Modeling the Health of Trees in New York City: A Report

Overview

Problem

Keeping track of the trees growing in the City of New York is a huge responsibility. Since there are over half a million trees throughout the city, keeping tabs on each one can prove to be a challenge. There are over 600,000 trees in the city, and every year, some get sick or die due to neglect and a lack of proper attention. In order to minimize this number, we will use tree census data provided by NYC Parks & Rec to create a model that predicts (strength of correlation) the health of a tree based on its location, trunk diameter, tree type, root condition, trunk condition, and branch condition (to list a few). That way, volunteers and city officials can respond in time to care for the ailing tree. The model will go a step further by taking into consideration where deceased trees are located and provide suggestions for where to grow new trees in order to maintain an evergreen city. We need to identify where and which trees have poor health.

Client

The client is the New York City Parks and Recreation Center, who are responsible for keeping track of all trees within city limits. They provided a tree census gathered from 2015 as the datasource for creating a model that predicts the health of trees grown in the city. This helps Parks and Rec locate ailing trees quickly and efficiently and determine where new trees need to be planted.

Data

The dataset comes from NYC Open Data and is collected by volunteers and staff from the NYC Parks and Rec Center.

Data Wrangling

The purpose of this step is to thoroughly analyze and clean the raw data prior to use. Things to address here include missing values, duplicates, adding / subtracting columns, and outliers. Data types are mostly categorical with a few numerical variables. The dataset is fairly large, with 683,788 rows and 45 columns.

Removing duplicates and null values

After importing the dataset into a Jupyter notebook and looking at the first couple of rows and shape, I did a quick search for any duplicate rows and did not find any. There were six columns containing missing or null values after the unnecessary columns were removed. Looking at the status and health columns, it became clear that if a tree was classified as either a stump or dead under status, the health column would be empty. So those rows were removed along with the status column since we are interested in living trees. The remaining rows with missing values could all be removed since they only accounted for ~1-49 rows, which is minimal considering the size of the dataset.

Selecting columns to keep and remove

The goal here is to only keep columns that can help determine a tree's health and remove extraneous ones. In the initial search, the columns block_id, created_at, user_type, address, postcode, zip_city, community board, borocode (the boroughs column already states the borough), cncldist, st_assem, st_senate, nta, nta_name, boro_ct, state (all trees are located in NY), x_sp, y_sp, council district, census tract, bin, and bbl were dropped. We excluded block_id, address, postcode, zip_city, x_sp, y_sp, and postcode from the dataset since the latitude and longitude columns were sufficient for determining the tree's location. Any political and census data is irrelevant so we removed cncldist, community_board, st_assem, st_senate, nta, nta_name, boro_ct, council district, and census tract. We removed created_at and user_type because we do not need to know about the time of entry or who created the entry.

Removing outliers

This section focuses on the tree diameter (tree_bdh) variable since most columns are categorical. The range goes from 0 to 425 inches, with both ends being highly unrealistic for a tree grown in a compact city. I removed all trees with 0 inches listed because 0 inches since those trees could be saplings (babies, not grown trees) or the value was accidentally left blank. Just 67 trees have diameters over 100 inches, and 294 trees have diameters between 50 and 100 inches, making them a very small minority

compared to the rest of the trees. Since the dataset is robust, I removed those two groups since the model being made needs to represent the majority of trees and these values are considered outliers given the rest of the diameters. The average diameter of the trees after cleaning was 11.67 inches with a standard deviation of 8.37 inches.

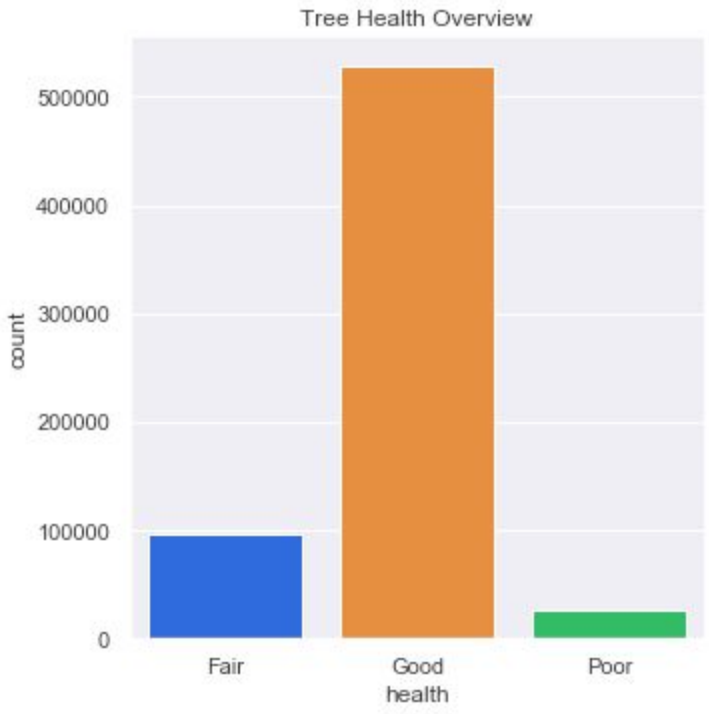
Cleaned data overview

The final dataset consists of 651,535 rows and 21 columns. Our target variable is the health column since it states the tree's condition, classifying it as good, fair, or poor. Below is a chart of remaining variables and their data types.

#	Column	Non-Null Count	Dtype
0	tree_id	651535 non-null	int64
1	tree_dbh	651535 non-null	int64
2	curb_loc	651535 non-null	object
3	health	651535 non-null	object
4	spc_common	651535 non-null	object
5	steward	651535 non-null	object
6	guards	651535 non-null	object
7	sidewalk	651535 non-null	object
8	problems	651535 non-null	object
9	root_stone	651535 non-null	object
10	root_grate	651535 non-null	object
11	root_other	651535 non-null	object
12	trunk_wire	651535 non-null	object
13	trnk_light	651535 non-null	object
14	trnk_other	651535 non-null	object
15	brch_light	651535 non-null	object
16	brch_shoe	651535 non-null	object
17	brch_other	651535 non-null	object
18	borough	651535 non-null	object
19	longitude	651535 non-null	float64
20	latitude	651535 non-null	float64

Data Storytelling and Visualization

Once the data is clean and ready to use, we use exploratory data analysis to take a closer look at the data. Numerical and categorical data are handled separately. Our goal is to understand as much as possible about the data by creating visuals in the form of graphs and charts that explain the relationship between the target and independent variables. This is also a great time to take note of any abnormalities and skewness in the data. Now is a good time to reaffirm our target variable as the health variable since we are looking to build a model around predicting the health of a tree.



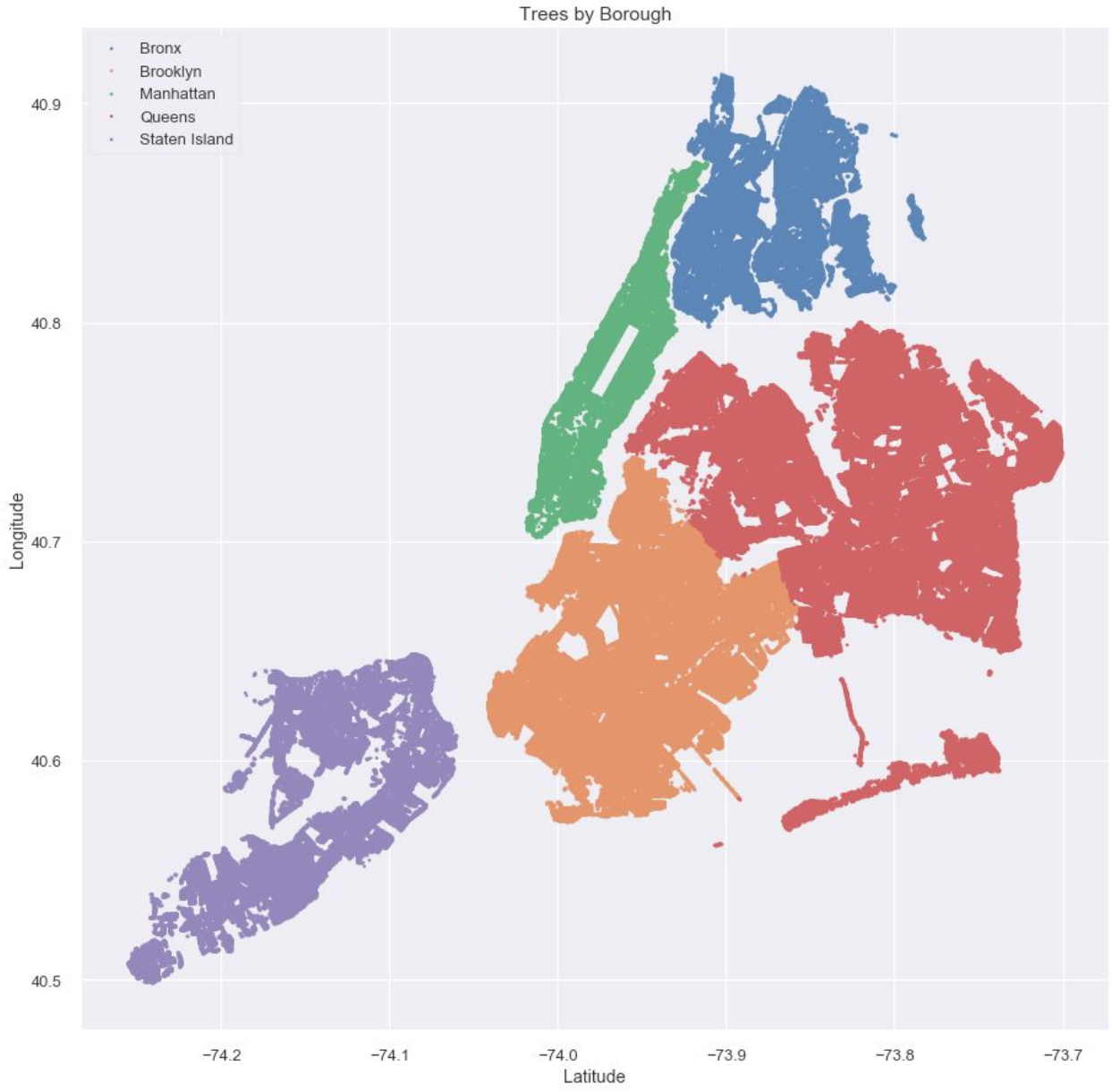
The majority of trees are in good health. Fair and poor trees are minorities. When we're building the machine learning model, special techniques will be needed to balance this imbalanced data.

Boroughs

borough	health	
Bronx	Good	82.65
	Fair	13.51
	Poor	3.84
Brooklyn	Good	81.42
	Fair	14.77
	Poor	3.80
Manhattan	Good	75.86
	Fair	18.36
	Poor	5.78
Queens	Good	81.53
	Fair	14.52
	Poor	3.95
Staten Island	Good	81.49
	Fair	14.34
	Poor	4.17

Queens has the highest number of trees, followed by Brooklyn, Staten Island, the Bronx, and Manhattan. It should be noted that Central Park is in Manhattan and has some 20,000 trees living in there, which are not included in the census. Wall Street is in the lower financial district and there aren't many trees in an area so busy. Suburban areas like Queens and Brooklyn will have more trees on city streets.

Aerial Distribution



Stewardships

The Parks and Rec website defines stewardship as the number of unique signs of stewardship observed for a tree. A tree steward is a volunteer who regularly cares for a tree by watering it and removing weeds.

steward	health	
1or2	Good	80.36
	Fair	15.26
	Poor	4.38
3or4	Good	81.35
	Fair	14.76
	Poor	3.90
4orMore	Good	84.52
	Fair	12.43
	Poor	3.05
None	Good	81.28
	Fair	14.67
	Poor	4.04

Most trees do not have any stewards, or caretakers, while around 20% have 1 or 2. A minority have 3 or 4 and less than 0.5% have 4 or more stewards. This means most trees are left to fend for themselves. Understandably, trees with 4 or more stewards are likelier to be in good health and less likely to be in poor health. Interestingly, it's the trees with 1 or 2 stewards that have the lowest percentages of good trees, and the highest percentage of poor and fair trees! It looks like trees are fine being on their own.

Guards

A guard is described as a perimeter set up around a tree that protects from the urban environment. Census takers rate the guard as helpful, harmful, unsure, or none if there is no guard present.

guards	health	
Harmful	Good	75.66
	Fair	18.95
	Poor	5.39
Helpful	Good	82.21
	Fair	13.81
	Poor	3.98
None	Good	81.25
	Fair	14.70
	Poor	4.05
Unsure	Good	76.30
	Fair	17.76
	Poor	5.94

Most trees do not have a guard protecting it from the city. Of those that do, the ones with a harmful guard are more likely to have poor trees and less likely to have good trees. Only trees with guards that are unsure have the highest percentages of poor trees, though it's only slightly more than harmful trees.

Sidewalks

Sidewalks describe the condition of the sidewalk that is closest to the tree.

sidewalk	health	
Damage	Good	81.14
	Fair	15.33
	Poor	3.53
NoDamage	Good	81.07
	Fair	14.59
	Poor	4.35

It looks like sidewalk conditions have a marginal impact on the tree since the distributions of trees in all classes are nearly identical.

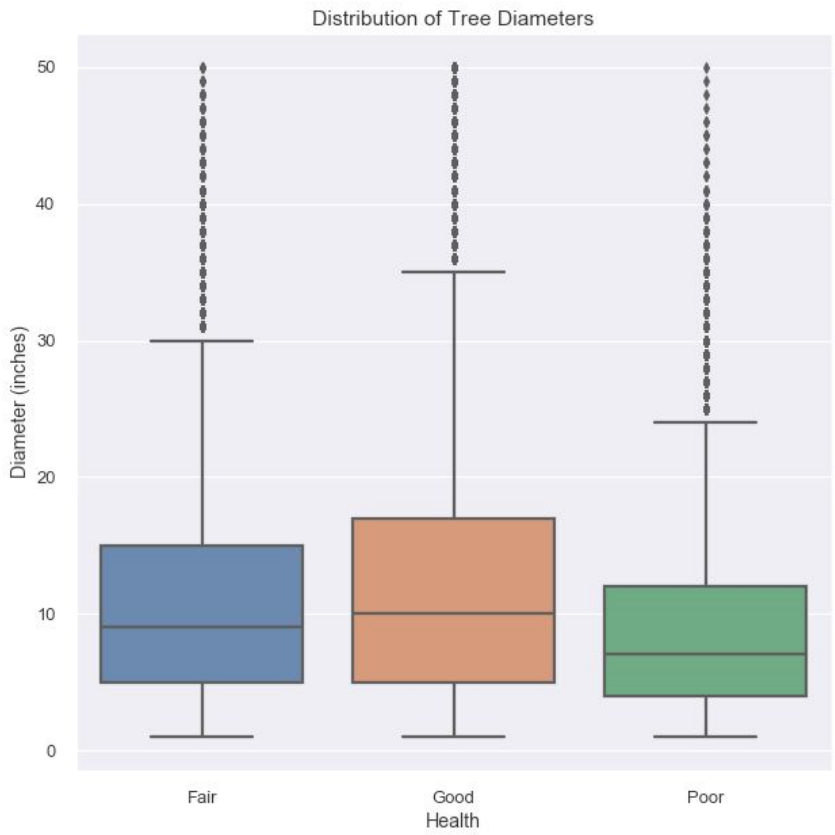
Curb Location

This is the location of the tree bed in relation to the curb; trees are either along the curb (OnCurb) or offset from the curb (OffsetFromCurb).

curb_loc	health	
OffsetFromCurb	Good	80.68
	Fair	15.59
	Poor	3.73
OnCurb	Good	81.11
	Fair	14.77
	Poor	4.13

Once again, the impact is minimal since the distributions are nearly identical.

Tree Diameters



On average, healthier trees have larger diameters than poor trees. Good trees have the highest average diameters and more trees in the upper percentiles. Poor trees are smaller in diameter on average compared to both fair and good trees. They also have a smaller distribution of diameters.

Problems

There is a column, problems, that lists the problems observed on a tree, which is identical to nine columns that state possible tree problems. For example, if there was a problem with wires on a tree, the problems column would list wires rope and the trunk wire column would indicate yes. For this reason, I counted the number of problems for each tree and placed the number in a newly created column, num_problems. I removed the problems columns since there are other columns that indicate the same thing.

Statistical Analysis

This part consists of using statistical techniques to establish independence between the target variable and remaining variables. Due to the large amounts of categorical data, label encoding is needed for converting strings into integers.

Label encoding

The data is mostly categorical, which means strings instead of integers, with several binary columns mixed in. For the binaries, I converted the strings into 0 or 1 based on No or Yes, respectively using a label encoder. The remaining columns contain nominal data, meaning there is no intrinsic order to the responses. For example, in the guards column, the responses are None, Harmful, Helpful, Unsure. There is no relationship or order between these variables.

Chi-square test for association

For numerical variables, I used the chi-squared test for association in order to determine if there is a statistically significant relationship between health and the variable in question. Each null hypothesis states that the relationship between variables is independent. The p-value threshold is 0.05. Every variable had a p-value far below the threshold, meaning there is significance between that variable and the target variable.

ANOVA – analysis of variance test

For the two numerical columns, tree_dbh and num_problems, I divided the data in each column based on the corresponding tree health. Since there are three categories, we used the analysis of variance, or ANOVA, test in order to compare the means of each group against each other. This is an extension of the t-test, which is used to determine statistical significance between the means of two variables. Both variables were deemed significant to the target variable.

Correlation between variables

In order to determine if the variables are independent of each other, I created a heat map with correlation coefficients. Two sets of columns had coefficients less than the threshold of -0.7: None and 1 or 2 and None (different column) and Helpful. In both cases, I dropped the None column.

Machine Learning

The final step is to build a model that can correctly predict the health of a tree. Since this is a multivariate classification problem, we cannot solely depend on the accuracy score. We also must look at the classification report, which has the precision, recall, and f1-score. Those are our benchmarks for success. Previously, we learned that the number of good trees vastly outnumber fair and poor trees, making this a highly imbalanced dataset. Therefore, I am implementing under and over sampling techniques in an effort to increase the accuracy, precision, recall, and f1-scores.

Which algorithms did you choose and why?

For this classification problem, I chose logistic regression, k-nearest neighbors classifier, decision tree classifier, random forest classifier, and Gaussian Naive Bayes as the algorithms to use in building a model that can correctly classify a tree into one of three groups: good, fair, or poor. These algorithms are best suited for handling classification problems.

Which under sampling methods did you choose and why?

Under sampling methods balance the number of majority and minor samples by removing majority class samples until the numbers from both classes are equal. The methods used are random under sampler, Tomek links, edited nearest neighbours, and near miss. Random Under Sampler is exactly what it says: it randomly selects majority samples to remove. Tomek Links remove majority class samples by looking at where classes overlap. The end result is that all nearest neighbors are of the same class. Edited Nearest Neighbours uses the edited nearest neighbor method to remove samples so that samples of the same class are closest to each other. Near Miss reduces majority samples by looking at points where majority and minority samples are near and removes the majority sample.

Which over sampling methods did you choose and why?

Over sampling methods are the opposites to under sampling, meaning that they increase the minority class sample size to match the majority class by duplicating minority samples. The methods used are random over sampler, synthetic minority over-sampling technique (SMOTE), and adaptive synthetic (ADASYN). Random Over Sampler is a simple method that randomly duplicates minority samples till they match majority samples in size. Synthetic minority over sampling technique (SMOTE) randomly picks minority points, locates their closest neighbors (five is the default),

draws a line from the point a to b, and creates a new sample in the middle. Adaptive synthetic (ADASYN) is similar to SMOTE but takes it one step further by adding random values to the points to make them seem more realistic.

Which combination methods did you choose and why?

Combination under and over sampling methods selectively under sample majority data while oversampling minority data. We use two combination methods, SMOTE-Tomek and SMOTE-ENN. SMOTE-Tomek over samples using SMOTE technique while cleaning with Tomek links. SMOTE-ENN uses SMOTE oversampling and cleans using edited nearest neighbors.

What are the best fit models?

The best models for this dataset use the under sampling technique, edited nearest neighbours. Decision tree classifier and random forest classifier models produce similar results with high accuracy, precision, and f1- scores. Since precision and recall have an inverse relationship, we are choosing to focus more on the former this time. Though the recall scores for good and fair trees are also very high.
One of the downsides of using decision trees and random forests is that the models tend to become overfitted during the training session, as is evident in this case where the accuracy scores for the training set is slightly higher than the test set. Since the difference is ~1%, the data is not very overfitted.

```
Decision Tree Classifier

Accuracy Score, Training Set: 0.965269240707535
Accuracy Score, Test Set: 0.9565896571569514
Confusion Matrix:
 [[  334    23    36]
 [   19 77849   269]
 [  112  3237  3262]]

Classification Report

              precision    recall  f1-score   support

        Fair       0.72      0.85      0.78       393
        Good       0.96      1.00      0.98     78137
        Poor       0.91      0.49      0.64      6611

    accuracy                           0.96     85141
   macro avg       0.86      0.78      0.80     85141
weighted avg       0.96      0.96      0.95     85141
```

```
Random Forest Classifier

Accuracy Score, Training Set: 0.965269240707535
Accuracy Score, Test Set: 0.957799415087913
Confusion Matrix:
 [[  330    23    40]
 [    6 77980   151]
 [   96  3277  3238]]

Classification Report

              precision    recall  f1-score   support

        Fair       0.76      0.84      0.80       393
        Good       0.96      1.00      0.98     78137
        Poor       0.94      0.49      0.65      6611

    accuracy                           0.96     85141
   macro avg       0.89      0.78      0.81     85141
weighted avg       0.96      0.96      0.95     85141
```

Future Plans

The next step is to look into tree census data from over the years and see if the model created for 2015 can be applied to other years as well. It would be interesting to see which predictions made one year are accurate as time goes on. Another interesting topic is taking into consideration the species of tree and figuring out how that can influence health over time.