Michael Segaline

D209 Data Mining 1

Dr. Elleh

2/9/2022

Abstract:

Bottom line up front: *How well can the Random Forrest Classifier classify a patient who will readmit to the hospital?*

A major hospital chain has distributed a dataset to mitigate patient readmission to the hospital. Outside agencies implement monetary punishments to hospitals that have high readmission rates. The purpose of the dataset is to explore how effective a Random Forrest algorithm will be for the hospital and what insights might be gleaned from it. The dataset was partly processed but not normalized. The same variables used to predict patient readmission for a previous Logistic Regression Classifier, where used in this Random Forrest model. The model produced the same score as the Logistic Regression model; meaning that non-standardized data is not a factor in reducing the accuracy in Random Forrest. Additionally, given some pre-selected variables, one goal of the data analysis is to produce the best tuned Radom Forrest Classifier with no standardization of the data. The hyperparameters of the final tunned are identified with an accuracy of 63.4%. However, future data added to the model may cause overfitting and require the model to be re-tuned.

Random Forrest is a supervised learning method that uses many Classifications and Regression Tree (CART) machine learning algorithm. Specifically, the Decision Tree is utilized with different random data and produces a final decision when going through a hierarchy of nested ‘if’ statements that comprise the tree. These individual comparison operators makes up a hierarchy of questions or nodes. The first question or “Root” node produces two more questions called “Internal” nodes, which produce one of two responses each. The number of Internal nodes can vary up to the final answer produced on a “Leaf” node. The algorithm will split the decisions points to as many leaf nodes as possible. A TowardsDataScience.com article summarized, “The intuition behind Decision Trees is that you use the dataset features to create *yes / no* questions and continually split the dataset until you isolate all datapoints belonging to each class… each question creates branches segmenting the feature space into disjoint regions and uses a loss function that evaluates the split based on the purity of the resulting nodes.” (2021). Those impurity measurements are functions or parameters in the Sklearn RandomForrestClassifier() as Gini Impurity (the default setting) or Entropy.

The Gini Impurity measures variance across different classes and compares the class distribution before and after the split and is used to shrink the feature space or reduce the number of variables. The Gini function of a node is the sum of the probabilities of picking a datapoint from class “k” or “pk”, multiplied by the probability of not picking pk (1-pk). The Gini formula is: G(node) =. Variance is one of the three decompositions of “f-hat” or the sample feature. Variance is how much f-hat is inconsistent over different training sets, bootstrapped, or never seen before sets. According to Datacamp, “The end goal for supervised learning is to reduce the Generalization Error and create a model in which f-hat approximates the real or unseen features” (2021). The Gini Impurity measures that Generalization Error. A Random Forrest consists of multiple (ensemble) or “forest” of Decision Trees inferring class labels from the same dataset; Yielding, “further randomization in training individual trees” (Datacamp). Each node is measured for purity via Information Gain (IG). IG is a function taking two arguments, the specific feature, and the split point. The split point is the conditional statement and the feature importance is derived by measuring how much a variable decreases the weighted impurity.

The Random Forrest Classifier is a Bagging method because it Bootstraps the dataset by selecting random variables with replacement and aggregates the final answers of each tree in the forest. A certain percentage of the set is not trained on the model, called the “Out of Bag” set. This set is then used for testing the model to simulate how well it can infer on unseen data. The Random Forrest Classifier, then aggregates the yes / no answers at the leaf nodes creating a metamodel. The meta model gives one binary output as the final ensemble prediction. Because of so much randomizing per node, per tree, per bootstrap, It can be used to train different models on the same dataset and the final prediction is more robust and less prone to errors (Datacamp). Therefore, making the model a better inference for unseen datasets.

One assumption of a CARTs is no standardization is required. Therefore, no need to normalize to the data to the same scale. One preprocessing goal, is no normalization of data and testing the assumption that the Random Forrest will yield an accurate model. Through splitting the features into 100 different Decision Trees, running the Gini Impurity function per node, will reduce the noise and create accuracy comparable to other Machine Learning algorithms. Variables previously used in a logistic regression assignment will give a baseline for expectation. The Logistic Regression model yielded a 63 % accuracy and a precision of 63.4%. The following variables have been previously explored with a correlation heat map and show no multicollinearity. The model used the same variables predicting Hospital readmission ‘ReAdmis\_Yes’ (Categorical) with population of the patients city ‘Population’ (Continuous), if the patient is married ‘Marital\_Married’ (Categorical), if the patient has high blood pressure ‘HighBlood\_Yes’ (Categorical), number of ‘Children’ the patient has (Continuous), if the patient as suffered a stroke ‘Stroke\_Yes’ (Categorical), if the patient is Overweight ‘Overweight\_Yes’ (Categorical), Diabetes ‘Diabetes\_Yes’ (Categorical), anxiety ‘Anxiety\_Yes’ (Categorical), and asthma ‘Asthma\_Yes’.

Python will be the programing language used in the Juypter Notebooks environment. The Pandas package will be imported because it creates a data frame that can handle quick manipulation of large datasets. Additionally, Sklearn is a statistical library from which contains statistical packages containing ML models along with scoring metrics; Such as the model accuracy, the Mean Squared Error, the parameters of the best model. Also, Python has the train\_test\_split() function that allows the data to be separated into training and testing sets. Additionally, Sklearn is a statistical library from which has a various statistical packages imported containing machine learning models along with scoring metrics; Metrics such as the model score, the Mean Squared Error, and the parameters of the best model. The impurity measurements are functions imbedded as parameters in Sklearn’s RandomForrestClassifier() as Gini Impurity (default setting) or Entropy.

The first step in preprocessing the data is the import the .csv file as a Pandas Data frame then drop the null values. The next step is to drop all the features that will not be used. Third is to call the get\_dummies() function that converts categorical features to binary. Then the data is separated into two different data frames ‘X’ and ‘y’; One containing the target feature “ReAdmis\_Yes” and one containing the all the predictor variables. These two data frames are passed as arguments into the train\_test\_split() function, that returns a tuple of four variables for training and testing. 20% of the data will be allocated for testing while 80% for training. From the split, the RandomForrestClassifier() algorithm will be instantiated. Then the Grid Search Cross Validation will be used for model tunning. The GridSearchCV() function from Sklearn, takes a dictionary or “grid” of parameters for tunning the RandomForrestClassifier() model. The maximum depth of the tree, “max\_depth”, according to sklearn.org dictates the number of nodes per tree, “If default = None then nodes are expanded until all leaves are pure or until all leaves contain less than minimum number of samples” (2021). Furthermore, the grid contains a minimum sample size (from the data) per leaf, ranging from 4% to 12%. Then max number of features “max\_features” used per node and will be set to an array of numbers ranging from 20% to 90% of the features. Another parameter for Cross Validation is the number of ‘K’ folds for the model to be cross validated. The number ‘K’ equals the number of strata arrays to be used in the data.

For this model, Cross Validation ‘cv’ will be set to ten. Moreover, a default 100 Decision trees comprising the Random Forrest, will be used with the parameter grid on ten different sets of the data. These ten sets are split into testing and training data and passed through the Random Forrest. K-Fold Cross Validation is best described by Stackabuse.com, “The algorithm is trained and tested ‘K’ times, each time a new set is used as testing set while remaining sets are used for training. Yielding the average of the results obtained on each set” (2020). The last Cross Validation parameter to set is for dictating model processing speed, ‘n\_jobs’. According to AnalyticsVidhya.com, “This parameter tells the engine how many processors it is allowed to use. A value of “-1” means there is no restriction whereas a value of “1” means it can only use one processor” (2015). The new GridSearchCV() object is then fitted to the data with the fit() method. Once fitted, the data will be used to create several models with different combinations of these parameters until the best model is returned with parameters tuned. The fitted model can then be used to make predictions with the predict() method, extract the optimal hyperparameters with ‘best\_params\_’ method and the best model with ‘best\_estimator\_’.

The following is the code to produce the model:

#-- Importing necessary libraries --#

from sklearn.model\_selection import GridSearchCV

import pandas as pd

import numpy as np

from sklearn.ensemble import RandomForestClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_squared\_error as MSE

**#-- Importing the dataset --#**

**med\_df\_raw = pd.read\_csv("/Users/lindasegalini/Desktop/WGU/New Program/D208 Predictive Modeling/medical\_clean.csv")**

**#-- Dropping missing values --#**

**med\_df\_raw.dropna()**

**#-- Dropping unessesary features --#**

**med\_df = med\_df\_raw.drop(columns = ['Gender','Complication\_risk','Initial\_admin','Additional\_charges','TotalCharge','Initial\_days','VitD\_levels','Age','CaseOrder','Allergic\_rhinitis','Reflux\_esophagitis','Doc\_visits','Full\_meals\_eaten','vitD\_supp','Soft\_drink','Customer\_id','Interaction','UID','City','State','County','Zip','Lat','Lng', 'Area','TimeZone','Job','Income','Services','Item1','Item2','Item3','Item4','Item5','Item6','Item7','Item8', 'Arthritis','Hyperlipidemia','BackPain'])**

**#-- Changing categorical to binary with get\_dummies() and dropping the first column.**

**med\_df = pd.get\_dummies(med\_df, drop\_first =True)**

**#-- Dropping the previously combined columns to keep the number of columns down and create a tidier dataset --#**

**med\_clean = med\_df.drop(columns = ['Marital\_Never Married','Marital\_Separated','Marital\_Widowed'])**

**#-- Saving a copy of the cleaned dataset --#**

**med\_clean.to\_csv('/Users/lindasegalini/Desktop/WGU/New Program/D209 Data**

**Mining/RandomForests.csv')**

**#-- Split the data into X & y data sets --#**

**X = med\_clean.drop('ReAdmis\_Yes', axis = 1).values**

**y = med\_clean['ReAdmis\_Yes']**

**y = y.astype(int)**

**#-- Make train and test sets --#**

**from sklearn.model\_selection import train\_test\_split**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20,**

**shuffle=True, random\_state=2)**

**#-- Instantiate RandomForrestClassifier --#**

**rc = RandomForestClassifier(random\_state = 1)**

**#-- print out the 'rc's hyperparameters --#**

**rc.get\_params()**

**#-- Define the grid of hyperparameters 'params\_RC' --#**

**params\_RC ={'max\_depth': [3,4,5,6,8,10],**

**'min\_samples\_leaf': [0.04, 0.06,0.08,0.10,0.12],**

**'max\_features': [0.2, 0.4, 0.6, 0.8, 0.10, 0.12]}**

**#-- Instanitate a 10-fold CV grid search object 'grid\_RC' --#**

**grid\_RC = GridSearchCV(estimator = rc,**

**param\_grid = params\_RC,**

**scoring = 'accuracy',**

**cv = 10,**

**n\_jobs =-1)**

**#-- Fit 'grid\_RC' to the training set --#**

**grid\_RC.fit(X\_train, y\_train)**

**#-- Predict the test set labels 'y\_pred' --#**

**y\_pred = grid\_RC.predict(X\_test)**

**#-- Extracting the best hyperparameters from 'grid\_RC' --#**

**best\_hyperparams = grid\_RC.best\_params\_**

**print('Best hyperparameters : \n', best\_hyperparams)**

**#-- Extracting the best model from 'grid\_RC' --#**

**best\_model = grid\_RC.best\_estimator\_**

**#-- Evaluate the accuracy --#**

**test\_acc = best\_model.score(X\_test, y\_test)**

**#-- Print the best\_model accuracy --#**

**print("Test set accuracy of the best model: {:.3f}".format(test\_acc))**

**#-- Evaluate test set MSE 'mse\_test' --#**

**mse\_test = MSE(y\_test, y\_pred) \*\*(1/2)**

**#-- Print 'mse\_test' --#**

**print('Test set MSE: {:.2f}'.format(mse\_test))**

#-- Feature importance is an average of how each variables effect in reducing the model's noise--#

importances = grid\_RC.best\_estimator\_.feature\_importances\_

importancesmed\_clean.drop('ReAdmis\_Yes', axis = 1)

#-- Displaying feature names matched with the amount of correlation to the target variable

#-- 'Population' is the first in the array above with 'Children' next at 28.4% and so on.

med\_clean.drop('ReAdmis\_Yes', axis = 1)

#-- Saving a copy of the training and testing datasets --#

X\_train =pd.DataFrame(X\_train)

X\_test = pd.DataFrame(X\_test)

y\_train = pd.DataFrame(y\_train)

y\_test = pd.DataFrame(y\_test)

X\_train.to\_csv('/Users/lindasegalini/Desktop/WGU/New Program/D209 Data Mining/RandomForests/Submissions.csv')

X\_test.to\_csv('/Users/lindasegalini/Desktop/WGU/New Program/D209 Data Mining/RandomForests/Submissions.csv')

y\_train.to\_csv('/Users/lindasegalini/Desktop/WGU/New Program/D209 Data Mining/RandomForests/Submissions.csv')

y\_test.to\_csv('/Users/lindasegalini/Desktop/WGU/New Program/D209 Data Mining/RandomForests/Submissions.csv')

The following is the output from the model :

Graphical user interface, text, application, email

Description automatically generated

The final accuracy of the best model is 63.4% which is the exact accuracy of the Logistic Regression model. Given the same variables, the Random Forrest Classifier ensemble method performs just as well as Logistic Regression without standardized data. The hierarchical randomization, along with bootstrapping, Cross Validation, and the Gini Impurity function did a good job standardizing the data. The best model yielded hyperparameters with a maximum depth of 3 nodes, with 20% of the features per node, and a minimum sample size, 4% of the data, per leaf. The Mean Square Error (MSE) is the squared average distance all the residual datapoints are from the line of best fit. The lower the MSE the better the model and in this case being 0.37. MSE is not used as a scoring measure for Random Forest Classification models, only for Regression models. In this case, the conversion of the dataset using the get\_dummies() function is predicting a binary outcome. According to sklean.org on the MSE() function, the binary conversion still creates the correct arguments necessary to compile. Which 1- 0.37 = 0.63, which approximates the accuracy.

In final analysis, the two predominant features are Population 27.9% at variance and number of children per patient at 28.4%. One drawback of the model, “Finding the best tree is ideal in theory, but as the dataset grows, it becomes computationally unfeasible… a perfect score on training data is usually a good indicator that we will face a disappointing drop in performance on new data” (Towardsdatascience.com). Overfitting is likely when the hospital doubles or triples the data and the model will have to be tuned again. Regardless, the Random Forrest Classifier will still be a useful algorithm with no worries of normalizing new data. To negate overfitting with new data, the number of trees can be increased during the tuning processes. Now that a baseline score is established, a recommended course of action is to add more data, retune the model, and check the new accuracy.

Work Cited

Bento, C. (2021, July 18). *Decision tree classifier explained in real-life: Picking a vacation destination*. Medium. Retrieved February 10, 2022, from https://towardsdatascience.com/decision-tree-classifier-explained-in-real-life-picking-a-vacation-destination-6226b2b60575

*Generalization error: Python*. campus.datacamp.com. (n.d.). Retrieved February 10, 2022, from https://campus.datacamp.com/courses/machine-learning-with-tree-based-models-in-python/the-bias-variance-tradeoff

Loading... (n.d.). Retrieved February 10, 2022, from https://scikitlearn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Malik, U. (2019, March 13). *Cross validation and Grid Search for model selection in Python*. Stack Abuse. Retrieved February 10, 2022, from https://stackabuse.com/cross-validation-and-grid-search-for-model-selection-in-python/

*Random Forest parameter tuning: Tuning Random Forest*. Analytics Vidhya. (2020, June 26). Retrieved February 10, 2022, from https://www.analyticsvidhya.com/blog/2015/06/tuning-random-forest-model/

*Sklearn.ensemble.randomforestclassifier*. scikit. (n.d.). Retrieved February 10, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html