

Predicting Blight in Detroit

Stuart Barnum

5/20/2018

My aim in this project is to gain insight into patterns of urban blight, using open data sets published by the city of Detroit. Blight is understood in terms of issues related to buildings: citations for blight-related offenses such as failure to maintain a building or its grounds, blight-related complaints to a city-run hotline, local crime rates, and indications that the building is or was likely to be demolished. A building is deemed to have become irredeemably blighted if city records indicate that the building was either demolished or likely to be demolished. Using this notion, I constructed a variety of models for predicting which buildings will become irredeemably blighted, each based on certain features constructed from the data; for example, the number of blight-related offenses associated with a given building. Although as many as 13 features appear to be at least somewhat effective as predictors—removing any one them appears to make the most effective models at least somewhat less accurate—two of the features stand out as most effective: the number of buildings (other than the building for which the prediction is taking place) within 100 meters that became irredeemably blighted before the time over which the models were trained, and the current number of vacant lots within 200 meters. Using a decision tree to predict on the basis these two features alone consistently produced models with predictive accuracies of 70%. Two of the more sophisticated approaches, random forest and adaboost, made use of all 13 of the features and yielded predictive accuracies of roughly 73%.

Complete development of the dataset used for modeling, including code and exploratory graphics, can be found at https://stuartbarnum.github.io/Detroit-Demolitions/Detroit_Draft_3.html.

Complete development of the models used for predicting can be found at https://stuartbarnum.github.io/Detroit-Demolitions/Detroit_models_from_tallies.html.

General methods

After a significant amount of data cleaning (as described in the next section), one of the first tasks in this project was to define a list of buildings. This could be done in one of at least two ways. In the first, suggested as a possible method in the guidelines for this project, buildings might be identified with clusters of incidents such as, perhaps, recorded crime incidents and citations for blight. I elected not to use this method for a number of reasons. The first reason is the possibility that there may be clusters of incidents associated with non-buildings such as vacant lots and parks, and the possibility that there may be buildings at which there were very few or no incidents. As it would be impossible to identify all or most of the cases of either of these types, such cases may distort the analysis. Another reason was the availability of a relatively clean set of parcel (property lot) data, which included both geographical information about the parcels and information about any buildings on the parcels. Buildings can thus be represented in terms of a subset of these parcels. One of the drawbacks of this latter approach is that some of the parcels contain, or have contained, more than one building. And although it would be possible to eliminate from the data the parcels that, according to the data, *currently* contain more than one building, it is not possible to discern from the available data the parcels that have, in the past, contained more than one building (for the cases in which, for example, all of the buildings within a parcel have been demolished). Although either of the two methods may provide valuable insight, I elected to use the parcel data. The analysis was done on parcels that have, since May of 2016, contained at least one building. For example, although the data may indicate that no building currently

exists on a given parcel, the data may indicate that a building on that parcel was dismantled in 2017, thus indicating that a building existed on that parcel.

The next step in the analysis was to assign a set of labels, irredeemably blighted or not irredeemably blighted (henceforth simply “blighted” or “not blighted”), to the buildings. A building is deemed to be or have been blighted if it (1) was demolished, as indicated in an online list of buildings that have been demolished under the Detroit Demolitions Program (see <https://data.detroitmi.gov/Property-Parcels/Detroit-Demolitions/rv44-e9di>), (2) is contained in a list of upcoming demolitions under this program (see <https://data.detroitmi.gov/Property-Parcels/Upcoming-Demolitions/tsqq-qtet>) or (3) has a demolition permit associated with it (see <https://data.detroitmi.gov/Property-Parcels/Building-Permits/xw2a-a7tf>, in which the demolition permits are “building permits” for which the specified type is “dismantle”). I found it useful to use all of these datasets because there were reasons to believe that any one of them would fail to list significant numbers of blighted parcels. For example, the documentation associated with the completed demolitions dataset indicated that the dataset fails to include some demolitions that were completed on an emergency basis. On the other hand, there are a significant number of buildings listed in the completed demolitions dataset that, based on the demolitions permits data, appear to not have had demolition permits associated with them, thus suggesting that the demolitions permits data is also incomplete.

I should note that one possible fault in my approach to the labeling is with the fact that we used the demolitions permits data within our operational construal of blight. After all, for example, a wealthy resident might purchase an impeccably maintained home and then demolish it (with the requisite permit) to build a larger home. Likewise in the case of other buildings demolished in order to make room for other buildings. With this issue in mind, I restricted my analysis to those areas within Detroit that have been identified as Hardest Hit Areas (see http://d3-d3.opendata.arcgis.com/datasets/383eb730952e470389f09617b5448026_0), for which federal funding is available for the demolition program. The assumption, here, is that, in such areas, a smaller proportion of the buildings that were torn down were sound (not genuinely blighted) buildings that were, again, torn down simply to make room for other buildings (or parking lots, etc.).

Another matter of fundamental methods concerns the manner of association of the various potential predictors, such as numbers of blight-related citations associated with a particular building, with the relevant buildings. To make the associations, I used both spatial relationships and, where both necessary and possible, parcel numbers. For example, each record of a blight-related citation includes both a parcel number (identifier of the parcel) and a pair of latitude and longitude coordinates. If the position indicated by the coordinates was within a certain parcel, then that parcel was assumed to be the parcel associated with the citation. Otherwise, if an association could be made by means of identity of parcel numbers (the parcel number for the building and the parcel number recorded with the citation), then the parcel thus associated was deemed to be the parcel associated with the citation. Other associations were made merely by means of geometric relationships (with no consideration of parcel numbers). For example, recorded crime incidents, divided between violent crimes, property crimes, and nuisance crimes, were associated with parcels in virtue of spatial proximity of 200 meters.

Data munging and cleaning

The project was implemented in R and, at the data munging and cleaning stages, made extensive use of the `tidyverse` packages for data manipulation, `ggmap` for investigative visual maps and for geocoding in cases of missing position coordinates, and the `sf` package for handling spatial information and relationships. The R `sf` package provides an implementation of the simple features standard, ISO 19125-1:2004, for representing real-world objects in computers. Data frames in which spatial information is thus represented become simple features data frames, and it was possible to read the shapefile-format parcels dataset from the city of Detroit (see <https://data.detroitmi.gov/Property-Parcels/Parcel-Map/fxkw-udwf/data>) directly into a simple features dataframe. Likewise in the case of datasets I used for the hardest hit areas and for the council districts—these relatively small files, in the form of simple maps of the geometries of the respective areas of the city, were read directly into the simple features format. The other datasets, containing geographic point information, were converted into simple features data frames after I had both eliminated some geographical coordinate information that was clearly incorrect (e.g. blight citations for which the coordinate information indicated a

position well outside of Detroit) and, where possible, filled in the missing coordinate information by means of the `ggmap` `geocode` function, which accesses the Google API for geocoding. Among the data in all of the datasets I used in this project, a total of roughly 5,000 locations were geocoded. Data for which it was impossible to obtain usable location information was discarded. The conversion into an `sf` data frame also required some string manipulation, of the raw coordinate information. The coordinate reference system (providing the mapping of coordinates to locations) for all of the `sf` data frames was 4326, which is standardly used in GPS systems. (See <https://www.nceas.ucsb.edu/~frazier/RSpatialGuides/OverviewCoordinateReferenceSystems.pdf> for a brief overview of coordinate reference systems.)

Another aspect of the data cleaning involved the parcels dataset. Two notable issues in this data were (1) identity of the parcel numbers among pairs of rows for which the parcel geometries were disjoint (non-overlapping) and (2) identity of the space covered by certain pairs of rows for which the parcel numbers were distinct. Although there were less than 100 pairs of either of these types, I addressed these issues as follows. For pairs of the former type (1), I made the parcel numbers (represented as strings in the data) distinct by appending unique identifiers at the end. For pairs of the latter type (2), I eliminated one of elements of each pair from the data. All of these cases (both (1) and (2)) were identified by means of the `sf` function `st_join`, by which one may implement SQL-like joins on the basis of spatial relationships. This required that the spherical representation of the latitude and longitude coordinates be projected into a plane, which, given the relatively small size of the city of Detroit in relation to Earth, provides a roughly accurate representation.

Another aspect of the parcels data that I investigated carefully was several thousand pairs of parcels that, according to an application of `st_join`, overlapped. I plotted a random sample of 100 of these pairs of parcels and found no evidence of overlap in the plots. I concluded that the apparent overlap may have been due to the projection (perhaps imperfect projection) of the spherical representation (coordinate representation system 4326 as per above) into a flat representation, and was likely not a problematic issue in my data.

After the various investigative and cleaning steps such as the above were completed, I decided to restrict my analysis to those parcels in the Hardest Hit Areas, and so I removed from the analysis all of the parcels that were not within one of these areas (using, again, `st_join`). I then constructed a set of labels for the remaining parcels, consisting of, for each parcel, the parcel number and the correct value with respect to blight (blighted or not blighted). I then cut out most of the rows in the labels dataset for which the value with respect to blight indicated *not blighted*, so as to create a relative balance between positive instances (blighted) and negative instances (not blighted). The parcels thus removed from the data were selected at random.

Construction of a tallies dataset

The predictive models that were eventually constructed in this project used data in the form of tallies (and other sums) from the data that had been processed as described above. All of these values were calculated using SQL-style joins on the basis of spatial relationships (`st_join`, again) or, in a few cases, identity of parcel numbers (using the `dplyr` function `inner_join`). After the joining, the results were then grouped and counted (using the `dplyr` package) as one does in SQL. In the applications of `st_join`, there were two types of tallies. In the first, I simply looked for incidents, such as citations for failure to maintain a property, occurring within a parcel. The other type of application of `st_join` involved looking for incidents that were within a certain distance from a parcel. In the latter type of case, I created a “buffer” around each parcel using the `sf` operation `st_buffer`, thus expanding each of the parcels by a certain distance, depending on what I was attempting to count. With the parcels thus expanded, I applied `st_join`, now looking for incidents (and some other entities such as vacant lots) within the expanded parcels. At the end of all of this, the tallies dataset consisted of the parcel numbers for parcels in the balanced set of labels as described above, the value corresponding to blighted or not blighted, and 14 variables providing such tallies. The variables were, roughly, defined as follows.

Variable Name	Variable Description
later_recorded_crime_incidents	Total recorded crime incidents after June of 2016
number_of_violations_by_parcel	Number of blight related citations for offences at the parcel
total_fines_by_parcel	Total fines assessed for blight citations at the parcel
improve_issues_tallies	"Improve Detroit Issues" at the parcel related to blight
earlier_property_crime_incidents	Property crime incidents through June of 2016 within 200 meters (m)
earlier_violent_crime_incidents	Violent crime, including robbery, through June of 2016 within 200 m
earlier_nuisance_offences	Certain other crimes, e.g. prostitution, through June of 2016 within 200 m
total_acre	Area of the parcel, in acres
council_di	Council district in which the parcel is located
frontage	Frontage of the parcel, in feet
num_vacant_parcel	Number of vacant parcels within 200 m
num_nearby_blighted_parcel	Buildings within 100 m that became blighted before June of 2016
num_violations_nearby_parcel	Blight violations within 100 m
sum_fines_nearby_parcel	Sum of fines for blight violations at buildings within 100 m

As indicated in the following summary statistics, the dataset from which models were constructed consists of 4283 rows.

```
library(tidyverse)
complete_tally_set <- read_rds("calculated_tallies.rds")
```

```
summary(complete_tally_set)
```

```
##   parcelnum      later_recorded_crime_incidents    blighted
## Length:4283      Min.   : 1.0                      Min.   :0.0000
## Class :character  1st Qu.: 33.0                     1st Qu.:0.0000
## Mode  :character  Median : 48.0                     Median :0.0000
##                      Mean   : 52.3                   Mean   :0.4863
##                      3rd Qu.: 66.0                     3rd Qu.:1.0000
##                      Max.   :263.0                    Max.   :1.0000
##
##   number_of_violations_by_parcel total_fines_by_parcel
## Min.   : 0.0000              Min.   : 0.0
## 1st Qu.: 0.0000              1st Qu.: 0.0
## Median : 0.0000              Median : 0.0
## Mean   : 0.9479              Mean   : 322.9
## 3rd Qu.: 1.0000              3rd Qu.: 200.0
## Max.   :52.0000              Max.   :55925.0
##
##   improve_issues_tallies earlier_property_crime_incidents
## Min.   : 0.00          Min.   : 3.0
## 1st Qu.: 18.00         1st Qu.:126.0
## Median : 27.00         Median :172.0
## Mean   : 38.43         Mean   :178.4
## 3rd Qu.: 41.00         3rd Qu.:222.0
## Max.   :488.00         Max.   :707.0
##
##   earlier_violent_crime_incidents total_acre council_di
## Min.   : 1.0                      Min.   : 0.0000 1:853
## 1st Qu.:136.5                    1st Qu.: 0.0880 2:631
## Median :191.0                    Median : 0.1050 3:418
## Mean   :200.0                    Mean   : 0.1348 4:720
## 3rd Qu.:249.0                    3rd Qu.: 0.1230 5:455
```

```

## Max.      :720.0          Max.      :16.6980    6:528
##                                     7:678
##      frontage      num_vacant_parcel      num_nearby_blighted_parcel
## Min.      : 0.00      Min.      : 0.00      Min.      : 0.000
## 1st Qu.: 35.00      1st Qu.: 19.00      1st Qu.: 0.000
## Median : 40.00      Median : 41.00      Median : 2.000
## Mean     : 41.84      Mean     : 52.39      Mean     : 3.028
## 3rd Qu.: 41.00      3rd Qu.: 75.00      3rd Qu.: 5.000
## Max.     :725.00      Max.     :219.00      Max.     :28.000
##
## num_violations_nearby_parcel sum_fines_nearby_parcel
## Min.      : 0.00          Min.      : 0
## 1st Qu.: 16.00          1st Qu.: 4100
## Median : 27.00          Median : 7600
## Mean     : 32.85          Mean     : 10022
## 3rd Qu.: 43.00          3rd Qu.: 13000
## Max.     :300.00          Max.     :162125
##
## earlier_nuisance_offences
## Min.      : 1.000
## 1st Qu.: 1.000
## Median : 1.000
## Mean     : 2.311
## 3rd Qu.: 3.000
## Max.     :64.000
##

```

Further details about this dataset, including the code that was used in its construction from the raw data, can be found (again) at https://stuartbarnum.github.io/Detroit-Demolitions/Detroit_Draft_3.html. In the end, because our analysis is restricted to dismantle permits and demolitions after May of 2016, the variable `later_recorded_crime_incents` was not used in our models. I should also note that, as I built up the models, I did not treat all of the incidents in each of the datasets the same. For example, I assumed that complaints from, say, neighbors about failure to keep up a building would have a clear association with blight on the property for which the complaint was made, whereas complaints for failure of the city to fill in potholes in a certain area may have a less clear connection with blight. Likewise, again, although it is difficult to make the distinction in a fully satisfactory way, I created separate variables for property crime, violent crime, and nuisance crime. And in fact, in many of my decision-tree models, violent crime had a weak but positive association with blight, whereas property crime had a weak but negative association with blight.

Models

For the modeling, I removed 15% of the data, to be reserved for a final check on our models. I then partitioned the remaining data into 10 subsets, for k-fold cross validation. I constructed a number of models as I built up the tally dataset, beginning with both decision-tree models using `rpart` and logistic regression models using `glm`, on the numbers of blight violations for each parcel and the total amount of fines for blight violations for each parcel. The initial decision-tree models on these two variables achieved an accuracy of roughly 60% percent, while the logistic regression models achieved an accuracy of 58%. Both of these model-types suggested that fine totals for each parcel provided a somewhat better predictor than number of violations for each parcel. Both of the model-types were notably poor predictors for positive instances (buildings that were blighted), as the following reconstruction, which includes confusion matrices for the k-fold cross-validation, indicates. (Note that blight corresponds to truth = 1.)

```
library(rpart)
```

```
#the parcel numbers for the examples that were not reserved for final testing
```

```

training_parcelnums <- read_rds("training_parcelnums.rds")

#separate the examples to be used for the training from the fifteen percent of
#the examples to be withheld for final testing
train <- complete_tally_set %>% filter(parcelnum %in% training_parcelnums)

train$blighted <- as.factor(train$blighted)

#partition the training set into ten subsets, while maintaining a balance between
#examples labeled as blighted and examples not so labeled
set.seed(294)
k_folds <- caret::createFolds(train$blighted)

#train over the training portion of each of the folds
models <- 1:10 %>% map(~ rpart(blighted ~ total_fines_by_parcel +
                             number_of_violations_by_parcel,
                             data = train[-k_folds[[.x]],]))

predictions <- 1:10 %>% map(~ predict(models[[.x]], newdata = train[k_folds[[.x]],],
                                     type = "class"))

accuracies <- 1:10 %>%
  map(~ mean(unlist(predictions[[.x]] == train[k_folds[[.x]],]$blighted)))

#summary statistics over the models
mean(unlist(accuracies))

## [1] 0.5951633
sd(unlist(accuracies))

## [1] 0.02146108

#confusion matrices for each of the 10 models constructed in our k-fold cross validation
for (index in 1:10) {
  print(table(pred = (predictions[[index]]),
              truth = train[k_folds[[index]],]$blighted))
}

##      truth
## pred   0   1
##    0 140  97
##    1  46  81
##      truth
## pred   0   1
##    0 148  97
##    1  38  81
##      truth
## pred   0   1
##    0 150 107
##    1  37  71
##      truth
## pred   0   1
##    0 141 104
##    1  45  74
##      truth

```

```
## pred  0  1
##    0 145 105
##    1  41  73
##      truth
## pred  0  1
##    0 135 114
##    1  51  64
##      truth
## pred  0  1
##    0 147 107
##    1  39  71
##      truth
## pred  0  1
##    0 145 108
##    1  41  70
##      truth
## pred  0  1
##    0 136  94
##    1  50  84
##      truth
## pred  0  1
##    0 139 106
##    1  47  72
```

*#All of the rpart models in the k-fold cross-validation contained only one split--on
#the total amount of fines related to blight on the parcel, as in the following.*

```
models[[3]]
```

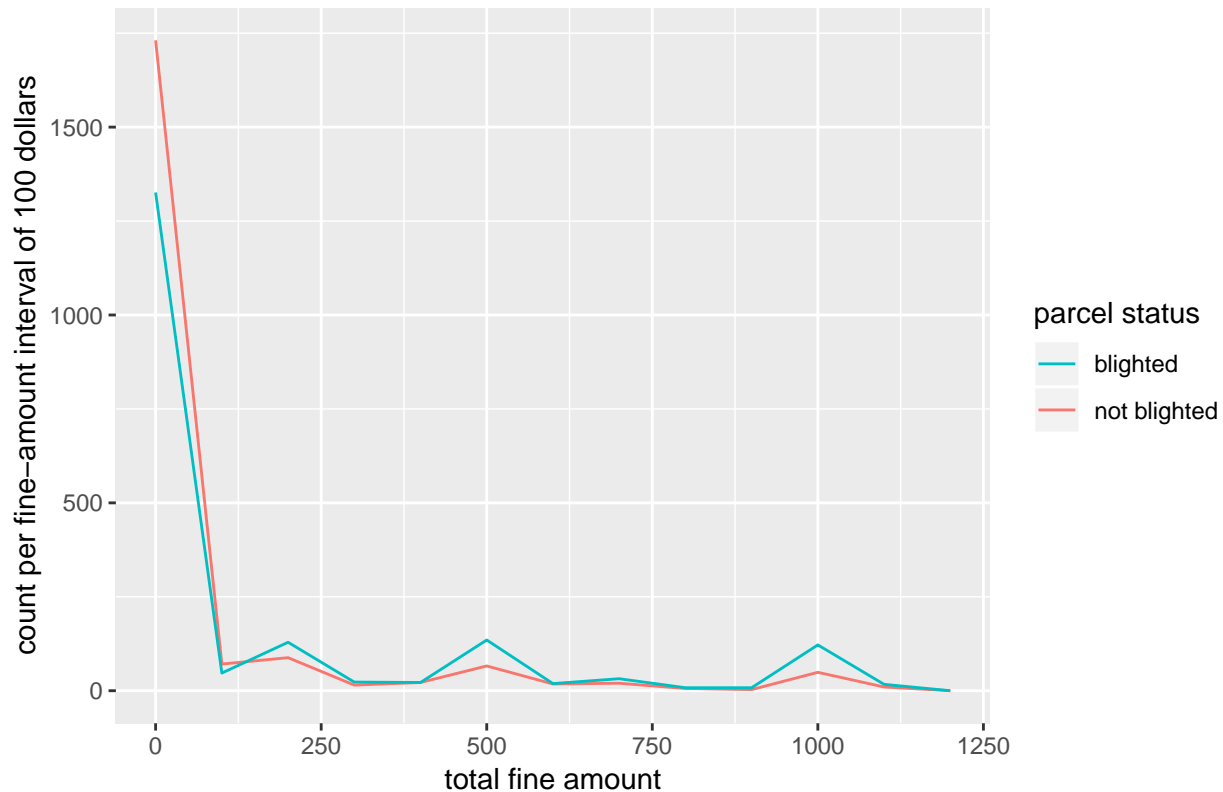
```
## n= 3276
##
## node), split, n, loss, yval, (yprob)
##      * denotes terminal node
##
## 1) root 3276 1602 0 (0.5109890 0.4890110)
##   2) total_fines_by_parcel< 35 2208  932 0 (0.5778986 0.4221014) *
##   3) total_fines_by_parcel>=35 1068  398 1 (0.3726592 0.6273408) *
```

The rather weak but significant association between the irredeemable blight of a building and blight-related citations can also be seen in the following plot.

```
library(tidyverse)

complete_tally_set %>% mutate(blighted = as.factor(blighted)) %>%
  filter(total_fines_by_parcel < 1200) %>%
  ggplot(aes(x = total_fines_by_parcel,
             color = blighted)) +
  geom_freqpoly(binwidth = 100) +
  labs(x = "total fine amount",
       y = "count per fine-amount interval of 100 dollars",
       title = "Distribution across Parcels of Total Fine Amounts") +
  theme(plot.title = element_text(hjust = 0.5)) +
  coord_cartesian(xlim = c(0, 1200)) +
  scale_color_discrete(name = "parcel status",
                      breaks = c("1", "0"),
                      labels = c("blighted", "not blighted")) +
  scale_x_continuous(limits = c(0, 1250))
```

Distribution across Parcels of Total Fine Amounts



As more variables were added to the tally dataset, I also applied random forest (using the `randomForest` package) and, finally, adaboost (using my own implementation in R) and support vector machines (using the `e1071` package). When the tally dataset was complete, I attempted to tune the models by changing the values of some of the parameters, such as the complexity parameter associated with the `rpart` function, which creates decision trees. In the end, most of the improvement in accuracy as I worked on this project was with the introduction of variables concerning blight at nearby parcels, especially in the case of the number of nearby vacant lots (other than the parcel for which prediction is taking place) and the number of nearby parcels, other than the parcel for which prediction is taking place, that became blighted prior to June of 2016. With the complexity parameter set at 0.003 (reduced from the default value of 0.01), the decision-tree models achieved an average accuracy, across the ten models constructed in the k-fold cross validation, of 71%. Similar calculations of average accuracies for random forest and adaboost models yielded 0.73. The accuracy of 0.73 in the adaboost models was achieved with the decision trees (the “weak” classifiers as per the adaboost algorithm) in the form of decision stumps—trees that contained at most one split, and in some cases no splits at all. Adaboost over trees containing more than one split resulted in accuracies at least somewhat less than this. The support vector machine models, after standardization of the numerical predictors, yielded an average accuracy of 0.72.

I should emphasize that the counts of nearby blighted buildings did not include the buildings for which we were predicting, and were for the buildings that became blighted (as per our definition) before the time over which my models attempt to predict. Nevertheless, I tried running some of our approaches to model-building without using the variable for nearby blighted buildings. For the decision-tree models, this resulted in an average accuracy, over the 10 models constructed during k-fold cross validation, of 0.70.

Final thoughts and analysis

Given the relative concreteness and understandability of decision trees, together with the fact that these models were only somewhat less effective as predictors than the more advanced models, I suspect that, for at

least some purposes, decision trees may provide the most useful models. With this in mind, it is especially notable that a remarkably simple model may be useful.

Consider the analysis that provided the most-accurate decision trees.

```
#we use the complete tally set here, and implement k-fold cross-validation over
#this somewhat larger set
complete_tally_set$blighted <- as.factor(complete_tally_set$blighted)

#partition the complete tally set into ten subsets, while maintaining a balance
#between examples labeled as blighted and examples not so labeled
set.seed(451)
k_folds <- caret::createFolds(complete_tally_set$blighted)

complete_formula <- blighted ~ total_fines_by_parcel + number_of_violations_by_parcel +
  improve_issues_tallies + earlier_property_crime_incidents +
  earlier_violent_crime_incidents + total_acre + council_di +
  frontage + num_vacant_parcel + num_nearby_blighted_parcel +
  num_violations_nearby_parcel + earlier_nuisance_offences

#train over the training portion of each of the folds
models <- 1:10 %>% map(~ rpart(complete_formula,
                               data = complete_tally_set[-k_folds[[.x]],],
                               method = "class", control = rpart.control(cp = 0.003)))

predictions <- 1:10 %>% map(~ predict(models[[.x]],
                                       newdata = complete_tally_set[k_folds[[.x]],],
                                       type = "class"))

accuracies <- 1:10 %>%
  map(~ mean(as.numeric(predictions[[.x]] ==
                        complete_tally_set[k_folds[[.x]],]$blighted)))

#summary statistics over the models
mean(as.numeric(accuracies))

## [1] 0.7132922

sd(as.numeric(accuracies))

## [1] 0.01996122

#confusion matrices for each of the 10 models constructed in
#our k-fold cross validation
for (index in 1:10) {
  print(table(pred = predictions[[index]],
              truth = complete_tally_set[k_folds[[index]],]$blighted))
}

##      truth
## pred   0   1
##    0 152  46
##    1  68 162
##      truth
## pred   0   1
##    0 166  60
##    1  54 148
```

```

##      truth
## pred   0   1
##      0 145  48
##      1  75 161
##      truth
## pred   0   1
##      0 155  64
##      1  65 144
##      truth
## pred   0   1
##      0 160  78
##      1  60 131
##      truth
## pred   0   1
##      0 147  61
##      1  73 147
##      truth
## pred   0   1
##      0 159  60
##      1  61 149
##      truth
## pred   0   1
##      0 159  56
##      1  61 152
##      truth
## pred   0   1
##      0 151  45
##      1  69 163
##      truth
## pred   0   1
##      0 162  66
##      1  58 142

```

```
models[[5]]
```

```

## n= 3854
##
## node), split, n, loss, yval, (yprob)
##      * denotes terminal node
##
## 1) root 3854 1874 0 (0.5137519 0.4862481)
##    2) num_nearby_blighted_parcel< 1.5 1807  509 0 (0.7183177 0.2816823)
##      4) num_vacant_parcel< 50.5 1512  344 0 (0.7724868 0.2275132)
##        8) total_fines_by_parcel< 25 1045  156 0 (0.8507177 0.1492823) *
##        9) total_fines_by_parcel>=25 467  188 0 (0.5974304 0.4025696)
##          18) total_fines_by_parcel< 512.5 238   73 0 (0.6932773 0.3067227) *
##          19) total_fines_by_parcel>=512.5 229  114 1 (0.4978166 0.5021834)
##            38) num_violations_nearby_parcel>=58 66   23 0 (0.6515152 0.3484848) *
##            39) num_violations_nearby_parcel< 58 163   71 1 (0.4355828 0.5644172)
##              78) earlier_violent_crime_incidents< 149 48   19 0 (0.6041667 0.3958333)
##                156) council_di=1,2,5,7 31    7 0 (0.7741935 0.2258065) *
##                157) council_di=3,4,6 17    5 1 (0.2941176 0.7058824) *
##              79) earlier_violent_crime_incidents>=149 115  42 1 (0.3652174 0.6347826)
##                158) frontage< 35.5 32   13 0 (0.5937500 0.4062500) *
##                159) frontage>=35.5 83   23 1 (0.2771084 0.7228916) *

```

```
##      5) num_vacant_parcel>=50.5 295 130 1 (0.4406780 0.5593220)
##      10) earlier_property_crime_incidents>=107.5 182 82 0 (0.5494505 0.4505495)
##      20) total_fines_by_parcel< 75 115 42 0 (0.6347826 0.3652174) *
##      21) total_fines_by_parcel>=75 67 27 1 (0.4029851 0.5970149)
##      42) earlier_violent_crime_incidents< 137 11 2 0 (0.8181818 0.1818182) *
##      43) earlier_violent_crime_incidents>=137 56 18 1 (0.3214286 0.6785714) *
##      11) earlier_property_crime_incidents< 107.5 113 30 1 (0.2654867 0.7345133) *
##      3) num_nearby_blighted_parcel>=1.5 2047 682 1 (0.3331705 0.6668295)
##      6) num_nearby_blighted_parcel< 4.5 1045 445 1 (0.4258373 0.5741627)
##      12) total_fines_by_parcel< 25 677 330 1 (0.4874446 0.5125554)
##      24) num_vacant_parcel< 65.5 405 168 0 (0.5851852 0.4148148)
##      48) council_di=5,6 70 17 0 (0.7571429 0.2428571) *
##      49) council_di=1,2,3,4,7 335 151 0 (0.5492537 0.4507463)
##      98) improve_issues_tallies>=79 32 7 0 (0.7812500 0.2187500) *
##      99) improve_issues_tallies< 79 303 144 0 (0.5247525 0.4752475)
##      198) num_nearby_blighted_parcel< 3.5 220 92 0 (0.5818182 0.4181818) *
##      199) num_nearby_blighted_parcel>=3.5 83 31 1 (0.3734940 0.6265060)
##      398) improve_issues_tallies< 23.5 25 9 0 (0.6400000 0.3600000) *
##      399) improve_issues_tallies>=23.5 58 15 1 (0.2586207 0.7413793) *
##      25) num_vacant_parcel>=65.5 272 93 1 (0.3419118 0.6580882)
##      50) num_violations_nearby_parcel>=44.5 25 8 0 (0.6800000 0.3200000) *
##      51) num_violations_nearby_parcel< 44.5 247 76 1 (0.3076923 0.6923077) *
##      13) total_fines_by_parcel>=25 368 115 1 (0.3125000 0.6875000)
##      26) frontage>=91.5 10 2 0 (0.8000000 0.2000000) *
##      27) frontage< 91.5 358 107 1 (0.2988827 0.7011173) *
##      7) num_nearby_blighted_parcel>=4.5 1002 237 1 (0.2365269 0.7634731) *
```

I should note that, although I have only displayed one of the models that were generated in this implementation of k-fold cross validation, the other models are similarly complex. Furthermore, experimentation with the complexity parameter in the `rpart` function suggests that, despite the relative complexity of these models, they may not be overfit—adjusting the complexity parameter even somewhat lower produces a simpler model that is somewhat less accurate (yields a somewhat lesser average accuracy under k-fold cross validation). (On the other hand, adjusting the complexity parameter lower results in a more complex model that predicts less well: thus perhaps a model that *is* overfit). However, if we adjust the complexity parameter to the `rpart` default value of 0.01, we obtain a strikingly simple model that is only somewhat less accurate than the rather complex models constructed in the above.

```
models <- 1:10 %>% map(~ rpart(complete_formula,
                              data = complete_tally_set[-k_folds[[.x]],],
                              method = "class", control = rpart.control(cp = 0.01)))

predictions <- 1:10 %>% map(~ predict(models[[.x]],
                                      newdata = complete_tally_set[k_folds[[.x]],],
                                      type = "class"))

accuracies <- 1:10 %>%
  map(~ mean(as.numeric(predictions[[.x]] ==
                        complete_tally_set[k_folds[[.x]],]$blighted)))

#summary statistics over the models
mean(as.numeric(accuracies))

## [1] 0.6999749

sd(as.numeric(accuracies))
```

```
## [1] 0.02784021
#confusion matrices for each of the 10 models constructed in
#our k-fold cross validation
for (index in 1:10) {
  print(table(pred = predictions[[index]],
              truth = complete_tally_set[k_folds[[index]],]$blighted))
}
```

```
##      truth
## pred  0   1
##      0 123 37
##      1  97 171
##      truth
## pred  0   1
##      0 146 38
##      1  74 170
##      truth
## pred  0   1
##      0 126 33
##      1  94 176
##      truth
## pred  0   1
##      0 147 60
##      1  73 148
##      truth
## pred  0   1
##      0 160 74
##      1  60 135
##      truth
## pred  0   1
##      0 118 49
##      1 102 159
##      truth
## pred  0   1
##      0 131 33
##      1  89 176
##      truth
## pred  0   1
##      0 160 60
##      1  60 148
##      truth
## pred  0   1
##      0 160 54
##      1  60 154
##      truth
## pred  0   1
##      0 127 45
##      1  93 163
```

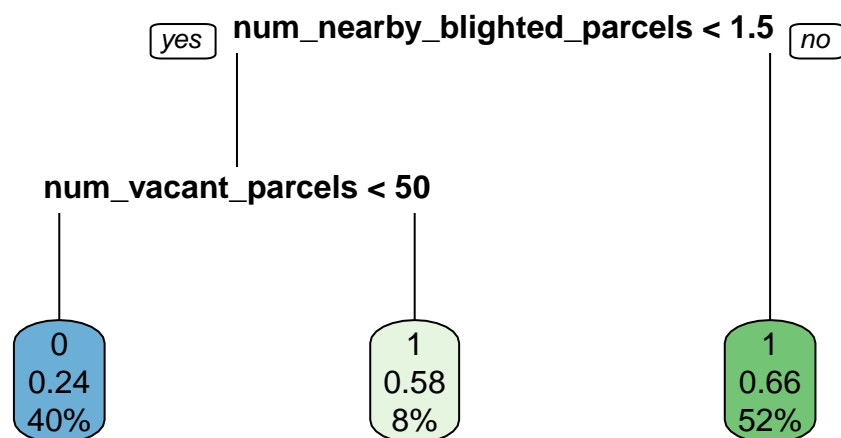
```
models[[3]]
```

```
## n= 3854
##
## node), split, n, loss, yval, (yprob)
##      * denotes terminal node
```

```
##
## 1) root 3854 1874 0 (0.5137519 0.4862481)
## 2) num_nearby_blighted_parcel< 1.5 1833 531 0 (0.7103110 0.2896890)
## 4) num_vacant_parcel< 50.5 1540 362 0 (0.7649351 0.2350649) *
## 5) num_vacant_parcel>=50.5 293 124 1 (0.4232082 0.5767918) *
## 3) num_nearby_blighted_parcel>=1.5 2021 678 1 (0.3354775 0.6645225) *
```

I should note that all of the models generated in the above k-fold cross-validation process have the same form as does the above decision-tree model (with somewhat different break points, such as with 52.5 instead of 50.5 at nodes 4 and 5). Plotted in a proper decision-tree format (with, again, “1” standing for *blighted* and “0” standing for “not blighted”), the model appears as follows. (Recall that `num_vacant_parcel` pertains to the number of vacant parcels within 200 meters, while `num_nearby_blighted_parcel` pertains to the number of buildings within 100 meters that, according to my data and definitions, became blighted before June of 2016. Also, recall that both of these quantities exclude the building being for which we are predicting.)

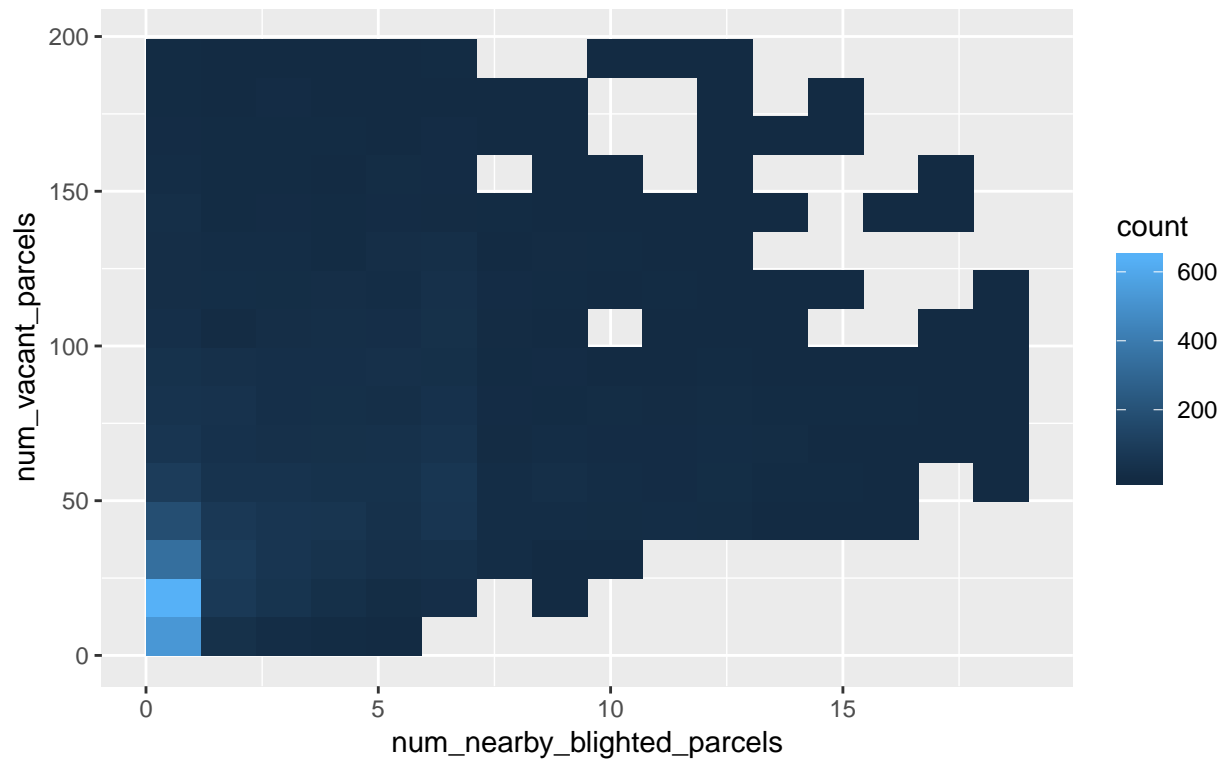
```
library(rpart.plot)
rpart.plot(models[[3]], type = 0)
```



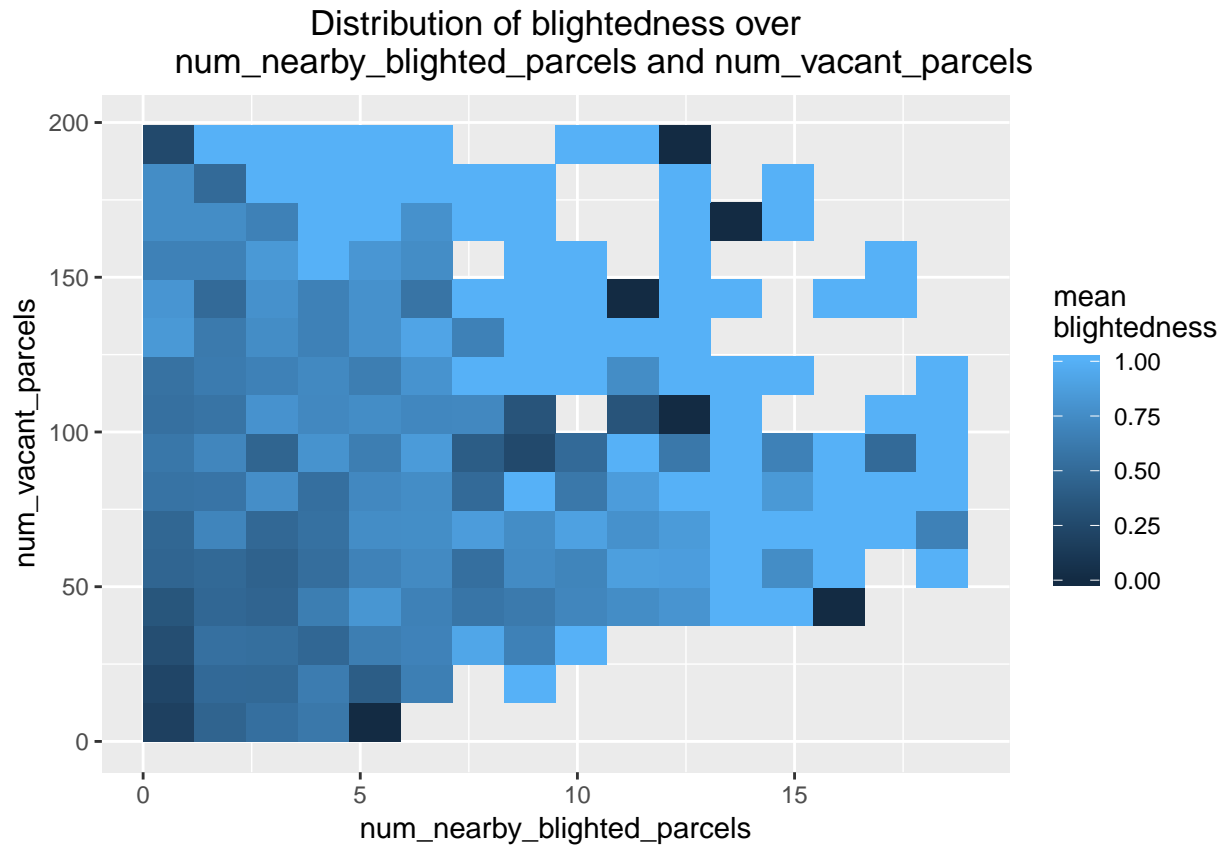
As is well-known about the city of Detroit, many of the vacant parcels once contained buildings, which were torn down for blight-related reasons. Given the suggestion that the number of nearby blighted parcels and the number of nearby vacant parcels are especially effective as predictors, we see that past and more recent nearby blight, in our strong sense of the probable destruction of a building, may provide the best predictors of the blightedness of any given building. It may thus be interesting, finally, to consider a plot of the joint distribution of these two predictors across the parcels in our balanced dataset, and then a plot of average blightedness as a function of the two predictors.

```
complete_tally_set %>%
  filter(num_nearby_blighted_parcel < 20 & num_vacant_parcel < 200) %>%
  ggplot() + stat_bin_2d(aes(x = num_nearby_blighted_parcel,
                             y = num_vacant_parcel),
                        bins = 16) +
  ggtitle("Distribution of num_nearby_blighted_parcel and num_vacant_parcel
           (within the balanced dataset)") +
  theme(plot.title = element_text(hjust = 0.5))
```

Distribution of num_nearby_blighted_parcels and num_vacant_parcels
(within the balanced dataset)



```
complete_tally_set %>% mutate(blighted = as.numeric(blighted) - 1) %>%
  filter(num_nearby_blighted_parcels < 20 & num_vacant_parcels < 200) %>%
  ggplot() + stat_summary_2d(aes(x = num_nearby_blighted_parcels,
                                y = num_vacant_parcels,
                                z = blighted),
                             bins = 16) +
  ggtitle("Distribution of blightedness over
           num_nearby_blighted_parcels and num_vacant_parcels") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_fill_continuous(name = "mean\nblightedness")
```



I should emphasize that the above plots represent a subset of the buildings in the Hardest Hit Areas—reduced so as to have a base rate of blightedness (percentage across the dataset) of roughly 50%. (The actual rate of blightedness in these areas is roughly 2%.) Application of our simple decision-tree model, or any of the other models discussed in this report, would need to be done carefully so as to avoid base rate bias (see https://en.wikipedia.org/wiki/Base_rate_fallacy). Nevertheless, the fact that past building destruction stands out in this way as a predictor of blight may be a useful consideration to policymakers.