, it is not obvious at all how this can be done…".

The concept of gradient boosting involves basically three steps. First, a proper differentiable loss function should be identified that is suitable for the given problem. One benefit of the gradient boosting model is that for different loss functions, new algorithms are not required to be derived; it is enough that a suitable loss function be chosen and then incorporated with the gradient boosting framework. Second, a weak learner is created to make the predictions. In gradient boosting a decision tree is chosen as a weak learner. Specifically, regression trees are used that produces real value output for splits and whose output can be added together, allowing subsequent outputs of different models to be added. This approach enables the improvement of the residuals in the predictions leading to more precise predictions. The trees are created in a greedy manner and often certain constraints are imposed in order to ensure that the weak learners continue to be weak learners and still the trees can be created using a greedy approach. Third, creation of an additive model to add up the predictions of the weak learners so as to reduce the loss function. This process of adding the trees happens one at a time. The output produced in the new tree is then added to the output of the pre-

existing sequence of trees in order to improve the final output of the model. This process stops once the proper optimized value for the loss function is reached.

6 Algorithm and its flow chart

XGBoost has speedily become amongst the most accepted methods use for classification into machine learning. It is a performance over the gradient boosting.

Algorithm 1:

Import the library like pandas, numpy and other important libraries

```
1.  Read the dataset overdose
ods = pd.read_csv('overdoses.csv')
2.  import xgboost as xgb
3.  def cv (alg,X,y):
metrics = ['auc', 'map']
xgtrain = xgb.DMatrix(X,y)
        param = alg.get_xgb_params()
4.      cvresult = xgb.cv(param,
5.                  xgtrain,
6.
num_boost_round=alg.get_params()['n_estimators'],
7.                  nfold=7,
8.                  metrics=metrics,
9.                  early_stopping_rounds=50)
10.
alg.set_params(n_estimators=cvresult.shape[0])
11.     #Predict training set:
12.     alg.fit(X,y,eval_metric=metrics)
13.     # Show features, rated by fscore
14.     features = alg.booster().get_fscore()
15.     feat_imp =
pd.Series(features).sort_values(ascending=False)
16.     feat_imp[:50].plot(kind='bar',
title='Feature Importances', figsize=(9,6))
17.     plt.ylabel('Feature Importance Score')
18.     # sort for human readability
19.     import operator
20.     sorted_features = sorted(features.items(),
key=operator.itemgetter(1))
21.     print('features by importance',
sorted_features)
22.     return features, cvresult
23.     dentists = ps[ps['Specialty'] == 'Dentist']
24.     dentists['Gender'] =
pd.get_dummies(dentists['Gender'])
25.     target = 'Opioid.Prescriber'
26.     X = dentists.drop(['NPI', 'Specialty',
'Credentials', 'State', target], 1)
27.     y = dentists[target]
28.     (X.shape, y.shape)
29.     print("Opiod Prescriber")
30.     print("Accuracy of Xgboost Classifier")
```

Algorithm 2:

```
1.  from sklearn.model_selection import
train_test_split
2.  alg = xgb.XGBClassifier(
i.  learning_rate =0.1,
ii. n_estimators=500,
iii. max_depth=4,
iv. min_child_weight=2,
v.  gamma=0,
vi. subsample=0.8,
vii. colsample_bytree=0.8,
viii. nthread=4,
ix. objective="binary:logistic",
x.  scale_pos_weight=1,
xi. seed=27)
3.  X_train, X_test, y_train, y_test =
train_test_split( X, y, test_size=0.33,
random_state=42)
4.  features, cvresults = cv(alg,X_train,y_train)
5.  cvresults[-3:]
dentists['Gender'] =
pd.get_dummies(dentists['Gender'])
```

XGBoost also has gradient boosting at its core. However, the difference between simple gradient boosting algorithm and XGBoost algorithm is that unlike in gradient boosting, the process of addition of the weak learners does not happen one after the other; it takes a multi-threaded approach whereby

proper utilization of the CPU core of the machine are utilized, leading to greater speed and performance. Apart from that, there is sparse aware implementation which also involves automatic handling of missing data values, then block structure to support the parallelization of tree construction, and the process of continued training so that one can further boost an already fitted model on new data. It is to be noted that XGBoost has been seen to dominate structured or tabular datasets on classification and regression and predictive modeling problems.

6.1 Result Analysis

In this section we show the comparison on the basis of accuracy between xgboost and improved XGBoost. The whole program is implemented in Python 3.6 which is one of the latest algorithms.

6.1.1 Opiod Prescriber

Here, we used the opiod prescriber dataset with various features by importance for calculate the accuracy using XGBoost classifier as well as Improved XGBoost classifier.

- **Features by importance**

[('CEPHALEXIN', 4), ('AZITHROMYCIN', 5), ('AMOX.TR.POTASSIUM.CLAVULANATE', 10), ('IBUPROFEN', 15), ('OXYCODONE.ACETAMINOPHEN', 16), ('CHLORHEXIDINE.GLUCONATE', 19), ('CLINDAMYCIN.HCL', 20), ('ACETAMINOPHEN.CODEINE', 30), ('HYDROCODONE.ACETAMINOPHEN', 33), ('AMOXICILLIN', 37)]

- **Improved Xgboost Classifier**

C:\Users\welcome\Anaconda3\lib\site-packages\sklearn\preprocessing\label.py:151:

Deprecation Warning: The truth value of an empty array is ambiguous. Returning False, but in future this will result in an error. Use `array.size > 0` to check that an array is not empty.

if diff:

- **Features by importance**

[('Gender', 1), ('CEPHALEXIN', 1), ('AZITHROMYCIN', 3), ('CLINDAMYCIN.HCL', 4), ('IBUPROFEN', 6), ('TRAMADOL.HCL', 8), ('CHLORHEXIDINE.GLUCONATE', 10), ('AMOX.TR.POTASSIUM.CLAVULANATE', 11), ('OXYCODONE.ACETAMINOPHEN', 12), ('AMOXICILLIN', 14), ('HYDROCODONE.ACETAMINOPHEN', 21), ('ACETAMINOPHEN.CODEINE', 22)]

C:\Users\welcome\Anaconda3\lib\site-packages\sklearn\preprocessing\label.py:151:

Deprecation Warning: The truth value of an empty array is ambiguous. Returning False, but in future this will result in an error. Use `array.size > 0` to check that an array is not empty.

if diff:

6.1.2 Non-Opiod Features

Here, we used the opiod prescriber dataset with various non-opiod features by importance for calculate the accuracy using XGBoost classifier as well as Improved XGBoost classifier.

- **Features by importance**

[('DOXYCYCLINE.HYCLATE', 11), ('AZITHROMYCIN', 15), ('CEPHALEXIN', 43), ('Gender', 47), ('IBUPROFEN', 112), ('CHLORHEXIDINE.GLUCONATE', 129), ('AMOXICILLIN', 385), ('State', 472)]

- **Improved Xgboost Classifier**

C:\Users\welcome\Anaconda3\lib\site-packages\sklearn\preprocessing\label.py:151:

Deprecation Warning: The truth value of an empty array is ambiguous. Returning False, but in future this will result in an error. Use `array.size > 0` to check that an array is not empty.

if diff:

- **Features by importance**

[('AZITHROMYCIN', 10), ('DOXYCYCLINE.HYCLATE', 11), ('CEPHALEXIN', 31), ('Gender', 37), ('CHLORHEXIDINE.GLUCONATE', 98), ('IBUPROFEN', 110), ('AMOXICILLIN', 262), ('State', 351)]

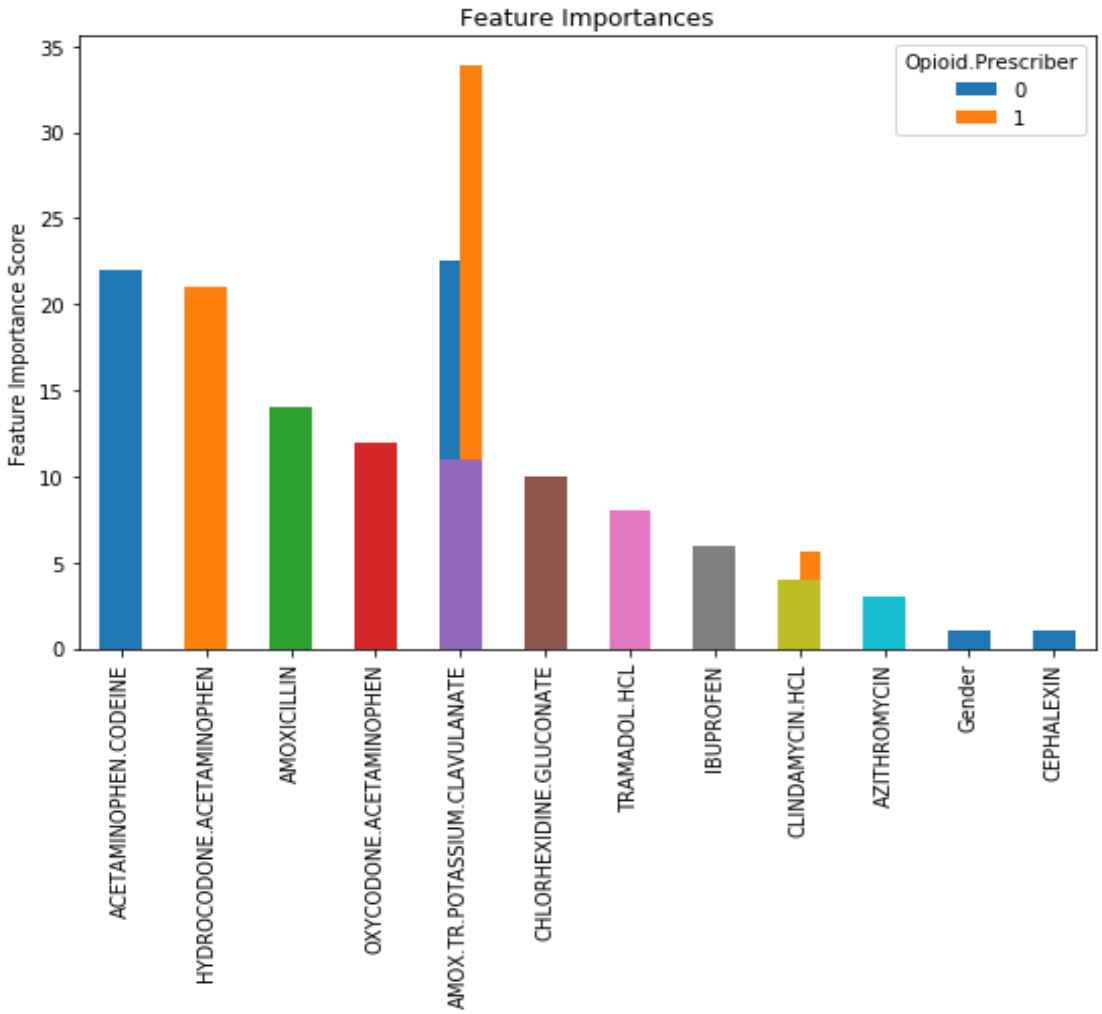

Figure 1 Feature Importance Score

In the above given figure 1, we can predict quite easily that whether a doctor is an opiate prescriber based on their prescription history. It make sense that prescriber label will be depend on prescriber history, but here we will try to detect doctors early who may prescribe the opiate later. Now these are the features which are showing how they are correlated to each other and with prescribed label.

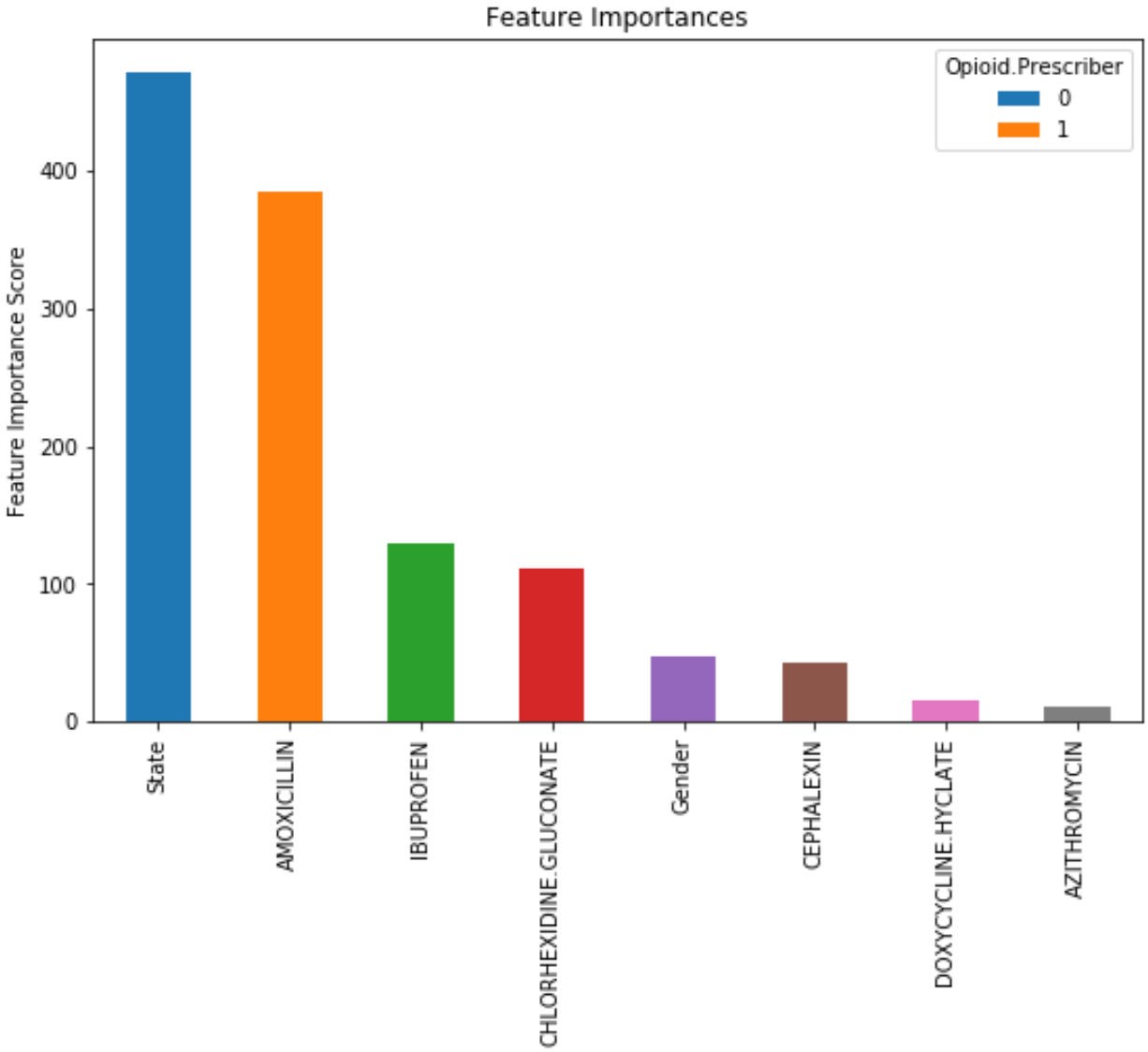

Figure 2 Feature Importance Score with different parameters

In the above figure 2, it is clearly mentioned that state and drugs are the features which are very much correlated for predicting the label as the non opiod prescriber and opiod prescriber respectively. According to above given figure amoxicillin is an antibiotic and it is mostly prescribed by the opiod prescriber. Here, shows that the graphs between various feature important score with different – different parameters groups, parameter examination with feature parameter score in terms of percentile of opiod prescriber which is shows that the existence of its quantity. Here, 0 indicate the presence of opiod prescriber and 1 indicate the absence of opiod prescriber.

		XG Boosting	**Improved XG Boosting**
Opiod Prescriber	Accuracy	0.9221	0.9242
	AUC Score (train)	0.921397	0.925079

Non- Opiod Prescriber	Accuracy	0.7294	0.7305
	AUC Score (test)	0.762865	0.768115

Table 1 Comparitive study of XGBoost and Improved XGBoost Classifier

Here, table 1 shows that the comparison with XGBoost and Improved XGBoost Classifiers in terms of Accuracy and AUC Score for train and test datasets, for Opiod Prescriber and Non- Opiod Prescriber.

7 Conclusions and Future Work

In this paper, we described the lessons we learnt when building XGBoost, a scalable tree boosting system that is widely used by data scientists and provides state-of-the-art results on many problems. We show the comparison on the basis of accuracy between xgboost and improved xgboost. The whole program is implemented in Python 3.6 which is one of the latest algorithm.

Finally, gradient boosting has proven many times to be an effective prediction algorithm for both classification and regression tasks. By selecting the number of components included in the model, we can easily control the so-called bias varianc trade-off in the estimation.

References

[1] Chen, Tianqi, and Carlos Guestrin. "Xgboost: A scalable tree boosting system." In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, pp. 785-794. ACM, 2016.

[2] J. Friedman. Greedy function approximation: a gradient boosting machine. Annals of Statistics, 29(5):1189,1232, 2001.

[3] Baciu M, Perrone-Bertolotti M (2015) What do patients with epilepsy tell us about language dynamics? A review of fMRI studies. Rev Neurosci 26(3):323–341

[4] L. Breiman. Random forests. Maching Learning, 45(1):5{32, Oct. 2001.

[5] C. Burges. From ranknet to lambdarank to lambdamart: An overview. Learning, 11:23{581, 2010.

[6] Josse G, Tzourio-Mazoyer N (2004) Hemispheric specialization for language. Brain Res Rev 44(1):1–12

[7] Dijkstra KK, Ferrier CH (2013) Patterns and predictors of atypical language representation in epilepsy. J Neurol Neurosur Psychiatry. doi:10.1136/jnnp-2012-303141

[8] T. Chen, S. Singh, B. Taskar, and C. Guestrin. Efficient second-order gradient boosting for conditional random fields. In Proceeding of 18th Artificial Intelligence and Statistics Conference (AISTATS'15), volume 1, 2015.

[9] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. Journal of Machine Learning Research, 9:1871{1874, 2008.

[10] J. Friedman. Greedy function approximation: a gradient boosting machine. Annals of Statistics, 29(5):1189{1232, 2001.

[11] Billingsley RL, McAndrews MP, Crawley AP, Mikulis DJ (2001) Functional MRI of phonological and semantic processing in temporal lobe epilepsy. Brain 124(6):1218.

[12] Young, G. and Smith, R. “Essentials of Statistical Inference. Cambridge Se-ries in Statistical and Probabilistic Mathematics”, Cambridge University Press, 2005.

[13] Murphy, K. P. “Machine Learning: A Probabilistic Perspective” The MIT Press, 2012.

[14] Shmueli, Galit. "To explain or to predict?." Statistical science 25.3 (2010): 289-310.

[15] Freund, Y. and Schapire, R. E., “Experiments with a new boosting algorithm”, In Saitta, L., editor, Proceedings of the Thirteenth International Conference on Machine Learning (ICML 1996), pages 148–156. Morgan Kaufmann, 1996.