Classification Tree Analysis of Tobacco Product Use in the Population Assessment of Health Study

Introduction

Tobacco products, such as cigarettes, e-cigarettes and hookah, are commonly used among adolescents aged from 12 - 17. In 2020, nearly 16.2% of middle and high school students in the United States reported current use of any tobacco product, approximately one in four high school students and one in fifteen middle school students had used tobacco products in the past 30 days [1]. Most of the tobacco products contain nicotine, which is a highly addictive chemical compound that can alter the reward system in our brain by over-activating the reward circuitry and lead to compulsive nicotine seeking behaviors when it is not compensated [2].

Young people are at increased risk for nicotine addiction, as their brains are still under development. Studies show that the use of tobacco products in adolescence can lead to prolonged nicotine exposure and dependence, therefore when the individual tries to quit or reduce the amount of exposure, such dependence will mostly result in symptoms of withdrawal and consequently unsuccessful quit attempts [2, 3]. In the United States and other countries, using tobacco products remains a leading cause of preventable deaths. On average, nearly 435,000 people in US die from smoking-related diseases each year, and the chance that a lifelong smoker will die prematurely due to the use of nicotine products is approximately 50% [4]. Apart from that, tobacco products use is also a strong risk factor for various diseases, including cancer, cardiovascular disease,

pulmonary disease, and respiratory tract infections. Therefore, interventions are needed to control and reduce adolescents' use of all tobacco products.

Previous studies have reported several sets of risk factors that may have an association with tobacco use [5]. These factors include: (1) Sociodemographic factors, such as socioeconomic status, developmental challenges of adolescence, gender, and race/ethnicity; (2) Environmental factors, such as acceptability and availability of tobacco products, interpersonal variables, perceived environmental variables; (3) Behavioral factors, including academic achievement, problem behaviors, influence of peer groups, participation in activities, and behavioral skills; (4) Personal factors, such as knowledge of the long-term health consequences of using tobacco, functional meanings of tobacco use, subjective expected utility of tobacco use, variables related to self-esteem, and personality; and (5) Current behavior relative to tobacco use, for example, intentions to smoke and smoking status. However, there is a lack of sufficient literature on how these risk factors may be utilized to classify and predict the smoking behaviors among adolescents.

To better understand and predict the use of tobacco products, classification tree analysis is needed as it exhaustively identifies subgroups of a population whose members share common characteristics that influence the outcome variable [6]. It begins with one parent node that contains all the samples, and continuously splits the group into binary subgroups according to some predetermined criterion. The method, which is also called recursive partitioning, is widely used in the field of public health

and statistical learning. Compared to other classification algorithms, decision tree is simpler, intuitive and easy to interpret, and requires less effort for data preparation during pre-processing. It can be combined with other decision techniques, such as logistic regression, and improve the overall prediction accuracy. Missing values in the data do not affect the process of tree construction, therefore it can generate values even with rougher data.

Decision tree method also has several disadvantages, for instance, it usually tends to be unstable and have relatively less accurate results compared to other algorithms. One way of improvement is to combine it with different classification techniques, while another way is to apply the gradient boosting method by assembling multiple weak prediction models together, and form the so-called ‘gradient boosted trees’, which usually outperforms random forest [7].

In this study, I plan to construct a decision tree learner to classify and predict the smoking behavior among US adolescents who participated in the Population Assessment of Health (PATH) Study at Wave 1, and conduct exploratory analyses to determine which risk factors are most strongly associated with the use of tobacco products. The expected outcome is that the classifier will be able to predict the smoking category (Smoker vs Never-user) of a given individual based on a set of certain predictors, and the out-of-sample accuracy will be able to reach at least 70%. Logistic regression will be combined with the decision tree to improve the overall performance of prediction. Gradient boosting method will also be applied to the data,

and the optimal parameters will be selected based on cross validation. All the above models will be evaluated according to their prediction accuracies on the test data. The main hypothesis is that both the prediction accuracies of the gradient boosted tree and the tree combined with logistic regression will be higher than that of the original decision tree.

Study Design

Data and Participants

Analyses are conducted using the Population Assessment of Tobacco and Health dataset, a nationally representative, longitudinal cohort study with 13651 adolescents (aged 12 - 17 at Wave 1) and over 43000 adults across US. Data were collected by trained interviewers in person, including information regarding use of cigarettes, e-cigarettes, cigars, hookah, smokeless tobacco, and other tobacco products. Only the adolescents at Wave 1 who do not have completely missing information are used in this study. The primary outcome of interest is whether the individual has ever tried or used tobacco products prior to the interview. Due to low counts for some products, I only focused on cigarettes, e-cigarettes, cigar-like products, smokeless tobacco, and hookahs. The total number of the study population is 8615.

Methods

The decision tree method is a non-parametric statistical model for identification, classification, and multi-stage decision making. A tree is constructed by recursively partitioning the feature space of the training dataset. The objective of this process is to

find a series of decision rules that partition the feature space and provide a robust and informative hierarchical classification [8]. It begins with one parent node that stores all the samples, and split the parent node to generate two child nodes. Within each node, the algorithm examines the remaining independent variables to determine which variable results in the best split based on the pre-specified criterion. Then it repeats this process and split both child nodes into four separate offspring nodes. Such process continues through each branch until a certain terminal rule is reached. All the subjects within each node will be assigned to a certain class. For binary classification, the impurity of a node T is defined as a nonnegative function of the probability $P(Y = 1 \mid T)$, which represents the rate of misclassified subjects in that node T. Splitting criteria of a decision tree is defined based on the concave-shaped impurity functions $\Phi[P(Y = 1 \mid T)]$. Properties for the impurity functions include: (1) $\Phi \geq 0$; (2) $\Phi(P) = \Phi(1 - P)$ for $P \in (0, 1)$; and (3) $\Phi(0) = \Phi(1) < \Phi(P)$. The splitting criterion selects the split that results in the largest difference between the impurity of the parent node and a weighted average of impurities of the child nodes. Most commonly, the impurity function takes one of the three following forms:

(I) Bayes error: $\Phi(P) = \min(P, 1 - P)$;

(II) Entropy: $\Phi(P) = -P \log(P) - (1 - P) \log(1 - P)$;

(III) Gini index: $\Phi(P) = P (1 - P)$.

The terminal nodes of a decision tree are where the splitting ends. To evaluate the homogeneity of the terminal nodes, define $R(T) = \Sigma P(t) r(t)$ for a decision tree T and its terminal nodes t. The purpose of pruning a decision tree is to find the best subtree

such that R(T) is minimized. Here, r(t) represents the misclassification cost, which is set to zero when correctly classified and c(0 | 1) or c(1 | 0) when misclassified. A node t will be assigned to class j if $\Sigma[c(j \mid i)\, P(Y = i \mid t)] \leq \Sigma[c(1 - j \mid i)\, P(Y = i \mid t)]$.

In order to prevent the tree from over-growing, cost-complexity needs to be used for selecting the right-sized tree: $R_\alpha(T) = R(T) + \alpha \mid t \mid$, where the non-negative α is the complexity parameter and | t | is the number of terminal nodes in tree T. For any value of the complexity parameter α, there exists a unique smallest subtree of T that minimizes the cost-complexity [9].

To determine when to prune off a certain branch, we can compute α with the following formula [10]:

$$\frac{R^s(t) - R^s(T)}{|T| - 1}$$

Where $R^s(T)$ represents the sum of the re-substitution misclassification costs of the offspring terminal nodes of an internal node t. The smallest α over all | T | - 1 internal nodes is the first positive threshold α_1. And with α_1, we can prune off the offspring nodes of internal node t when $R^s(t) + \alpha_1 \leq R^s(T) + \alpha_1 \mid T \mid$. Similarly, the other threshold α_i can be derived and used to remove extra branches until there is no terminal nodes can be pruned off. In this way, a set of nested optimal subtrees can be constructed with $0 < \alpha_i < \alpha_n$, $T\alpha_0 \supset T\,\alpha_i \supset T\,\alpha_n$. Apart from pruning back a grown tree, researchers may specify their own stopping rules by defining the maximum depth of a decision tree, or the minimum number of individuals within a terminal node.

One of the disadvantages of decision tree classifier is that it usually has lower accuracy than other classification tools. To make more efficient use of data and improve the performance of decision tree, we can combine it with other analytic strategies, usually the binary logistic regression. Logistic regression models the linear relationship between the predictors and the natural log of the odds ratio for our dependent variable. It can be expressed as the following:

$$l = log\frac{p}{1-p} = \beta_0 + \Sigma\beta_i x_i$$

$$p = \frac{1}{1+e^{-(\beta_0+\Sigma\beta_i x_i)}} = S(\beta_0 + \Sigma\beta_i x_i)$$

The probability of outcome equaling to 1 is denoted as p. S is often called the Sigmoid function.

The first approach to combine logistic regression with decision trees is to take the linear relationship derived from the regression as a new predictor, and use the created variable, along with other predictors, for tree classification. Another approach is to grow a decision tree first, and then fit the logistic regression. Create binary variables that correspond to each terminal nodes of the decision tree, treat them as the new predictors, and use the created variables along with the original predictors to refit the logistic regression. In this study, I choose the first approach to combine the two models.

To further improve the performance of tree classifiers, it is also common to adopt an ensemble of weak tree classifiers, and combine them into a single strong leaner in an iterative fashion. Such method is called gradient boosting. The objective of gradient boosting is to minimize sum of squared errors. Our first step is to initialize a weak classifier $F_0 = 1 / N \Sigma y_i$. At each iteration stage m, an imperfect model $F_m(x)$ is fitted according to the residuals of $y - F_{m-1}(x)$. Gradient boosting utilizes the additive model, that is, the trees are added one at a time, and existing trees in the model will not be changed. Therefore, the newly fitted model will then be combined to the previous models with a learning rate λ, as we set $S_{m+1}(x) = S_m(x) + \lambda F_m(x)$. In general, the algorithm can be presented as:

(I) Initialize $S_0(x) = f_0(x) = 1 / N \Sigma_i y_i$;

(II) For m in 1 to M, calculate the negative gradient (pseudo-residuals) :

$$r_m = -\frac{\alpha L(y,\hat{y})}{\alpha \hat{y}}$$

Here L is the loss function and y.hat represents the estimated value. In the case of binary classification, logarithmic loss is often used for L.

(III) Create the working dataset $W_m = (x, r_m)$. Fit a new weak learner to W_m that minimizes the errors between r_m and F.

$$F_m(x) = \text{argmin}_F L(r_m, F).$$

(IV) Choose the learning rate λ_m that minimizes $L(y_i, S_{m-1}(x_i) + \lambda F_m(x_i))$.

$$\lambda_m = \text{argmin}_\lambda L(y_i, S_{m-1}(x_i) + \lambda F_m(x_i)).$$

(V) Set $S_{m+1}(x) = S_m(x) + \lambda_m F_m(x)$.

(VI) Return $S_M(x) = \Sigma \lambda F_i(x) \{i = 0, 1, \ldots, M\}$.

K-fold cross validation will be used in order to find the optimal parameters for gradient boosting. The total study samples will be randomly divided into k equal sized subgroups, and one of these subgroup will be left out for testing, as the remaining k - 1 groups of subsamples will be used to train the models. Repeat such process k times, with each subgroup used exactly once as the validation data. Average the results over all k runs and estimate parameters accordingly.

Statistical Analyses

Based on whether the participants have ever used or tried cigarettes, e-cigarettes, cigars, smokeless tobacco, or hookahs prior to the PATH study, the major outcome of interest in this study is coded as a binary variable: 0 for non-users, and 1 for ever-users. Risk factors that may have a major impact on the individual's smoking behaviors are selected according to previous literatures, including: age, race / ethnicity, gender, parental education, number of impulsivity-related symptoms, number of internalizing-related symptoms, number of externalizing-related symptoms, whether other people use tobacco products at home, rules about using combustible and non-combustible tobacco, whether parents or guardians had conversations about not using tobacco products, advertisement exposure, and access of tobacco products. Among these predictors, impulsivity-related symptoms, internalizing-related symptoms, and externalizing-related symptoms are continuous variables ranged from 1 to 5, while

others are dichotomous variables coded as 0 or 1. The levels and descriptive statistics of these predictors are presented in Table 1:

PREDICTORS	LEVEL	FREQUENCY / WEIGHTED PERCENT
AGE	12 -14	4358 (50.59%)
	15 - 17	4257 (49.41%)
SEX	Male	4368 (50.70%)
	Female	4247 (49.30%)
RACE / ETHNICITY	Non-hispanic white	4146 (48.13%)
	Others	4469 (51.87%)
PARENT EDUCATION	High school or less	3375 (39.18%)
	More than high school	5240 (60.82%)
OTHER USE AT HOME	Someone uses tobacco	2925 (33.95%)
	No one uses tobacco	5690 (66.05%)
RULES ABOUT COMBUSTIBLE AND NON COMBUSTIBLE TOBACCOS	Not allowed	6393 (74.21 %)
	Allowed	2222 (25.79 %)
CONVERSATIONS ABOUT USE	Yes	4463 (51.80%)
	No	4152 (48.20%)
ADVERTISEMENTS	No	4537 (52.66 %)
	Yes	4078 (47.34 %)
EASY ACCESS	Easy	6393 (74.21 %)
	Difficult	2222 (25.79 %)

PREDICTORS	MEAN	MEDIAN	STANDARD DEVIATION
IMPULSIVITY-RELATED SYMPTOMS	0.91	1.00	0.79
INTERNALIZING-RELATED SYMPTOMS	2.42	2.00	1.79
EXTERNALIZING-RELATED SYMPTOMS	1.64	2.00	1.40

Table 1: Levels and descriptive statistics of the predictors used for tree classifiers.

Anaconda Python 3.0 is used for tree construction, logistic regression, and gradient boosting. Packages including sklearn, xgboost, and statsmodels are imported for analyses. Decision trees are constructed using the DecisionTreeClassifier function, with splitting criterion set to be the entropy impurity. Data are divided into training and test sets on a 7 : 3 ratio basis. After fitting the tree model, accuracy score of predicted values is evaluated on the test dataset. Optimization is conducted by specifying the maximum depth of the tree, and the accuracy score reached the highest at depth = 4. Comparing the performance of the classifier with and without pre-specified depth, the model with specified depth = 4 is selected as the final classifier. A logistic regression model is fitted using the above predictors and a training set that contains 70% of the total population. The coefficients of the logistic model are then used to define a new predictor named “combined score”. Update the previous classifier by adding this new predictor to the original data and refitting the decision tree. This time, the optimal depth is changed to 3.

To determine the most suitable parameters for gradient boosting, a 10-fold cross validation is adopted to select the best tree depth, the optimal learning rate, and the total number of trees to be added. Range of the tree depth is defined as [1, 9], learning rate is chosen from 0.005, 0.01, 0.05, 0.1, 0.3, 0.5, and 1, and the possible number of trees ranges from 1 to 100. All these parameters are tested using the cross validation, and the combination which results in the smallest mean error is selected. Then a final model with the chosen parameters is trained using a training set containing 70% of the study population. The total number of iterations is set to be 1000, and the error of each iteration is measured by the logistic loss. Prediction accuracy of the final model is evaluated by the test set.

Finally, a gradient boosting classifier with the combined score is also trained with 1000 iterations after determining the optimal parameters. The prediction accuracy is evaluated and compared with previous models. Structures of all the above classifiers are plotted.

Results

Figure 1 shows basic structures of the decision trees before and after combining the logistic regression. The out-of-sample prediction accuracy of the first tree is 80.309%. After fitting the regression, the combined score from the logistic equation is defined as:

Score = - 3.3501 + 1.6373 Age - 0.0691 Race / Ethnicity - 0.3884 Gender - 0.1618 Parental Education - 0.1449 Impulsivity symptoms + 0.0924 Internalizing symptoms +

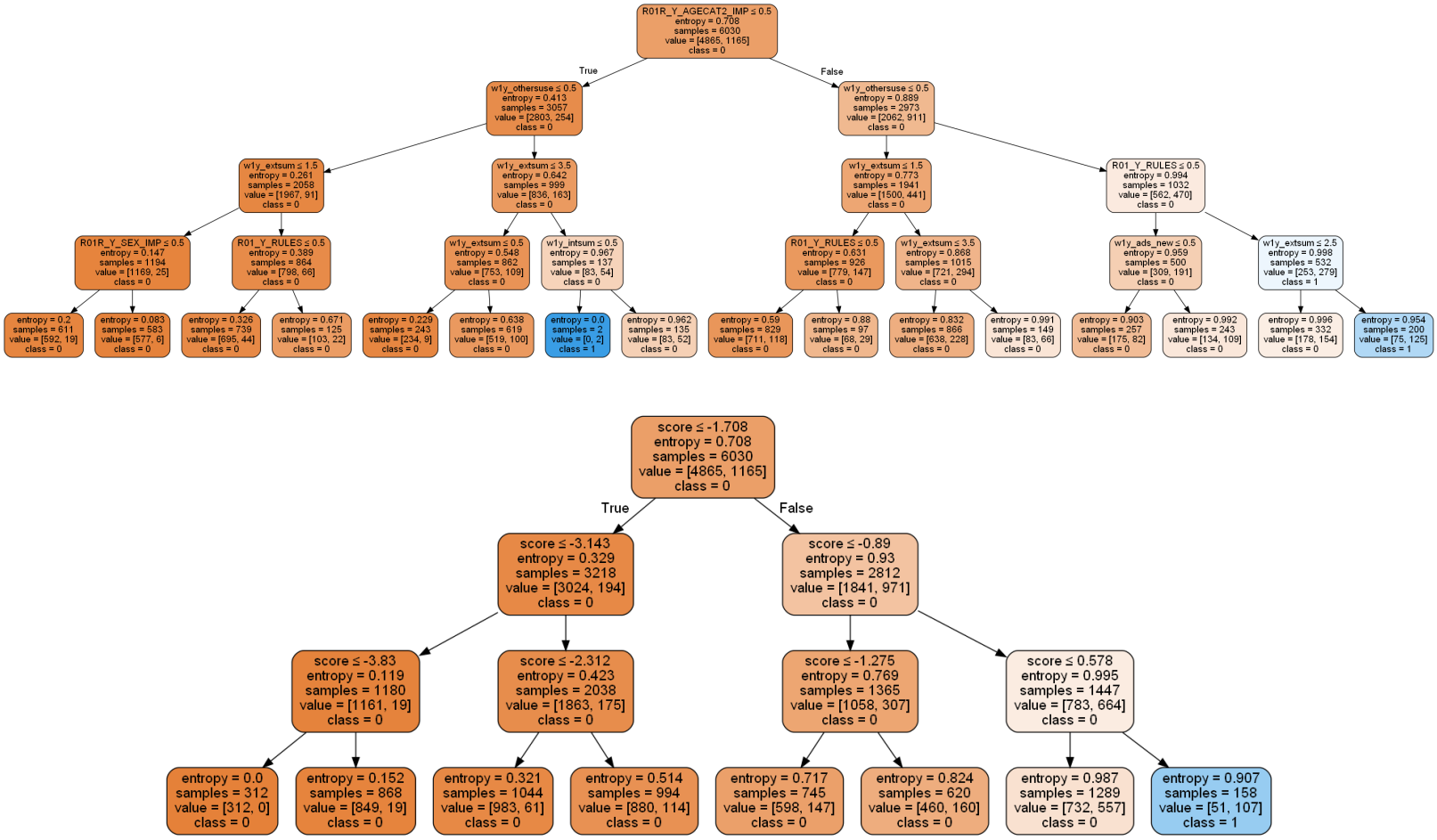


Figure 1: Decision tree structures.

0.3058 Externalizing symptoms + 0.8246 Other use + 0.0576 Conversation about use + 0.3013 Advertisements - 0.2935 Access + 0.6675 Rules

Prediction accuracy of the logistic model is 80.348%, and the overall accuracy of the new decision tree after combination is 80.619%. From the above structure, we can see that age category (12- 14 vs 15- 17) is the most important risk factor, followed by whether others use tobacco at home, externalizing-related symptoms, and rules about tobacco use. The combined score turns out to be the dominant predictor in the second tree. Individuals with a score over 0.578 are classified as tobacco users. The test

accuracy of the second tree is slightly higher than the first tree, indicating small improvement after combining logistic regression.

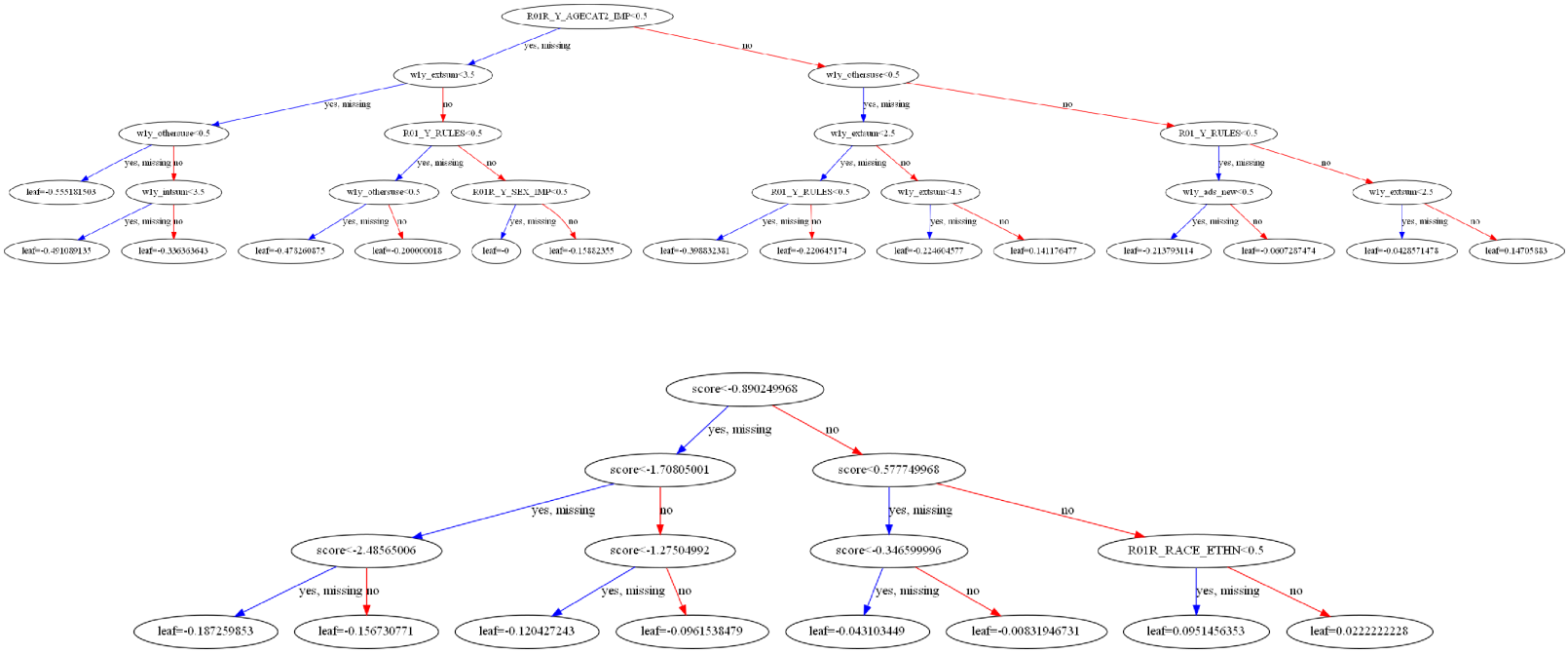


Figure 2: Gradient boosted trees.

Figure 2 presents the structures of gradient boosted trees before and after combining the logistic regression. After cross validation, the maximum depth of the first tree is chosen to be 4, and 0.3 is selected as the optimal the learning rate. The total number of subtrees added to the model is 7, and the averaged error of each parameter is shown in Figure 3. After training the gradient boosting model with the above parameters and 1000 iterations, the final accuracy score reached 80.387%. When the combined score is added to the dataset as a new predictor, the maximum depth chosen after cross validation changed to 3, and the learning rate in this case is 0.1. A total number of 78 trees are added to the final boosted model. The averaged error of each parameter is

also presented in Figure 3. After training with the optimal parameters, the out-of-sample prediction accuracy of the second tree classifier is 80.116%.

From the above structures, we can see that the age category is again the most important predictor of initialization of smoking. Externalizing-related symptoms, other people use tobacco at home, and rules about tobacco use are also major factors of tobacco use. After combining with logistic regression, the new score turns out to be the main predictor of tobacco product use. Apart from that, race / ethnicity also contributes to the classification of smokers. The overall prediction accuracy of the first gradient boosted tree is slightly higher than the decision tree without combination, indicating the gradient boosting method can somewhat improve the performance of recursive partitioning. However, combining the logistic regression to the gradient boosted tree does not help with the prediction accuracy in this case.

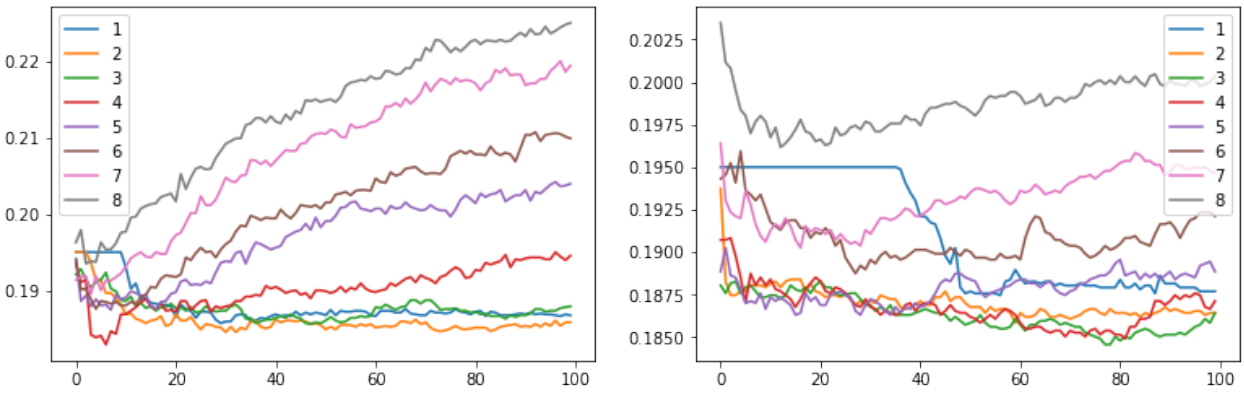


Figure 3: Cross validation errors for parameters depth = [1, 9], learning rate = 0.3 / 0.1, number of trees = [1, 100].

Left: Tree without combined score. Right: Tree with combined score.

Conclusion

This study constructed decision tree classifiers and gradient boosted trees to distinguish tobacco users and those who never used tobacco products before based on data from the Population Assessment of Health Study. The results indicate that age, externalizing-related symptoms, whether other people use tobacco at home, and rules about tobacco use are the major predictors of the initialization of tobacco use. Combination of logistic regression with decision tree and use of gradient boosting can respectively improve the prediction accuracy of the classifier. Several limitations of this study are: (1) Insufficient predictors included in the analyses. There may be other risk factors that have significant influence on the use of tobacco products; and (2) Imbalance of the two categories. The proportion of tobacco users in this dataset is small, which may eventually lead to inaccuracy classification.

Reference

[1] Gentzke, A. S., Wang, T. W., Jamal, A., Park-Lee, E., Ren, C., Cullen, K. A., & Neff, L. (2020). Tobacco Product Use Among Middle and High School Students - United States, 2020. MMWR. Morbidity and mortality weekly report, 69(50), 1881–1888.

[2] Osibogun, O., Taleb, Z. B., Bahelah, R., Salloum R. G., Maziak, W. (2018). Correlates of Poly-Tobacco Use among Youth and Young Adults: Findings from the Population Assessment of Tobacco and Health Study, 2013–2014, Drug and Alcohol Dependence, Volume 187, Pages 160-164, ISSN 0376-8716.

[3] Henningfield, J.E., Rose, C.A., Giovino, G.A. (2002). Brave new world of tobacco disease prevention: promoting dual tobacco-product use? Am J Prev Med. 2002 Oct;

23(3):226-8. doi: 10.1016/s0749-3797(02)00502-0. Erratum in: Am J Prev Med. 2003 Jan; 24(1):110. PMID: 12350457.

[4] Benowitz N. L. (2010). Nicotine addiction. The New England journal of medicine, 362(24), 2295–2303.

[5] National Center for Chronic Disease Prevention and Health Promotion (US) Office on Smoking and Health. Preventing Tobacco Use Among Youth and Young Adults: A Report of the Surgeon General. Atlanta (GA): Centers for Disease Control and Prevention (US); 2012. 4, Social, Environmental, Cognitive, and Genetic Influences on the Use of Tobacco Among Youth.

[6] Lemon, S.C., Roy, J., Clark, M.A. et al. Classification and regression tree analysis in public health: Methodological review and comparison with logistic regression. ann. behav. med. 26, 172–181 (2003).

[7] Piryonesi, S. Madeh; El-Diraby, Tamer E. (2020-03-01). "Data Analytics in Asset Management: Cost-Effective Prediction of the Pavement Condition Index". Journal of Infrastructure Systems. 26 (1): 04019036.

[8] Myles, Anthony & Feudale, Robert & Liu, Yang & Woody, Nathaniel & Brown, Steven. (2004). An Introduction to Decision Tree Modeling. Journal of Chemometrics. 18. 275 - 285. 10.1002.

[9] Leo Breiman, Jerome Friedman, Charles J. Stone, R.A. Olshen (1984). Classification and Regression Trees. Taylor & Francis.

[10] Heping Zhang, Burton H. Singer (2010). Recursive Partitioning and Applications. Springer-Verlag New York

In [1]:
```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
import numpy as np
from sklearn.model_selection import KFold
import xgboost as xgb
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
#!pip install graphviz
#from graphviz import Digraph
#print(graphviz.__version__)
```

In [2]:
```
data = pd.read_csv("finaldata646.csv")
data.head()
```

Out[2]:

	ID	ecig1	ecig2	ecig3	ecig4	smkless1	smkless2	smkless3	smkless4	hookah1	...
0	1	0	0.0	0.0	0.0	0	0.0	0.0	0.0	0	...
1	2	0	1.0	1.0	1.0	0	0.0	0.0	0.0	0	...
2	3	0	0.0	0.0	0.0	0	0.0	0.0	0.0	0	...
3	4	0	0.0	0.0	0.0	0	0.0	0.0	0.0	0	...
4	5	0	0.0	0.0	0.0	0	NaN	NaN	NaN	0	...

5 rows × 344 columns

In [3]:
```
tree_depth_choices = np.arange(1, 9)
learning_rate_choices = [0.005, 0.01, 0.05, 0.1, 0.3, 0.5, 1]
possible_trees = np.arange(1,100)
```

In [4]:
```
features = ['R01R_Y_AGECAT2_IMP', 'R01R_RACE_ETHN', 'R01R_Y_SEX_IMP', 'R
01R_Y_PM0002',
            'w1y_impuls', 'w1y_intsum', 'w1y_extsum', 'w1y_othersuse',
'R01_YX0500',
           'w1y_ads_new', 'w1_access_bi','R01_Y_RULES', 'smoke']
df = data[features].dropna(axis = 0)
X = df.drop('smoke', axis = 1)
y = df['smoke']
```

In [38]:
```
#X.isna().sum()
X.head()
```

Out[38]:

	R01R_Y_AGECAT2_IMP	R01R_RACE_ETHN	R01R_Y_SEX_IMP	R01R_Y_PM0002	w1y_
0	1	0	0	1.0	2.0
1	1	1	0	1.0	0.0
2	0	0	1	1.0	2.0
3	1	1	1	1.0	2.0
4	0	0	0	1.0	0.0

In [6]:
```
def cvfunc(x, y, params, max_trees = 100, folds = 10):
    kf = KFold(n_splits=folds, shuffle=True)
    cv_error = list()
    for train_index, test_index in kf.split(x):
        dtrain = xgb.DMatrix(x[train_index,:],label = y[train_index])
        bst = xgb.train(params,dtrain, num_boost_round = max_trees)
        dtest = xgb.DMatrix(x[test_index,:],label = y[test_index])
        ytest = y[test_index]
        errors_vs_numtrees = np.array([np.mean(np.logical_xor(bst.predic
t(dtest,ntree_limit=j)>0.5, ytest)) for j in np.arange(1,101)])
        cv_error.append(errors_vs_numtrees)

    return np.mean(np.array(cv_error), axis = 0)
```

In [164]:
```
param = {'max_depth': 9, 'eta': 1, 'objective': 'binary:logistic',
          'eval_metric' : 'logloss'}
allerrors = np.zeros((len(tree_depth_choices),len(learning_rate_choices
), 100))

for i, treedepth in enumerate(tree_depth_choices, 0):
    param['max_depth']=treedepth
    for j, lr in enumerate(learning_rate_choices, 0):
        param['eta']=lr
        allerrors[i, j] = cvfunc(X.values, y.values, param)
```

In [168]:
```
x=(np.unravel_index(np.argmin(allerrors), allerrors.shape))
param['max_depth'] = tree_depth_choices[x[0]]
param['eta'] = learning_rate_choices[x[1]]
max_trees = possible_trees[x[2]]
```

In [169]:
```
param
```

Out[169]:
```
{'max_depth': 4,
 'eta': 0.3,
 'objective': 'binary:logistic',
 'eval_metric': 'logloss'}
```

In [167]:
```
max_trees
```

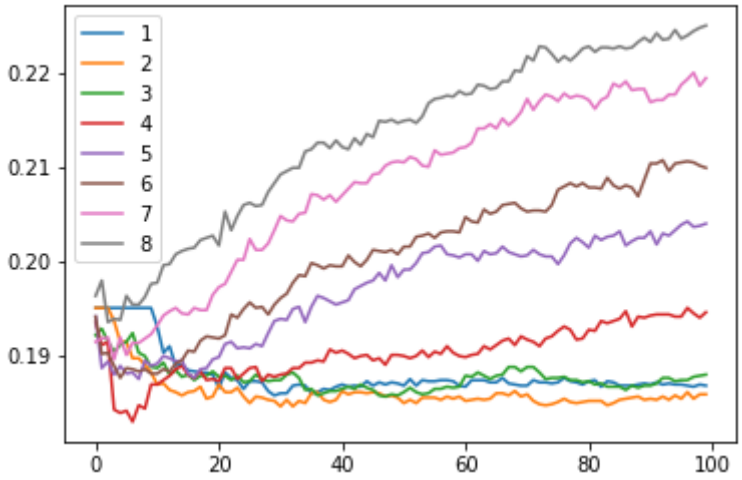
Out[167]: 7

In [110]:
```
allerrors.shape
```

Out[110]: (8, 7, 100)

In [171]:
```
for j in range(8):
    plt.plot(allerrors[j, 4, :], label=tree_depth_choices[j])
plt.legend()
```

Out[171]: <matplotlib.legend.Legend at 0x2807a58c18>



In [176]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.
3,
                                                    random_state = 0)
dtrain = xgb.DMatrix(X_train, y_train, feature_names = features[0:12])
dtest = xgb.DMatrix(X_test, y_test, feature_names = features[0:12])
params = {'max_depth': 4, 'eta': 0.3, 'objective': 'binary:logistic',
          'eval_metric': 'logloss'}

evallist = [(dtrain, 'train')]
iteration = 1000
bst = xgb.train(params, dtrain, iteration, evallist)
```

```
[969]   train-logloss:0.25524
[970]   train-logloss:0.25520
[971]   train-logloss:0.25516
[972]   train-logloss:0.25510
[973]   train-logloss:0.25506
[974]   train-logloss:0.25502
[975]   train-logloss:0.25497
[976]   train-logloss:0.25492
[977]   train-logloss:0.25487
[978]   train-logloss:0.25485
[979]   train-logloss:0.25480
[980]   train-logloss:0.25472
[981]   train-logloss:0.25469
[982]   train-logloss:0.25465
[983]   train-logloss:0.25463
[984]   train-logloss:0.25459
[985]   train-logloss:0.25457
[986]   train-logloss:0.25453
[987]   train-logloss:0.25450
[988]   train-logloss:0.25442
[989]   train-logloss:0.25437
[990]   train-logloss:0.25433
[991]   train-logloss:0.25430
[992]   train-logloss:0.25426
[993]   train-logloss:0.25421
[994]   train-logloss:0.25415
[995]   train-logloss:0.25413
[996]   train-logloss:0.25408
[997]   train-logloss:0.25402
[998]   train-logloss:0.25399
[999]   train-logloss:0.25392
```

In [178]:
```
y_pred = bst.predict(dtest, ntree_limit = 7)
predictions = [round(value) for value in y_pred]
accuracy_score(y_test, predictions)
```

Out[178]:
```
0.8038684719535784
```

In [20]:
```
from xgboost import XGBClassifier
from xgboost import plot_tree
from numpy import loadtxt
from sklearn import tree
```

In [21]:
```
import os
os.environ["PATH"] += os.pathsep + 'C:/Program Files/Graphviz/bin/'
```

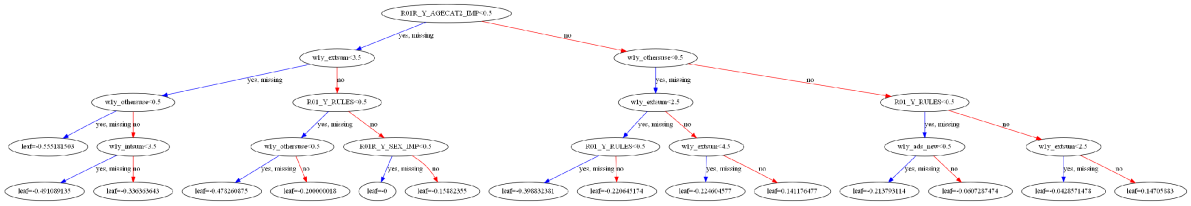
In [179]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.
3,
                                                    random_state = 0)

model = XGBClassifier(eta = 0.3, max_depth = 4, eval_metric = 'logloss')
model.fit(X_train, y_train)
```

```
C:\Users\applesd\Anaconda3\lib\site-packages\xgboost\sklearn.py:888: Us
erWarning: The use of label encoder in XGBClassifier is deprecated and
will be removed in a future release. To remove this warning, do the fol
lowing: 1) Pass option use_label_encoder=False when constructing XGBCla
ssifier object; and 2) Encode your labels (y) as integers starting with
0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
```

Out[179]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, eta=0.3,
              eval_metric='logloss', gamma=0, gpu_id=-1, importance_typ
e='gain',
              interaction_constraints='', learning_rate=0.300000012,
              max_delta_step=0, max_depth=4, min_child_weight=1, missin
g=nan,
              monotone_constraints='()', n_estimators=100, n_jobs=4,
              num_parallel_tree=1, random_state=0, reg_alpha=0, reg_lam
bda=1,
              scale_pos_weight=1, subsample=1, tree_method='exact',
              validate_parameters=1, verbosity=None)
```

In [180]:
```
fig, ax = plt.subplots(figsize = (50, 50))
xgb.plot_tree(model, ax = ax)
plt.show()
```



In [181]:
```
y_pred = model.predict(X_test)
predictions = [round(value) for value in y_pred]
accuracy = accuracy_score(y_test, predictions)
accuracy
```

Out[181]:
```
0.797678916827853
```

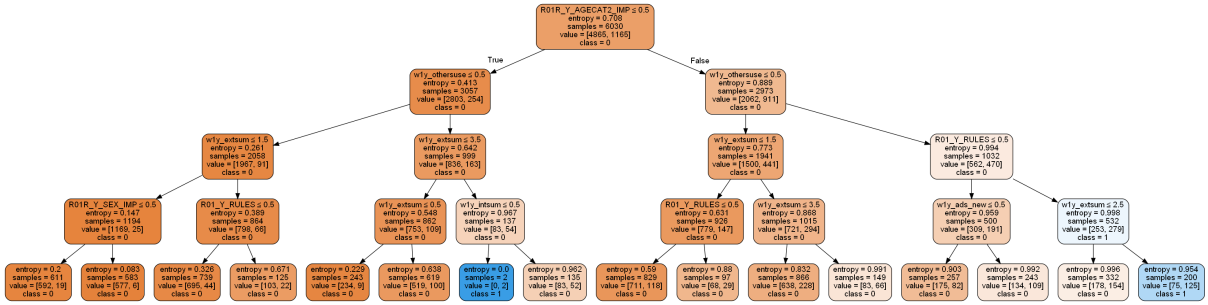
In [182]:
```
clf = DecisionTreeClassifier(criterion = 'entropy', max_depth = 4)
clf = clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
```

In [183]:
```
print('Accuracy:', metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.8030947775628626
```

In [184]:
```
from six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus
dot_data = StringIO()
export_graphviz(clf, out_file = dot_data, filled = True, rounded = True,
                special_characters = True, feature_names = features[0:12
],
                 class_names = ['0', '1'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('tobacco use.png')
Image(graph.create_png())
```

Out[184]:



In [7]:
```
import statsmodels.api as sm
from statsmodels.formula.api import logit



train, test = train_test_split(df, test_size = 0.3, random_state = 0)
formula = ('smoke ~ R01R_Y_AGECAT2_IMP + R01R_RACE_ETHN + R01R_Y_SEX_IMP
+ R01R_Y_PM0002 + w1y_impuls + w1y_intsum + w1y_extsum + w1y_othersuse +
R01_YX0500 + w1y_ads_new + w1_access_bi + R01_Y_RULES')
model_lgt = logit(formula, data = train).fit()
```

```
Optimization terminated successfully.
         Current function value: 0.399238
         Iterations 7
```

In [8]:
```
model_lgt.summary()
```

Out[8]:

Logit Regression Results

Dep. Variable:	smoke	**No. Observations:**	6030
Model:	Logit	**Df Residuals:**	6017
Method:	MLE	**Df Model:**	12
Date:	Mon, 17 May 2021	**Pseudo R-squ.:**	0.1866
Time:	09:07:50	**Log-Likelihood:**	-2407.4
converged:	True	**LL-Null:**	-2959.7
		LLR p-value:	5.916e-229

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
Intercept	-3.3501	0.149	-22.516	0.000	-3.642	-3.058
R01R_Y_AGECAT2_IMP	1.6373	0.084	19.460	0.000	1.472	1.802
R01R_RACE_ETHN	-0.0691	0.075	-0.921	0.357	-0.216	0.078
R01R_Y_SEX_IMP	-0.3884	0.076	-5.083	0.000	-0.538	-0.239
R01R_Y_PM0002	-0.1618	0.076	-2.117	0.034	-0.311	-0.012
w1y_impuls	-0.1449	0.059	-2.462	0.014	-0.260	-0.030
w1y_intsum	0.0924	0.029	3.220	0.001	0.036	0.149
w1y_extsum	0.3058	0.032	9.588	0.000	0.243	0.368
w1y_othersuse	0.8246	0.081	10.177	0.000	0.666	0.983
R01_YX0500	0.0576	0.073	0.789	0.430	-0.085	0.201
w1y_ads_new	0.3013	0.074	4.062	0.000	0.156	0.447
w1_access_bi	-0.2935	0.075	-3.897	0.000	-0.441	-0.146
R01_Y_RULES	0.6675	0.084	7.900	0.000	0.502	0.833

In [9]:
```
pred_lgt = model_lgt.predict(exog = test)
cutoff = 0.5
y_pred = np.where(pred_lgt > cutoff, 1, 0)
y_true = test['smoke']
accuracy_score(y_true, y_pred)
```

Out[9]: 0.8034816247582205

In [10]:
```
score = []
score = -3.3501 + 1.6373 * df['R01R_Y_AGECAT2_IMP'] + -0.0691 * df['R01R
_RACE_ETHN'] - 0.3884 * df['R01R_Y_SEX_IMP'] - 0.1618 * df['R01R_Y_PM000
2'] - 0.1449 * df['w1y_impuls'] + 0.0924 * df['w1y_intsum'] + 0.3058 * d
f['w1y_extsum'] + 0.8246 * df['w1y_othersuse'] + 0.0576 * df['R01_YX050
0'] + 0.3013 * df['w1y_ads_new'] - 0.2935 * df['w1_access_bi'] + 0.6675
* df['R01_Y_RULES']
df['score'] = score
```

In [11]:
```
X_new = df.drop('smoke', axis = 1)
y_new = df['smoke']
features_new = ['R01R_Y_AGECAT2_IMP', 'R01R_RACE_ETHN', 'R01R_Y_SEX_IMP'
, 'R01R_Y_PM0002',
'w1y_impuls', 'w1y_intsum', 'w1y_extsum', 'w1y_othersuse', 'R01_YX0500',
'w1y_ads_new', 'w1_access_bi','R01_Y_RULES', 'score']
```

In [12]:
```
#XGBoost
param = {'max_depth': 9, 'eta': 1, 'objective': 'binary:logistic',
          'eval_metric' : 'logloss'}
allerrors = np.zeros((len(tree_depth_choices),len(learning_rate_choices
), 100))

for i, treedepth in enumerate(tree_depth_choices, 0):
    param['max_depth']=treedepth
    for j, lr in enumerate(learning_rate_choices, 0):
        param['eta']=lr
        allerrors[i, j] = cvfunc(X_new.values, y_new.values, param)
```

In [13]:
```
x=(np.unravel_index(np.argmin(allerrors), allerrors.shape))
param['max_depth'] = tree_depth_choices[x[0]]
param['eta'] = learning_rate_choices[x[1]]
max_trees = possible_trees[x[2]]
```

In [14]:
```
param
```

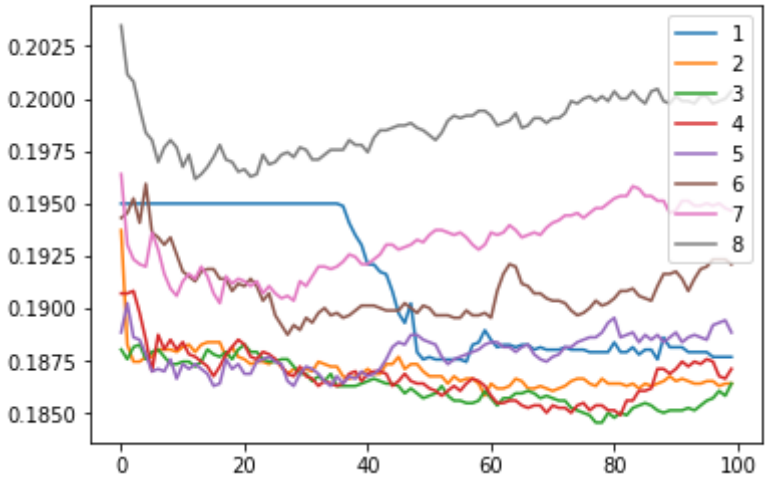
Out[14]:
```
{'max_depth': 3,
 'eta': 0.1,
 'objective': 'binary:logistic',
 'eval_metric': 'logloss'}
```

In [15]:
```
max_trees
```

Out[15]:
```
78
```

In [16]:
```
for j in range(8):
    plt.plot(allerrors[j, 3, :], label=tree_depth_choices[j])
plt.legend()
```

Out[16]: <matplotlib.legend.Legend at 0xbd26be62e8>



In [32]:
```
X_train, X_test, y_train, y_test = train_test_split(X_new, y_new, test_s
ize = 0.3,
                                                    random_state = 0)
dtrain = xgb.DMatrix(X_train, y_train, feature_names = features_new)
dtest = xgb.DMatrix(X_test, y_test, feature_names = features_new)
params = {'max_depth': 3, 'eta': 0.1, 'objective': 'binary:logistic',
          'eval_metric': 'logloss'}

evallist = [(dtrain, 'train')]
iteration = 1000
bst = xgb.train(params, dtrain, iteration, evallist)
```

```
[969]   train-logloss:0.30591
[970]   train-logloss:0.30591
[971]   train-logloss:0.30588
[972]   train-logloss:0.30587
[973]   train-logloss:0.30584
[974]   train-logloss:0.30583
[975]   train-logloss:0.30582
[976]   train-logloss:0.30579
[977]   train-logloss:0.30568
[978]   train-logloss:0.30564
[979]   train-logloss:0.30556
[980]   train-logloss:0.30553
[981]   train-logloss:0.30547
[982]   train-logloss:0.30546
[983]   train-logloss:0.30544
[984]   train-logloss:0.30543
[985]   train-logloss:0.30538
[986]   train-logloss:0.30533
[987]   train-logloss:0.30532
[988]   train-logloss:0.30531
[989]   train-logloss:0.30531
[990]   train-logloss:0.30515
[991]   train-logloss:0.30510
[992]   train-logloss:0.30501
[993]   train-logloss:0.30497
[994]   train-logloss:0.30496
[995]   train-logloss:0.30496
[996]   train-logloss:0.30491
[997]   train-logloss:0.30484
[998]   train-logloss:0.30478
[999]   train-logloss:0.30473
```

In [33]:
```
y_pred = bst.predict(dtest, ntree_limit = 78)
predictions = [round(value) for value in y_pred]
accuracy_score(y_test, predictions)
```

Out[33]: 0.8011605415860735

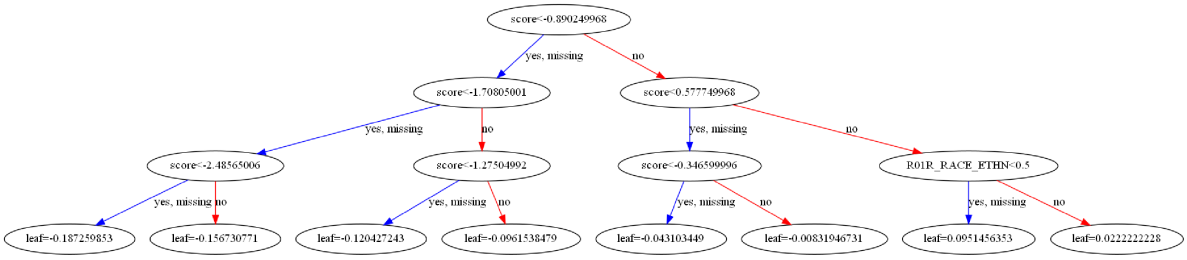
In [30]:
```
X_train, X_test, y_train, y_test = train_test_split(X_new, y_new, test_s
ize = 0.3,
                                                    random_state = 0)

model = XGBClassifier(eta = 0.1, max_depth = 3, eval_metric = 'logloss')
model.fit(X_train, y_train)
```

```
C:\Users\applesd\Anaconda3\lib\site-packages\xgboost\sklearn.py:888: Us
erWarning: The use of label encoder in XGBClassifier is deprecated and
will be removed in a future release. To remove this warning, do the fol
lowing: 1) Pass option use_label_encoder=False when constructing XGBCla
ssifier object; and 2) Encode your labels (y) as integers starting with
0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
```

Out[30]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, eta=0.1,
              eval_metric='logloss', gamma=0, gpu_id=-1, importance_typ
e='gain',
              interaction_constraints='', learning_rate=0.100000001,
              max_delta_step=0, max_depth=3, min_child_weight=1, missin
g=nan,
              monotone_constraints='()', n_estimators=100, n_jobs=4,
              num_parallel_tree=1, random_state=0, reg_alpha=0, reg_lam
bda=1,
              scale_pos_weight=1, subsample=1, tree_method='exact',
              validate_parameters=1, verbosity=None)
```

In [23]:
```
fig, ax = plt.subplots(figsize = (50, 50))
xgb.plot_tree(model, ax = ax)
plt.show()
```



In [31]:
```
y_pred = model.predict(X_test)
predictions = [round(value) for value in y_pred]
accuracy = accuracy_score(y_test, predictions)
accuracy
```

Out[31]:
```
0.8019342359767891
```

In [36]:
```
clf = DecisionTreeClassifier(criterion = 'entropy', max_depth = 3)
clf = clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print('Accuracy:', metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.8061895551257253
```

In [27]:
```
from six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus
dot_data = StringIO()
export_graphviz(clf, out_file = dot_data, filled = True, rounded = True,
                special_characters = True, feature_names = features_new,
                 class_names = ['0', '1'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('tobacco use.png')
Image(graph.create_png())
```

Out[27]: