Project 2 - Stroke Prediction

MaryGrace Gozzi, Victoria DaRosa, Tu-Yen Dang, Charlie Perez

DS 3001: Machine Learning

12/07/2023

Summary

The goal driving this project was to create a model that predicted the likelihood that different patients were going to suffer a stroke. A stroke, a serious medical emergency resulting from a blocked or ruptured artery, can be fatal if not treated immediately, and often causes lasting health problems. The severity of this condition emphasizes the importance of predicting stroke likelihood, because preventative measures and increased awareness have the potential to save lives. To build our model, we took both demographic information as well as pre-existing health conditions into consideration. Some of the most notable factors included 'heart-disease' and 'hypertension', both dummy variables representing whether or not a patient has heart disease or high blood pressure. Additional variables included 'avg_glucose_level' and 'bmi', both numeric variables representing their respective measures of health. The 'stroke' variable, the predictive goal of the model, was also a binary variable. A value of 1 indicated that the patient had suffered a stroke during the data sample period, and value of 0 indicated that the patient had not suffered a stroke during the same period. Based on the provided variables and nature of the project's guiding question, we chose to create a classification tree to determine the likelihood of patient strokes. The most accurate classification trees had lower depth values. The trees with depth values 1 through 3 had the highest R squared value and the lowest root mean square error (RMSE) values, equalling -0.040 and 0.197 respectively, and an accuracy of 0.961, but these

models simply predicted “stroke not likely” 100% of the time[1]. When the depth value increased beyond 3, the R squared value decreased and the RMSE value increased, indicating that the model had been overfit for the question, but ultimately our “best model” that retained some level of practicality (actually predicting a stroke, that is) was at depth 6 and minimum leaf samples of 2, with an R squared of -0.150, an accuracy of 0.957, and an RMSE of 0.207. Working through each variable and determining the impacts of each patient’s identity and health, the trees split patients into two categories, which are effectively ‘stroke likely and ‘stroke not likely’. Cases counted in the first column of each tree node matrix are considered ‘stroke not likely’, and cases counted in the second column of each tree node matrix are considered ‘stroke likely’.

Data

The dataset used in this project examines the health of numerous individuals in relation to strokes. The given data was used to predict the likelihood an individual has had a stroke. In order to do this effectively, the data was split into two different files: a training dataset and a testing dataset. With the two different datasets, the data can be trained with the training dataset, and predictions can be made with the testing data set. There are multiple key variables within the dataset. The numeric variables include age, hypertension, heart_disease, bmi, avg_glucose_level, which analyze the medical history of the individuals. The categorical variables include work_type, Residence_type, and smoking_status, which analyze different aspects of individuals' demographics and lifestyle choices. The stroke variable, indicating whether an individual suffered a stroke during the sample period, serves as a focal point for constructing a regression

[1] CP - On this fine Thursday morning, I woke up and decided to look through the documentation for scikit’s model.score() function, having at some point during my slumber realized what horrors I would find. Scikit confirmed that for classification models, the .score() function returns the accuracy, not the coefficient of determination. I have tried to rectify this everywhere in the paper, and specify accuracy versus r-squared, but I apologize if I missed a place. I have also committed a fully separate python notebook with the updated scoring (using the .metrics r2_score function) that you may use for reference. Luckily, RMSE did not need adjustment, so for the purposes of solving the question the project asked of us we did not need to adjust anything.

model. This model involves regressing the stroke variable against all other variables, which gives a broad view of how strokes play out in this dataset.

To begin cleaning the data, there were two major things to change about the columns. Firstly, all of the column names were changed to lowercase to maintain consistency (‘Residence_type’ became ‘residence_type’). Then, the first two columns were removed because they were not useful (‘Unnamed: 0’ and ‘id’). After this, the data sets were checked for null values and outliers. It was found that only the ‘bmi’ column had null values in each data set, where the training set had 159 null values, while the testing set had 42 null values. Combined, that totaled to 201 null values out of 5,110 values, which was fairly insignificant. So, the null values were replaced with the average value of ‘bmi’ to ensure the data was not skewed.

Despite this being the only technical null values in the dataset, the ‘smoking_status’ dataset also had a lot of unusable values. This was found when looking for string outliers, where it was discovered that there were over 1200 (almost 33%) ‘unknown’ recordings in the column. This was an issue because though a value was ‘recorded’, it did not provide any information, and was the equivalent of a null value. It would be unfair to move all these values into one category since it represented such a large portion of the data. So, it was decided that nothing would be done to this. There was no explanation for why there were so many ‘Unknown’ values, however this is also one of the more delicate subjects (similar to how the ‘bmi’ may be a sensitive topic to participants), so people may not feel comfortable stating their smoking status.

Outliers were also looked for in the process of data wrangling. For continuous numeric columns (‘bmi’, ‘avg_glucose_level, age’), boxplots were depicted to see if and how many outliers were present. The ‘age’ column had no outliers, but for ‘bmi’ and ‘avg_glucose_level’, there were a significant number of outliers in the boxplot. It was decided that because of how

many they were (and how they were concentrated around the same areas), they would be kept in case they provided indication of a stroke. For the discrete numeric columns (‘heart_disease’, ‘hypertension’, ‘stroke’), the ‘.value_counts()’ function was used to see if there were any outliers (values that weren’t recorded as 0 or 1). It was found that there were no outliers for the section.

Finally, string outliers were checked. After looking through each column’s unique value, it was decided that the values were not consistent with their capitalization, so all characters became lowercase. This was also when the ‘smoking_status’ issue was discovered with the unknown values. Furthermore, considering the context of the lab, it was assumed that ‘formerly smoked’ would have a similar/the same impact on the likelihood of having a stroke as the ‘smokes’ value was, so they were combined into one variable. It was also realized that the ‘ever_married’ values were binary, similar to the discrete numeric columns. So, to keep the consistent format, the column became numeric and the values were changed so that 1 would represent ‘yes’, and 0 would represent ‘no’.

Results

Attempts at modeling a classification tree were largely unsuccessful as positive predictive tools. The R^2 was minimized (-0.040 or -0.051, depending on how the null BMI values were treated) when the model simply predicted “no stroke” 100% of the time.

Our first course of action was to retain all variables as potential predictors. After the data had been cleaned, we used Scikit’s decision tree classification tool to create several tree models of varying depth. We quickly discovered that, because of the rarity with which strokes occurred and thus the heavy weight toward “no stroke” in the training data, the shallower decision tree depths, ranging from a depth of 1 to a depth of 4, would exclusively predict “no stroke”, or ‘0’, when fitted to the training data. This means that under our model, every time a stroke occurred, it

would be an instance of a false negative. We determined that this was not an acceptable predictive model and resolved to attempt further depths.

However, when we tried to fit tree depths of greater than 4, we quickly encountered overfit problems for much the same reason: the tree became produced a greater percentage of false positives than it did rectify true negatives. See the crosstabulations for reference: the first table is a crosstabulation of the prediction versus the actual for depth=4 with no adjustments made to the variables, and the second is for depth=7, the first depth at which a stroke was accurately predicted by the model. By this depth, as is evident in the table, the overall accuracy of the model had decreased, from 95.1% to 93.8%. We resolved to attempt several other modifications to the training data to attempt to find a more predictive model.

col_0	0
stroke	
0	973
1	50

col_0	0	1
stroke		
0	959	14
1	49	1

First, we chose to eliminate the null BMI values instead of replacing them with the mean BMI, in hopes of making BMI a more useful predictive tool; weighting BMI excessively toward the mean without regard for other metrics, such as heart disease, could be creating a problem for the decision tree. We also noticed that age was heavily affecting the tree's fitting. We removed the age variable because it proved to be ineffective at any meaningful depth; it simply biased the classifier further toward predicting "no stroke".[2] See right

depth 8

col_0	0	1
stroke		
0	928	15
1	37	1

0.9469928644240571

0.2302327856234705

depth 8

col_0	0	1
stroke		
0	927	16
1	33	5

0.9500509683995922

0.22349279988493534

[2] Note - CP: I also reorganized the "smoking status" variable and eliminated variables that were completely irrelevant in the original classification trees (marital status, work type, residence type, and gender), but this was merely for convenience when viewing the dataframe and it is highly unlikely to have had any effect on predictor performance.

for examples of crosstabulation of actual and predicted values by a tree with a depth constraint of 8, the top with age included and the bottom with age not included – this also includes accuracy scores (top number) and RMSE (bottom).

From further analysis, we determined that eliminating the rows with null BMI values does affect the answers, and thus that our imputation was at least marginally flawed. Interestingly, removing null BMI values resulted in the removal of a disproportionately high number of observed instances of strokes – that is to say, on average, if the individual was missing a BMI value, they were unusually likely to have a stroke. As an example of how extreme this influence is, in the test set, of the 42 null BMI values, 30 corresponded to a "stroke" observation of 0 (non-stroke) and 12 corresponded to 1 (stroke) – a ratio of 2.5:1, compared to an average ratio of 24.8:1 across the remainder of the dataset. Unfortunately, lacking more information about why these values are missing, we are unable to draw any conclusions as to the meaning behind this, and thought that this relatively small subset of the BMI variable, if left in, would disproportionately affect the effectiveness of the classifier as a whole while not contributing meaningfully to its usefulness as a predictive tool, particularly if we continued to impute using the mean BMI.

With this refined dataset, we produced a similar distribution of trees with depths varying from 1 to 7. In this case, though the R-squared and RMSE of the model in the depth range 1-3 is marginally better (-0.040 r^2 or 0.961 accuracy), this is completely insignificant because the model still predicts "no stroke" 100% of the time – the training and testing data has simply been altered to include a proportionately lower number of positives, as discussed in the BMI section above. Depths 4-6, however, remained more accurate in this model than when using the original

BMI variable. We also tested different constraints on the minimum number of samples per leaf to attempt to prevent overfitting. However, this had a negligible result.

As mentioned above, we faced a dilemma in deciding which model to submit as the optimal classification tree model. Our "best model" by the standard of R-squared and RMSE is a model which simply says 100% of the time that a stroke will not occur. The "best models" that actually successfully predicts a stroke are any models at a depth of 4 or the model at depth 6 and minimum leaf samples of 2. The latter is our chosen model because although it is not technically more accurate, it does feature a higher number of true positives. Assuming that the goal of this project is to predict strokes, we prefer a model that does this as often as possible. At depth 6 and minimum leaf samples of 2, the model predicts 935 (95.3%) true negatives, 34 (3.47%) false negatives, 8 (0.815%) false positives, and 4 (0.408%) true positives, for an accuracy of 95.7% (R^2 of -0.150) and an RMSE of 0.2069. This model is evidently not an excellent predictor by the metric of R^2, but further depths and greater minimum leaf samples generally serve to increase the number of false positives without a proportionate increase in true positives. See the Appendix for the complete text rendering of the model.[3]

Conclusion

We can draw several interesting conclusions from our analysis of stroke prediction. The first is that while there is no single predictor or even combination of predictors that guarantees a stroke, there are several factors that may signal higher likelihood of a stroke, including smoking status, hypertension, heart disease, high blood sugar, and BMI. We would consider the possibility that a linear regression model or other model which has the capacity to treat the likelihood of a stroke as laying on a continuum (example: instead of "stroke" or "no stroke", the model returns

[3] Incredibly hard to decipher, we know, but at least legible (unlike Jupyter's rendering of the tree itself). See the python notebook (near the bottom) if you would like to investigate in more depth.

“75% likelihood of stroke”, “10% likelihood of stroke”, etc.) may be a more practically useful tool in the field of medicine, if less “accurate”. Ultimately, this project proved what we all know: there is no single predictor of a stroke, and completely “healthy” people can experience them, as can rather “unhealthy” people avoid strokes altogether. However, as evidenced by the results of this project, it is certainly possible to create a predictive model that identifies people “more likely” to have a stroke – the final model, out of 12 predicted strokes, was right 33% of the time, compared to the far lower rate of prevalence of strokes in the test sample as a whole (3.47%).

Intriguingly, our analysis revealed that age had a surprisingly insignificant impact as a predictor in the tree classification model; the R^2 and RMSE values did not change by a large amount when age was included versus not included in the training data, and the inclusion of the age variable as a predictor contributed to the overfit and false negative problem. We would argue that aging may increase the likelihood of these other risk factors also being present, and thus operates as a semi-confounding variable – a surprise for all of us. It suggests that there may be significant correlation between age and other variables we treated as “predictors”.

There will be criticisms of this marvelous, beautiful, gorgeous decision tree classification model as “not robust”, or even (dare we say it) “having a worse coefficient of determination than the freaking example.” To this we respond: “Well, yeah, but would you have preferred the model that only ever tells you that you’re not at risk for a stroke? At least this way, 4 of you will have had a heads up. Also, Terry said we couldn’t use random forests because it ‘wasn’t sportsmanlike.’”

Appendix:

```
[Text(0.564935064935065, 0.9285714285714286, 'x[2] <= 162.045\ngini =
0.083\nsamples = 3928\nvalue = [3757, 171]'),
 Text(0.35064935064935066, 0.7857142857142857, 'x[0] <= 0.5\ngini =
0.061\nsamples = 3456\nvalue = [3347, 109]'),
 Text(0.19805194805194806, 0.6428571428571429, 'x[1] <= 0.5\ngini =
0.051\nsamples = 3214\nvalue = [3129, 85]'),
 Text(0.1038961038961039, 0.5, 'x[3] <= 22.75\ngini = 0.045\nsamples =
3107\nvalue = [3035, 72]'),
 Text(0.05194805194805195, 0.35714285714285715, 'x[4] <= 1.5\ngini =
0.013\nsamples = 770\nvalue = [765, 5]'),
 Text(0.025974025974025976, 0.21428571428571427, 'x[4] <= 0.5\ngini =
0.009\nsamples = 666\nvalue = [663, 3]'),
 Text(0.012987012987012988, 0.07142857142857142, 'gini = 0.021\nsamples =
186\nvalue = [184, 2]'),
 Text(0.03896103896103896, 0.07142857142857142, 'gini = 0.004\nsamples =
480\nvalue = [479, 1]'),
 Text(0.07792207792207792, 0.21428571428571427, 'x[3] <= 21.15\ngini =
0.038\nsamples = 104\nvalue = [102, 2]'),
 Text(0.06493506493506493, 0.07142857142857142, 'gini = 0.0\nsamples =
56\nvalue = [56, 0]'),
 Text(0.09090909090909091, 0.07142857142857142, 'gini = 0.08\nsamples =
48\nvalue = [46, 2]'),
 Text(0.15584415584415584, 0.35714285714285715, 'x[3] <= 32.05\ngini =
0.056\nsamples = 2337\nvalue = [2270, 67]'),
 Text(0.12987012987012986, 0.21428571428571427, 'x[3] <= 31.95\ngini =
0.069\nsamples = 1554\nvalue = [1498, 56]'),
 Text(0.11688311688311688, 0.07142857142857142, 'gini = 0.067\nsamples =
1539\nvalue = [1486, 53]'),
 Text(0.14285714285714285, 0.07142857142857142, 'gini = 0.32\nsamples =
15\nvalue = [12, 3]'),
 Text(0.18181818181818182, 0.21428571428571427, 'x[2] <= 84.94\ngini =
0.028\nsamples = 783\nvalue = [772, 11]'),
 Text(0.16883116883116883, 0.07142857142857142, 'gini = 0.011\nsamples =
362\nvalue = [360, 2]'),
 Text(0.19480519480519481, 0.07142857142857142, 'gini = 0.042\nsamples =
421\nvalue = [412, 9]'),
 Text(0.2922077922077922, 0.5, 'x[2] <= 111.54\ngini = 0.213\nsamples =
107\nvalue = [94, 13]'),
 Text(0.2597402597402597, 0.35714285714285715, 'x[3] <= 21.9\ngini =
0.143\nsamples = 90\nvalue = [83, 7]'),
 Text(0.23376623376623376, 0.21428571428571427, 'x[2] <= 90.83\ngini =
0.5\nsamples = 4\nvalue = [2, 2]'),
 Text(0.22077922077922077, 0.07142857142857142, 'gini = 0.0\nsamples =
2\nvalue = [0, 2]'),
 Text(0.24675324675324675, 0.07142857142857142, 'gini = 0.0\nsamples =
2\nvalue = [2, 0]'),
 Text(0.2857142857142857, 0.21428571428571427, 'x[2] <= 60.775\ngini =
0.11\nsamples = 86\nvalue = [81, 5]'),
 Text(0.2727272727272727, 0.07142857142857142, 'gini = 0.375\nsamples =
4\nvalue = [3, 1]'),
 Text(0.2987012987012987, 0.07142857142857142, 'gini = 0.093\nsamples =
82\nvalue = [78, 4]'),
```

```
 Text(0.3246753246753247, 0.35714285714285715, 'x[3] <= 26.45\ngini =
0.457\nsamples = 17\nvalue = [11, 6]'),
 Text(0.3116883116883117, 0.21428571428571427, 'gini = 0.0\nsamples =
2\nvalue = [0, 2]'),
 Text(0.33766233766233766, 0.21428571428571427, 'x[3] <= 33.7\ngini =
0.391\nsamples = 15\nvalue = [11, 4]'),
 Text(0.3246753246753247, 0.07142857142857142, 'gini = 0.278\nsamples =
12\nvalue = [10, 2]'),
 Text(0.35064935064935066, 0.07142857142857142, 'gini = 0.444\nsamples =
3\nvalue = [1, 2]'),
 Text(0.5032467532467533, 0.6428571428571429, 'x[2] <= 76.375\ngini =
0.179\nsamples = 242\nvalue = [218, 24]'),
 Text(0.461038961038961, 0.5, 'x[3] <= 27.45\ngini = 0.296\nsamples =
72\nvalue = [59, 13]'),
 Text(0.4155844155844156, 0.35714285714285715, 'x[2] <= 66.5\ngini =
0.475\nsamples = 18\nvalue = [11, 7]'),
 Text(0.38961038961038963, 0.21428571428571427, 'x[3] <= 24.45\ngini =
0.278\nsamples = 12\nvalue = [10, 2]'),
 Text(0.37662337662337664, 0.07142857142857142, 'gini = 0.0\nsamples =
7\nvalue = [7, 0]'),
 Text(0.4025974025974026, 0.07142857142857142, 'gini = 0.48\nsamples =
5\nvalue = [3, 2]'),
 Text(0.44155844155844154, 0.21428571428571427, 'x[4] <= 2.5\ngini =
0.278\nsamples = 6\nvalue = [1, 5]'),
 Text(0.42857142857142855, 0.07142857142857142, 'gini = 0.0\nsamples =
4\nvalue = [0, 4]'),
 Text(0.45454545454545453, 0.07142857142857142, 'gini = 0.5\nsamples =
2\nvalue = [1, 1]'),
 Text(0.5064935064935064, 0.35714285714285715, 'x[2] <= 72.93\ngini =
0.198\nsamples = 54\nvalue = [48, 6]'),
 Text(0.4935064935064935, 0.21428571428571427, 'x[2] <= 72.07\ngini =
0.245\nsamples = 42\nvalue = [36, 6]'),
 Text(0.4805194805194805, 0.07142857142857142, 'gini = 0.184\nsamples =
39\nvalue = [35, 4]'),
 Text(0.5064935064935064, 0.07142857142857142, 'gini = 0.444\nsamples =
3\nvalue = [1, 2]'),
 Text(0.5194805194805194, 0.21428571428571427, 'gini = 0.0\nsamples =
12\nvalue = [12, 0]'),
 Text(0.5454545454545454, 0.5, 'x[3] <= 20.3\ngini = 0.121\nsamples =
170\nvalue = [159, 11]'),
 Text(0.5324675324675324, 0.35714285714285715, 'gini = 0.5\nsamples =
2\nvalue = [1, 1]'),
 Text(0.5584415584415584, 0.35714285714285715, 'x[4] <= 2.5\ngini =
0.112\nsamples = 168\nvalue = [158, 10]'),
 Text(0.5454545454545454, 0.21428571428571427, 'x[3] <= 34.25\ngini =
0.141\nsamples = 131\nvalue = [121, 10]'),
 Text(0.5324675324675324, 0.07142857142857142, 'gini = 0.191\nsamples =
84\nvalue = [75, 9]'),
 Text(0.5584415584415584, 0.07142857142857142, 'gini = 0.042\nsamples =
47\nvalue = [46, 1]'),
 Text(0.5714285714285714, 0.21428571428571427, 'gini = 0.0\nsamples =
37\nvalue = [37, 0]'),
 Text(0.7792207792207793, 0.7857142857142857, 'x[1] <= 0.5\ngini =
0.228\nsamples = 472\nvalue = [410, 62]'),
```

```
 Text(0.6623376623376623, 0.6428571428571429, 'x[2] <= 162.235\ngini =
0.191\nsamples = 402\nvalue = [359, 43]'),
 Text(0.6493506493506493, 0.5, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]'),
 Text(0.6753246753246753, 0.5, 'x[0] <= 0.5\ngini = 0.184\nsamples =
400\nvalue = [359, 41]'),
 Text(0.6233766233766234, 0.35714285714285715, 'x[3] <= 26.05\ngini =
0.132\nsamples = 295\nvalue = [274, 21]'),
 Text(0.5974025974025974, 0.21428571428571427, 'x[4] <= 2.5\ngini =
0.033\nsamples = 60\nvalue = [59, 1]'),
 Text(0.5844155844155844, 0.07142857142857142, 'gini = 0.0\nsamples =
50\nvalue = [50, 0]'),
 Text(0.6103896103896104, 0.07142857142857142, 'gini = 0.18\nsamples =
10\nvalue = [9, 1]'),
 Text(0.6493506493506493, 0.21428571428571427, 'x[3] <= 26.15\ngini =
0.156\nsamples = 235\nvalue = [215, 20]'),
 Text(0.6363636363636364, 0.07142857142857142, 'gini = 0.5\nsamples =
2\nvalue = [1, 1]'),
 Text(0.6623376623376623, 0.07142857142857142, 'gini = 0.15\nsamples =
233\nvalue = [214, 19]'),
 Text(0.7272727272727273, 0.35714285714285715, 'x[3] <= 28.25\ngini =
0.308\nsamples = 105\nvalue = [85, 20]'),
 Text(0.7012987012987013, 0.21428571428571427, 'x[2] <= 224.195\ngini =
0.453\nsamples = 26\nvalue = [17, 9]'),
 Text(0.6883116883116883, 0.07142857142857142, 'gini = 0.476\nsamples =
23\nvalue = [14, 9]'),
 Text(0.7142857142857143, 0.07142857142857142, 'gini = 0.0\nsamples =
3\nvalue = [3, 0]'),
 Text(0.7532467532467533, 0.21428571428571427, 'x[2] <= 241.69\ngini =
0.24\nsamples = 79\nvalue = [68, 11]'),
 Text(0.7402597402597403, 0.07142857142857142, 'gini = 0.193\nsamples =
74\nvalue = [66, 8]'),
 Text(0.7662337662337663, 0.07142857142857142, 'gini = 0.48\nsamples =
5\nvalue = [2, 3]'),
 Text(0.8961038961038961, 0.6428571428571429, 'x[2] <= 208.175\ngini =
0.396\nsamples = 70\nvalue = [51, 19]'),
 Text(0.8441558441558441, 0.5, 'x[4] <= 2.5\ngini = 0.208\nsamples =
34\nvalue = [30, 4]'),
 Text(0.8181818181818182, 0.35714285714285715, 'x[2] <= 177.025\ngini =
0.071\nsamples = 27\nvalue = [26, 1]'),
 Text(0.8051948051948052, 0.21428571428571427, 'x[0] <= 0.5\ngini =
0.375\nsamples = 4\nvalue = [3, 1]'),
 Text(0.7922077922077922, 0.07142857142857142, 'gini = 0.0\nsamples =
2\nvalue = [2, 0]'),
 Text(0.8181818181818182, 0.07142857142857142, 'gini = 0.5\nsamples =
2\nvalue = [1, 1]'),
 Text(0.8311688311688312, 0.21428571428571427, 'gini = 0.0\nsamples =
23\nvalue = [23, 0]'),
 Text(0.8701298701298701, 0.35714285714285715, 'x[2] <= 195.74\ngini =
0.49\nsamples = 7\nvalue = [4, 3]'),
 Text(0.8571428571428571, 0.21428571428571427, 'gini = 0.0\nsamples =
2\nvalue = [0, 2]'),
 Text(0.8831168831168831, 0.21428571428571427, 'x[3] <= 31.7\ngini =
0.32\nsamples = 5\nvalue = [4, 1]'),
```

```
 Text(0.8701298701298701, 0.07142857142857142, 'gini = 0.5\nsamples =
2\nvalue = [1, 1]'),
 Text(0.8961038961038961, 0.07142857142857142, 'gini = 0.0\nsamples =
3\nvalue = [3, 0]'),
 Text(0.948051948051948, 0.5, 'x[3] <= 27.55\ngini = 0.486\nsamples =
36\nvalue = [21, 15]'),
 Text(0.922077922077922, 0.35714285714285715, 'x[2] <= 236.67\ngini =
0.18\nsamples = 10\nvalue = [9, 1]'),
 Text(0.9090909090909091, 0.21428571428571427, 'gini = 0.0\nsamples =
7\nvalue = [7, 0]'),
 Text(0.935064935064935, 0.21428571428571427, 'gini = 0.444\nsamples =
3\nvalue = [2, 1]'),
 Text(0.974025974025974, 0.35714285714285715, 'x[2] <= 239.9\ngini =
0.497\nsamples = 26\nvalue = [12, 14]'),
 Text(0.961038961038961, 0.21428571428571427, 'x[3] <= 32.75\ngini =
0.496\nsamples = 22\nvalue = [12, 10]'),
 Text(0.948051948051948, 0.07142857142857142, 'gini = 0.463\nsamples =
11\nvalue = [4, 7]'),
 Text(0.974025974025974, 0.07142857142857142, 'gini = 0.397\nsamples =
11\nvalue = [8, 3]'),
 Text(0.987012987012987, 0.21428571428571427, 'gini = 0.0\nsamples =
4\nvalue = [0, 4]')]
```