Final Report: Development of Patient No-show Predictive Models using RIPPER and Hoeffding Tree algorithms

Muneeb Shahid
mushahid@uiowa.edu
University of Iowa
Iowa City, Iowa, USA

ACM Reference Format:
Muneeb Shahid. 2020. Final Report: Development of Patient No-show Predictive Models using RIPPER and Hoeffding Tree algorithms. In *Iowa City '21: Machine Learning, May 07, 2021, Iowa City, IA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Patient's who do not show up on appointed time are termed as no-shows. Patient no-shows negatively affects healthcare system in terms of expenditures and management [1]. Especially, after COVID, there is a lot of burden and pressure on healthcare staff to achieve operational efficiency. One possible way to help the healthcare staff is the development of patient scheduling systems. The development of these systems is an ongoing research topic. Effectiveness of such systems can be improved by creating a predictive model that would determine the probability of patient no-show on given appointment date and time. However, various factors needs to be considered for developing patient no-show predictive model. Nonetheless, the availability of Electronic Health Records (EHR) and progress in machine learning algorithms have made accurate predictions possible [3]. The objective of these models is to maximize the resource utilization of healthcare system [7]. However, all the work that has been done so far to solve this problem are more focused towards discovering ways of improving accuracy of predicting patient no-show. None of these works evaluates the models in terms of interpret-ability and computational efficiency. In this project, we are going to apply methods that would help us in achieving the goals of interpret-ability and computational efficiency.

Interpret-ability of machine learning models makes comprehension and explanation of the predictions easier for the end-users. Especially, in the field of healthcare, lack of interpret-ability in machine learning models discourages healthcare staff going towards machine learning adoption, as they need to make sensitive decisions and they can not blindly rely on the models. High interpret-able models build trust between the model and the healthcare staff, such models allow healthcare experts to make reasonable and data-driven decisions to provide personalized decisions that can ultimately lead to high quality healthcare service, which is suggesting appropriate appointment times to patient in our case. [9].

Moreover, several appointments are scheduled on daily basis therefore, increasing the demand of memory and computational efficient machine learning algorithms [6]. Especially, when the new data arrives at a higher rate than they can be used for training model then percentage of unused data increases without any bound. Also, when the models are deployed in a system with limited resources then increase in data size causes memory and computation issues. Regarding optimization, there are many features available in EHR and most of these features might not be useful for patient no-show prediction. In a nutshell, optimal features subset needs to be selected that will contain relevant features. [1].

For patient no-show prediction, in this project, I will use RIPPER (Repeated Incremental Pruning to Produce Error Reduction) and Hoeffding Tree algorithm for prediction and SHAP (Shapley Additive explanations) for features selection. RIPPER is selected because of its compactness and its ability to facilitate comprehension. Hoeffding Tree is selected to address memory and computational efficiency issues, it has the ability to handle large data-sets in reasonable time [5]. The problem of model interpretation, has encouraged me to use SHAP for feature selection as it provides a model-agnostic approach to interpretation [4]. Despite the fact that I am already using interpret-able machine learning model for prediction, but, in case of bad performance of this algorithm in future, I might be encouraged to use more complex models, in that case, SHAP would still be helpful in achieving interpret-ability. [4].

2 RELATED WORK

There are many studies related to the development and validation of patient no-show predictive models. [1] proposes new wrapper methods (methods that evaluates every feature subset using a classifier) for feature selection. These wrapper methods are based on Self-Adaptive Cohort Intelligence (OSACI). The performance of these methods were tested on low dimensional data-set (21 features) and importance of the selected features were not studied. Some of the studies focuses on developing a probabilistic patient no-show model. [3] presents a deterministic integer linear programming model for offline scheduling of patients (scheduling after checking the availability of doctor's and resources) and [7] presents a stochastic overbooking model which compensates for patient no-shows. Interpret-ability and computational time of these models were not considered. The computational time of these methods appears to grow exponentially with the increase in size of the data-set. Accurate determination of appointment scheduling window is one of the popular use case of patient no-show predictive models, in that regard, [6] developed an optimization model. The objective

Iowa City '21, May 07, 2021, Iowa City, IA

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

functions of this optimization model are defined to minimize the cost of patient waiting time and physician idle time. The data-set used for this experiment was very limited (one month only).

Specific to the prediction of no-show appointments, various machine learning algorithms have been studied and applied. [5] used step-wise naive and mixed-effect logistic regression algorithm. This study neglected the missing values in the data-set their inclusion could alter the results of the predictive models. [4] compared the performance of several machine learning algorithms (Ada-Boost, Logistic Regression, Naive Bayes, SVM, SGD, Decision Tree Classifier, Extra Trees Classifier, Random Forest Classifier, and XGBoost) for this task. This study does discuss interpret-ability of the algorithms but fails to mention ways to improve interpret-ability of these algorithms and computational time of these algorithms were missed from the discussion.

Regarding interpret-able machine learning models, [9] gives an overview of different approaches for achieving interpret-ability in predictive models This study explains ways to address interpret-ability issues but in a general manner not specific to patient no show prediction problem. [8] conducts a comparative analysis of four machine learning models (logistic regression, random forests, gradient boosting machine, and ANN) in predicting patient punctuality. This comparison is done with respect to interpret-ability, performance, and computational time. This study aligns with our project but it is more focused on studying patient attributes and past behaviors while, our study is focused on appointment related attributes only.

3 METHODS

In this project, I have adopted a systematic and structured approach by dividing the research problem into different steps: (i) Feature engineering (ii) Overcoming Class Imbalance Problem (iii) Feature Selection (iv) Predictive Modeling using interpret-able and computationally efficient machine learning algorithms.

3.1 Feature Engineering

Based on my literature survey, there are few features that are not readily available in Electronic Health Records (EHR) and these features are important for predicting patient no show model. For this purpose, I wanted to generate features based on the existing features. Following is the list of features that were generated for this project:

- Scheduled Day of Week
- Scheduled Month
- Appointment Day of Week
- Appointment Month
- Difference in Days between scheduled and appointment date
- Same Day Appointment

3.2 Class Imbalance Problem

While, working on EHR, we need to be vary of imbalance class problem as well. Number of no-show records will be significantly lesser than number of show records. In that case, we need to cope with the imbalance between majority and minority classes before moving towards building a predictive model. If we do not address this problem then our model will be biased towards the majority class and treat the minority class as noise in the data. I have applied Synthetic Minority Oversampling Technique (SMOTE) to solve this problem. It is an oversampling technique in which synthetic samples are generated for minority class with the help of feature space.

SMOTE works by picking up a random instance from the minority class at first. Then K number of instances are found that are closest to that instance. Next, from these K nearest neighbors, a random instance is selected to create a synthetic instance based on these two selected instances. This method is effective in a sense that instances for minority class are created by combining instances that are close to each other in feature space. However, one drawback of this approach is that new instances are created without considering majority class instances. These new instances could deviate from the ground truth especially, if there are too many instances that overlap between majority and minority class. In this project, we did suffer from this drawback which will be discussed later.

3.3 Feature Selection

EHR comes with the large number of features, to avoid over-fitting and to reduce computational time, feature selection task becomes important. There are many ways to perform this task however, I will consider Shapley Additive Explanations (SHAP) in this project. The objective of the SHAP is to explain contribution of each feature in performing prediction [4]. Shapley values are calculated for each feature, and these values are computed by calculating average marginal contribution of each feature to each prediction across all permutations.

For example, if there are three features in the dataset, lets call them A, B and C and we want to calculate Shapley values for feature A. Then for every instance, the value of feature B and C are kept intact and value of feature A is randomly replaced by any other value in its own distribution, then the model predicts the target outcome for this simulated instance. After that, the difference between the average prediction across all instances and this simulated instance is calculated. This difference is attributed to randomly sampled feature A. Similarly, this process is repeated for all combinations of feature B and feature C. Finally, SHAP value for feature A is calculated by taking (weighted) average of all the marginal contributions to all possible combination.

Shapley values helps to achieve following objectives:

- Collective SHAP values helps in understanding the relation between predictor and target variable.
- Individual SHAP values helps in understanding the importance of each feature towards a particular prediction.

With these objectives, I will be able to identify important features and use them for predictive modeling.

Note: There is no direct link between SHAP feature method and the predictive models I have used for this project. I wanted to study how SHAP values can help us achieving interpret-ability in the feature selection phase regardless of type of models we use for predictive modeling.

3.4 Predictive Modeling

I have built predictive models to predict no-show appointments using RIPPER (Repeated Incremental Pruning to Produce Error Reduction) and Hoeffding algorithms. Additionally, the decision tree based models are popular interpret-able models and they have been used to solve patient no-show problem. But decision tree models, have a tendency to over-fit and are fragile as well, small change in training set can completely change rules of decision trees, even though there are methods to avoid these problems in decision-tree based models but these methods have their own downsides. Anyways, I wanted to study performance of rule-sets based algorithms because they produce simple and compact models and RIPPER Is one of the rule-sets model, that I used for this project.

3.4.1 RIPPER. RIPPER is a propositional rule learning algorithm. It was built as an optimized version of incremental reduced error pruning algorithm (IREP) [2].

RIPPER includes, three stages of rule processing: building, optimization, and clean-up. In the first stage, a training data-set is divided into growing and pruning sets. Rules are constructed on basis of a growing set. Later these rules are pruned with the help of pruning set. Second step is the optimization stage, in this stage all rules are reconsidered to reduce the error on entire data-set, value of K specifies the number of optimizations to run. Last stage in RIPPER is clean-up, it calculates complexity of rule-set in terms of description length, if the rule-sets complexity increases description length beyond the given threshold, then that rule-set is deleted, this helps us to avoid over-fitting as well.

Rule-set is simply dis-junctions of conjunctions. Rule-sets are derived for positive class only which is for no-show value of 1 in our case, any condition outside this rule-set will be predicted as negative class. RIPPER starts with empty rule and continues to add rules until there are no negative classes against that rule, for evaluation, it uses Information Gain, similar to decision trees.

Following are the advantages of using RIPPER algorithm:

- It performs well with datasets with imbalanced class distributions.
- Works well with noisy datasets as it uses validation set to prevent model overfitting.
- Since, it is a rule based learning algorithm, its predictions are easier to interpret as well.

As we are dealing with two classes (no-show and not no-show) so, following is the working of RIPPER in steps:

- Initially this algorithms starts with 0 rules. It splits the training data-set into a grow set and a prune set. Size of the grow set is 2/3 of the training set and size of prune set is the rest 1/3 of the training set.
- It applies information gain criterion on grow set to build an over-fitted model with rules.
- It applies pruning operator on prune set to remove the redundant rules. By doing so, the algorithm selects final set of rules that maximizes the value of following equation:

$$M(Rule, PositiveExamples, NegativeExamples) = \frac{p + (N - n)}{P + N}$$

Where, P is the total number of positive examples (minority class) in prune set, N is the number of negative examples (majority class), p is the number of positive instances covered by the rule and n is the number of negative instances covered by the rule.

- It continues to perform pruning steps until further pruning results in increased error on prune set.

3.4.2 Hoeffding Tree. Hoeffding Tree is also known as Very Fast Decision Tree because of its ability to perform computations in a very small amount of constant time per example. Hoeffding Tree is different from any other tree based classification methods in a sense that, other tree methods need all of the training data available in memory, while Hoeffding Tree needs to parse large data-set only once and build tree as the data comes and it does not even require all of the data to be stored in main memory.

This method learns from a stream of data in just one pass and it assumes that the data distribution will not change in upcoming streams hence, relatively small subset of data is sufficient enough to find a best attribute for a tree node. Based on this assumption, on receiving first batch of data, it selects a test for root node. Then successive batches are used to select test for next node in the tree, and so on.

There is a Hoeffding Bound based on which this method decides whether the test at the current node will pass or not. The purpose of these tests is to evaluate whether sufficient statistical evidence exist for an optimal splitting feature. However, if the test at a given node fails because other alternative attribute has more statistical evidence then a new sub tree is grown with the alternative splitting test at its root. If this new test yields better results then it replaces the old one [2].

4 EXPERIMENTATION AND EVALUATION

This section contains steps that were taken for the development of predictive models.

4.1 Code Development

I have written code in Google Colab environment (Notebook). All of the code is written using Python programming language, for machine learning I have used scikit-learn, scikit-multiflow libraries and for data visualization I have used seaborn and matplotlib packages.

4.2 Dataset

I have downloaded data medicalInvestigate Medical Appointment No Shows data-set from Kaggle for this experiment. Following are the statistics of the data-set before pre-processing:

Table 1: Dataset Statistics

Property	Value
No. of Features	14
No. of Records	110,528
percentage of no-shows	20

Following is the number of Yes (1) and No values (0) in 'No-show' / target feature in the data-set.

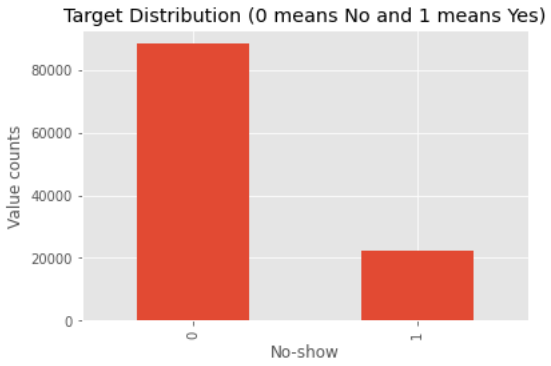

Figure 1: Class Distribution

4.3 Data Pre-processing

In this stage of experimentation, I have used pandas and numpy to pre-process the dataset and imblearn for class balancing.

First step in the data pre-proprocessing stage is to generate features as mentioned in the feature engineering section. After generating features, it was necessary to hot encode features because besides one or two features all the features are categorical and library that we are using to train machine learning models does not work with categorical features (with more than two categories). In hot encoding, each value in the categorical feature is converted into a separate boolean feature.

Next, I had to balance the number of instances for each class because there is huge disparity between number of no-show and non no-show examples in the data-set, which is evident from the bar graph displayed in data section. There were 80 percent non no-show examples and 20 percent no-show examples.

Following is the data-set statistics after hot enconding and class balancing:

Table 2: Dataset Statistics after hot encoding and class balancing

Property	Value
No. of Features	118
No. of Records	176,416

4.4 Dimensionality Reduction

In the pre-processing step, the number of features increased to 118 from 20 and baed on the literature survey there are many features that might not be useful for the prediction but, we can not operate on this assumption. To study the contribution of each feature towards the class label, I have used SHAP method with XGBOOST model. Figure 2 displays the top 20 important features selected by SHAP based on their Shapley values.

Visualization in Figure 2 is very helpful as it gives a global level interpretation and tells how much each feature contributes either positively or negatively to the target variable. The y-axis on the right side indicates the respective feature value being low vs high

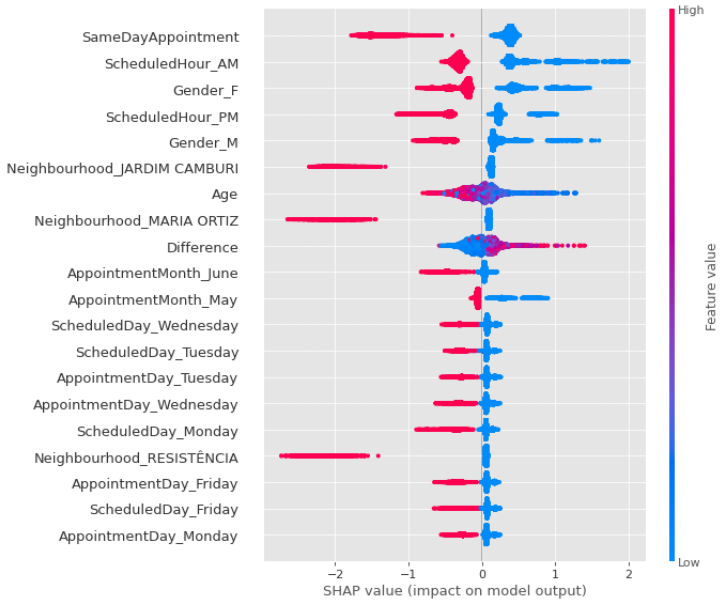

Figure 2: SHAP Feature Importance Plot

and each dot represents one instance in the data with the SHAP value. For example, if we look at most important feature i.e SameDayAppointment, its SHAP values tells us that higher the value of SameDayAppointment which is 1 as it is Boolean variable lower will be the value of prediction, in other words, if patients are scheduled on the same appointment date then it is highly likely that patient will show up and vice versa. Similarly, if we look at SheduledHourAM feature, which tells us whether the appointment was scheduled during AM hours or not. We can tell that if the appointments are scheduled during AM hours then it is highly likely that patient will show up and vice versa. This is how SHAP values helps us interpret contribution of each feature with respect to output. Figure 3 displays another visualization that helps us to interpret relative importance of each variable.

With the help of SHAP values we can interpret feature values from different perspective, in the visualization displayed in Figure 4, we can see which features pushed the prediction towards higher values and which ones pushed the prediction towards low values for one particular instance.

Anyways, with the help of SHAP, I selected important variables having feature importance score >=0.1. As a result, I got 25 important features, now the number of features are reduced to 25 from 118.

4.5 Building Predictive Models

I have used 5 Cross Folds Validation method using GridSearchCV module of scikit-learn to train the models and to select best parameters for these models. Before training the data, I split the data by 80 percent and 20 percent where 80 percent data is used for training and rest for testing data. I have separated testing data for evaluation and used training data for 5 cross fold validation. For building

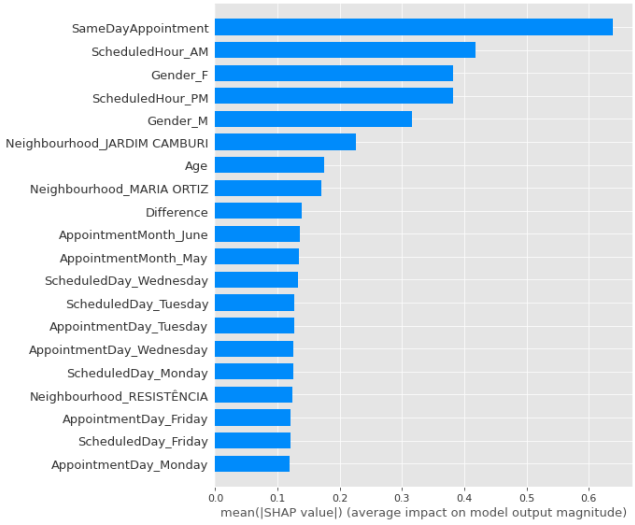

Figure 3: SHAP Feature Importance Horizontal Bar Plot

Figure 4: SHAP Feature Importance Local Interpreation

RIPPER model, I have used wittgenstein package and for Hoeffding Tree, I have used scikit-multiflow HoeffdingTreeClassifier.

4.5.1 RIPPER. The best parameters values I got was K = 2, and prune size = 0.33. Value of K specifies the number of optimizations to run. Following is the Final Ruleset generated after training RIPPER model with the training data:

FINAL RULESET:

[[SameDayAppointment=0 AND SMSreceived=0 AND ScheduledMonthMay=0 AND AppointmentMonthJune=0 AND ScheduledHourAM=0 AND ScheduledHourPM=0] OR [SameDayAppointment=0 AND SMSreceived=0 AND ScheduledMonthMay=0 AND AppointmentMonthJune=0 AND ScheduledDayWednesday=0 AND ScheduledDayFriday=0 AND ScheduledDayTuesday=0 AND ScheduledDayThursday=0 AND ScheduledDayMonday=0 AND AppointmentDayFriday=0 AND AppointmentDayThursday=0 AND AppointmentDayMonday=0] OR [SameDayAppointment=0 AND GenderM=0 AND GenderF=0] OR [SameDayAppointment=0 AND SMSreceived=0 AND AppointmentDayThursday=0 AND AppointmentDayWednesday=0 AND AppointmentDayFriday=0 AND AppointmentDayTuesday=0 AND AppointmentDayMonday=0] OR [SameDayAppointment=0 AND ScheduledHourAM=0 AND ScheduledHourPM=0] OR [SameDayAppointment=0 AND AppointmentMonthJune=0 AND AppointmentMonthMay=0 AND AppointmentDayFriday=0] OR [SameDayAppointment=0 AND SMSreceived=0 AND ScheduledMonthMay=0 AND AppointmentMonthJune=0 AND ScheduledDayWednesday=0 AND AppointmentDayMonday=1 AND ScheduledDayTuesday=0 AND ScheduledDayFriday=0 AND ScheduledDayThursday=0 AND ScheduledDayMonday=0 AND GenderF=1] OR [SameDayAppointment=0 AND SMSreceived=0 AND ScheduledMonthMay=0 AND AppointmentMonthJune=0 AND ScheduledDayWednesday=0 AND ScheduledDayTuesday=0 AND ScheduledDayThursday=0 AND ScheduledDayMonday=0 AND ScheduledDayFriday=0] OR [SameDayAppointment=0 AND SMSreceived=0 AND ScheduledMonthMay=0 AND AppointmentMonthJune=0 AND AppointmentDayMonday=1 AND ScheduledDayMonday=1 AND Difference=8-14 AND ScheduledHourAM=0 AND GenderF=1 AND Age=54-63] OR [SameDayAppointment=0 AND AppointmentMonthJune=0 AND ScheduledMonthMay=0 AND SMSreceived=0 AND Age=14-22 AND ScheduledHourAM=0 AND Difference=8-14 AND AppointmentDayThursday=0 AND AppointmentDayMonday=0]]

First clause of the rule-set tells that if the appointment is not scheduled on same day and patient didn't get SMS notification and appointment was scheduled outside the month of may and appointment is not on month of June and appointment scheduled outside AM and PM hours then it means the patient will not show up at appointed time. One thing to observe here is that the last two conditions in first clause are conflicting, it tells scheduled time outside AM and PM as well, which is not possible, this tells me that some instances have been added during class balancing process by SMOTE that are invalid and which would affect the test accuracy of the model. This is one of the examples, how interpret-ability helps in debugging the problems with the model.

4.5.2 Hoeffding Tree. The best parameters values I got was leaf prediction = naive bayes adaptive, and splitting criteria = Gini index. Following is the figure that tells the percentage reduction in number of samples required for Hoeffding Tree to converge:

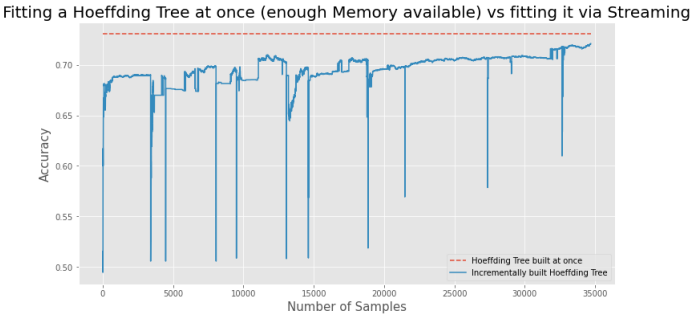

Figure 5: Hoeffding Tree Convergence

In the figure, orange line represents final accuracy obtained when Hoeffding tree was trained with full training dataset consisting of around 176 thousand records, and blue line represents the increase and decrease in accuracy when data is fed to the Hoeffding tree in batches, we can see that Hoeffding Tree needed only around 35,000 samples to converge to same accuracy as obtained by training on full training dataset. In this way, with the help of Hoeffding Tree we can reduce training time and memory requirements.

4.6 Evaluation

Following are the scores I obtained for both of these models, I also trained Random Forest and Adaboost Model, to use them as benchmark as based on literature survey for patient no-show prediction task these two models yielded better results.

Table 3: Evaluation Metrics Scores

MODEL	VALIDATION SCORE	ACCURACY
RIPPER	0.8	0.65
HOEFFDING TREE	0.7	0.73
RANDOM FOREST	0.8	0.83
ADABOOST	0.7	0.76

Table 4: Evaluation Metrics Scores

MODEL	SENSITIVITY	SPECIFICITY
RIPPER	0.28	0.99
HOEFFDING TREE	0.75	0.7
RANDOM FOREST	0.82	0.83
ADABOOST	0.78	0.74

Table 5: Evaluation Metrics Scores

MODEL	AUC
RIPPER	0.64
HOEFFDING TREE	0.73
RANDOM FOREST	0.82
ADABOOST	0.76

Table 6: No. of samples required for Hoeffding Tree

MODEL	No. of samples
Other models	176,416
HOEFFDING TREE	35,000

4.6.1 Discussion. Cross validation score obtained for RIPPER was better than any other model, but, the accuracy of RIPPER is least, the possible reasons could be because of addition of invalid instance during class balancing process as discussed earlier but to prove this reason we need to try any other class balancing methods like under-sampling or not perform class-balancing at all and see whether accuracy improves or not. Random Forest has the highest accuracy but Hoeffding Tree accuracy is not bad in comparison considering its importance in terms of memory and computational efficiency however, these are the tradeoffs we need to be cautious of. Sensitivity score for RIPPER is very low, which is not very good for the problem we are trying to solve, because if the model predicts no-show appointment and patient shows up then the patient might not get good treatment and this could be very dangerous if patient suffering from life-threatening disease, however, sensitivity score of Hoeffding Tree is not bad. The Specificity score of RIPPER is highest, while, specificity score of Hoeffding tree is lower than that off RIPPER's but it achieves good balance between sensitivity and specificity. This leads us to Area under the curve score, according to this metric Hoeffding TREE is able to distinguish between positive and negative clasess better than RIPPER and Random Forest and has highest AUC score.

In conclusion, Hoeffding Tree appears to provide a good balance between metric scores and memory efficiency (reducing sample size required for training by 80 percent), its scores are not bad as compared with the benchmark Random Forest model. While, the performance of the RIPPER is not good, the decision has to be made whether loosing accuracy and sensitivity is justifiable for obtaining interpret-ability with rule-set algorithm or not.

REFERENCES

[1] Mohammed Aladeemy, Linda Adwan, Amy Booth, Mohammad T. Khasawneh, and Srikanth Poranki. 2019. New feature selection methods based on opposition-based learning and self-adaptive cohort intelligence for predicting patient no-shows. https://www.sciencedirect.com/science/article/pii/S1568494619306477

[2] Alsubaie Ferwana Alnajem AlMuhaideb, Alswailem. [n.d.]. Prediction of hospital no-show appointments through artificial intelligence algorithms. https://pubmed.ncbi.nlm.nih.gov/31804138/

[3] Danae Carreras-García, David Delgado-Gómez, Enrique Baca-García, and Antonio Artés-Rodriguez. 2020. A Probabilistic Patient Scheduling Model with Time Variable Slots. https://www.hindawi.com/journals/cmmm/2020/9727096/

[4] Joseph Denney, Samuel Coyne, and Sohail Rafiqi. [n.d.]. Machine Learning Predictions of No-Show Appointments in a Primary Care Setting. https://scholar.smu.edu/datasciencereview/vol2/iss1/2/

[5] Henry Lenzi, Ângela Jornada Ben, and Airton Tetelbom Stein. [n.d.]. Development and validation of a patient no-show predictive model at a primary care setting in Southern Brazil. https://doi.org/10.1371/journal.pone.0214869

[6] Li Luo, Ying Zhou, Bernard T. Han, and Jialing Li. 2017. An optimization model to determine appointment scheduling window for an outpatient clinic with patient no-shows. https://link.springer.com/article/10.1007/s10729-017-9421-7

[7] Lawley Muthuraman. [n.d.]. A stochastic overbooking model for outpatient clinical scheduling with no-shows. https://www.tandfonline.com/doi/abs/10.1080/07408170802165823

[8] Sharan Srinivas. 2020. A Machine Learning-Based Approach for Predicting Patient Punctuality in Ambulatory Care Centers. https://doi.org/10.3390/ijerph17103703

[9] Gregor Stiglic, Primoz Kocbek, Nino Fijacko, Marinka Zitnik, Katrien Verbert, and Leona Cilar. 2020. Interpretability of machine learning-based prediction models in healthcare. https://onlinelibrary.wiley.com/doi/abs/10.1002/widm.1379